



CLOUD COMPUTING 2013

The Fourth International Conference on Cloud Computing, GRIDs, and
Virtualization

ISBN: 978-1-61208-271-4

May 27- June 1, 2013

Valencia, Spain

CLOUD COMPUTING 2013 Editors

Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany

Dariusz Król, Academic Computer Center CYFRONET - Krakow, Poland

Yong Woo Lee, University of Seoul, Korea

Aida Omerovic, SINTEF, Norway

CLOUD COMPUTING 2013

Foreword

The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2013), held between May 27 and June 1, 2013 in Valencia, Spain, continued a series of events intended to prospect the applications supported by the new paradigm and validate the techniques and the mechanisms. A complementary target was to identify the open issues and the challenges to fix them, especially on security, privacy, and inter- and intra-clouds protocols.

Cloud computing is a normal evolution of distributed computing combined with Service-oriented architecture, leveraging most of the GRID features and Virtualization merits. The technology foundations for cloud computing led to a new approach of reusing what was achieved in GRID computing with support from virtualization.

We take here the opportunity to warmly thank all the members of the CLOUD COMPUTING 2013 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to CLOUD COMPUTING 2013. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the CLOUD COMPUTING 2013 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that CLOUD COMPUTING 2013 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the areas of cloud computing, GRIDs and virtualization.

We are convinced that the participants found the event useful and communications very open. We hope that Valencia, Spain provided a pleasant environment during the conference and everyone saved some time to explore this historic city.

CLOUD COMPUTING 2013 Chairs:

CLOUD COMPUTING General Chair

Maria Dolores Cano Banos, Polytechnic University of Cartagena, Spain

CLOUD COMPUTING Advisory Chairs

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany
Yong Woo Lee, University of Seoul, Korea
Alain April, École de Technologie Supérieure - Montreal, Canada

CLOUD COMPUTING 2013 Industry/Research Chairs

Wolfgang Gentzsch, Senior HPC Consultant, Germany
Tony Shan, Keane Inc., USA
Donglin Xia, Microsoft Corporation, USA

CLOUD COMPUTING 2013 Research Institutes Chairs

Jorge Ejarque, Barcelona Supercomputing Center, Spain
Leslie Liu, IBM T.J Watson Research, USA

COULD COMPUTING 2013 Special Area Chairs

Virtualization

Toan Nguyen, INRIA, France

GRID

Jorge Ejarque, Barcelona Supercomputing Center, Spain
Javier Diaz, Indiana University, USA

Autonomic computing

Ivan Rodero, Rutgers the State University of New Jersey/NSF Center for Autonomic Computing, USA
Hong Zhu, Oxford Brookes University, UK

Service-oriented

Qi Yu, Rochester Institute of Technology, USA

Security

Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA

Platforms

Arden Agopyan, IBM Central & Eastern Europe, Russia, Middle East & Africa (CEE & MEA), Turkey
Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland

CLOUD COMPUTING 2013

Committee

CLOUD COMPUTING General Chair

Maria Dolores Cano Banos, Polytechnic University of Cartagena, Spain

CLOUD COMPUTING Advisory Chairs

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany

Yong Woo Lee, University of Seoul, Korea

Alain April, École de Technologie Supérieure - Montreal, Canada

CLOUD COMPUTING 2013 Industry/Research Chairs

Wolfgang Gentzsch, Senior HPC Consultant, Germany

Tony Shan, Keane Inc., USA

Donglin Xia, Microsoft Corporation, USA

CLOUD COMPUTING 2013 Research Institutes Chairs

Jorge Ejarque, Barcelona Supercomputing Center, Spain

Leslie Liu, IBM T.J. Watson Research, USA

CLOUD COMPUTING 2013 Special Area Chairs

Virtualization

Toan Nguyen, INRIA, France

GRID

Jorge Ejarque, Barcelona Supercomputing Center, Spain

Javier Diaz, Indiana University, USA

Autonomic computing

Ivan Rodero, Rutgers the State University of New Jersey/NSF Center for Autonomic Computing, USA

Hong Zhu, Oxford Brookes University, UK

Service-oriented

Qi Yu, Rochester Institute of Technology, USA

Security

Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA

Platforms

Arden Agopyan, IBM Central & Eastern Europe, Russia, Middle East & Africa (CEE & MEA), Turkey
Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland

CLOUD COMPUTING 2013 Technical Program Committee

Jemal Abawajy, Deakin University - Victoria, Australia
Imad Abbadi, University of Oxford, UK
Alain April, École de Technologie Supérieure - Montreal, Canada
Alvaro E. Arenas, Instituto de Empresa Business School, Spain
Marcelo Atenas, Polytechnic University of Valencia, Spain
Panagiotis Bamidis, Aristotle University of Thessaloniki, Greece
Luis Eduardo Bautista Villalpando, Autonomous University of Aguascalientes, Mexico
Ali Beklen, CloudArena, Turkey
Simona Bernardi, Centro Universitario de la Defensa / Academia General Militar - Zaragoza, Spain
Nik Bessis, University of Derby, UK
Stefano Bocconi, Delft University of Technology, Netherlands
William Buchanan, Edinburgh Napier University, UK
Ali R. Butt, Virginia Tech, USA
Massimo Canonico, University of Piemonte Orientale, Italy
Juan-Vicente Capella-Hernández, Universitat Politècnica de València, Spain
Carmen Carrión Espinosa, Universidad de Castilla-La Mancha, Spain
Simon Caton, Karlsruhe Institute of Technology, Germany
Hsi-Ya Chang, National Center for High-Performance Computing (NCHC), Taiwan
Rong N Chang, IBM T.J. Watson Research Center, USA
Antonin Chazalet, Orange, France
Shiping Chen, CSIRO ICT Centre, Australia
Ye Chen, Microsoft Corp., USA
Yixin Chen, Washington University in St. Louis, USA
Zhixiong Chen, Mercy College - NY, USA
William Cheng-Chung Chu(朱正忠), Tunghai University, Taiwan
Antonio Corradi, Università di Bologna, Italy
Marcelo Corrales, University of Hanover, Germany
Fabio M. Costa, Universidade Federal de Goiás (UFG), Brazil
Yuri Demchenko, University of Amsterdam, The Netherlands
Nirmit Desai, IBM Research - Bangalore, India
Edna Dias Canedo, Universidade de Brasília - UnB Gama, Brazil
Javier Diaz, Pervasive Technology Institute/Indiana University, USA
Qiang Duan, Pennsylvania State University Abington College, USA
Jorge Ejarque Artigas, Barcelona Supercomputing Center, Spain
Atilla Elçi, Suleyman Demirel University - Isparta, Turkey
Khalil El-Khatib, University of Ontario Institute of Technology - Oshawa, Canada
Erik Elmroth, Umeå University, Sweden
Mohamed Eltoweissy, Virginia Tech, USA
Javier Fabra, University of Zaragoza, Spain
Umar Farooq, Amazon.com - Seattle, USA
Maria Beatriz Felgar de Toledo, University of Campinas, Brazil
Sören Frey, University of Kiel, Germany

Wolfgang Gentsch, Senior HPC Consultant, Germany
Michael Gerhards, University of Applied Sciences, Germany
Katja Gilly, Miguel Hernandez University, Spain
Andres Gomez, Applications and Projects Department Manager Fundación CESGA, Spain
Nils Grushka, NEC Laboratories Europe - Heidelberg, Germany
Jordi Guitart, Universitat Politècnica de Catalunya - Barcelona Tech, Spain
Marjan Gusev, "Ss. Cyril and Methodius" University of Skopje, Macedonia
Weili Han, Fudan University, China
Haiwu He, INRIA, France
Neil Chue Hong, University of Edinburgh, UK
Kenneth Hopkinson, Air Force Institute of Technology - Dayton, USA
Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA
Anca Daniela Ionita, University "Politehnica" of Bucharest, Romania
César A. F. De Rose, Catholic University of Rio Grande Sul (PUCRS), Brazil
Qiang Duan, Pennsylvania State University, USA
Luca Foschini, Università degli Studi di Bologna, Italy
Song Fu, University of North Texas - Denton, USA
Spyridon Gogouvtis, National Technical University of Athens, Greece
Yi-Ke Guo, Imperial College London, UK
Richard Hill, University of Derby, UK
Uwe Hohenstein, Siemens AG, Germany
Benoit Hudzia, SAP Research, France
Shadi Ibrahim, INRIA Rennes - Bretagne Atlantique Research Center, France
Yoshiro Imai, Kagawa University, Japan
Ming Jiang, University of Leeds, UK
Xuxian Jiang, North Carolina State University, USA
Eugene John, The University of Texas at San Antonio, USA
Foued Jrad, KIT - Universität des Landes Baden-Württemberg, Germany
Carlos Juiz, Universitat de les Illes Balears, Spain
Sokratis K. Katsikas, University of Piraeus, Greece
Prashant Khanna, JK Lakshmipat University, Jaipur, India
Shinji Kikuchi, Fujitsu Laboratories Ltd., Japan
Tan Kok Kiong, National University of Singapore, Singapore
William Knottenbelt, Imperial College London - South Kensington Campus, UK
Sinan Kockara, University of Central Arkansas, USA
Joanna Kolodziej, University of Bielsko-Biala, Poland
Kenji Kono, Keio University, Japan
Arne Koschel, University of Applied Sciences and Arts - Hannover, Germany
George Kousiouris, National Technical University of Athens, Greece
Sotiris Koussouris, National Technical University of Athens, Greece
Nane Kratzke, Lübeck University of Applied Sciences, Germany
Heinz Kredel, Universität Mannheim, Germany
Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland
Hans Günther Kruse, Universität Mannheim, Germany
Eric Kuada, Aalborg University - Copenhagen, Denmark
Pierre Kuonen, College of Engineering and Architecture - Fribourg, Switzerland
Tobias Kurze, Karlsruher Institut für Technologie (KIT), Germany
Dharmender Singh Kushwaha, Motilal Nehru National Institute of Technology - Allahabad, India

Dimosthenis Kyriazis, National Technical University of Athens, Greece
Romain Laborde, University Paul Sabatier, France
Erwin Laure, KTH, Sweden
Alexander Lazovik, University of Groningen, The Netherlands
Grace Lewis, CMU Software Engineering Institute - Pittsburgh, USA
Jianxin Li, Beihang University, China
Kuan-Ching Li, Providence University, Taiwan
Maik A. Lindner, SAP Labs, LLC. - Palo Alto, USA
Maozhen Li, Brunel University - Uxbridge, UK
Xiaoqing (Frank) Liu, Missouri University of Science and Technology, USA
Xumin Liu, Rochester Institute of Technology, USA
H. Karen Lu, CISSP/Gemalto, Inc., USA
Ilias Maglogiannis, University of Central Greece - Lamia, Greece
Shikharesh Majumdar, Carleton University, Canada
Attila Csaba Marosi, MTA SZTAKI Computer and Automation Research Institute/Hungarian Academy of Sciences - Budapest, Hungary
Goran Martinovic, J.J. Strossmayer University of Osijek, Croatia
Philippe Massonet, CETIC, France
Michael Maurer, Vienna University of Technology, Austria
Reinhard Mayer, Universität Heidelberg, Germany
Aaron McConnell, University of Ulster Coleraine, UK
Andreas Menychtas, National Technical University of Athens, Greece
Jose Merseguer, Universidad de Zaragoza, Spain
Louise Moser, University of California - Santa Barbara, USA
Owen Molloy, National University of Ireland – Galway, Ireland
Claude Moulin, Technology University of Compiègne, France
Francesc D. Muñoz-Escóí, Universitat Politècnica de València, Spain
Hidemoto Nakada, National Institute of Advanced Industrial Science and Technology (AIST), Japan
Surya Nepal, CSIRO ICT Centre, Australia
Toàn Nguyễn, INRIA Grenoble Rhone-Alpes/ Montbonnot, France
Bogdan Nicolae, IBM Research, Ireland
P-O Östberg, Umeå University, Sweden
Alexander Paar, TWT GmbH Science and Innovation, Germany
Massimo Paolucci, DOCOMO Labs, Italy
Alexander Papaspyrou, Technische Universität Dortmund, Germany
Valerio Pascucci, University of Utah, USA
Aljosa Pasic, Atos Research, Spain
David Paul, University of Newcastle, Australia
Siani Pearson, Hewlett-Packard Laboratories, USA
Giovanna Petrone, University of Torino, Italy
Sabri Pllana, University of Vienna, Austria
Agostino Poggi, Università degli Studi di Parma, Italy
Jari Porras, Lappeenranta University of Technology, Finland
Thomas E. Potok, Oak Ridge National Laboratory, USA
Francesco Quaglia, Sapienza Univesita' di Roma, Italy
Rajendra K. Raj, Rochester Institute of Technology, USA
Christoph Reich, Hochschule Furtwangen University, Germany
Dolores Rexachs, University Autònoma of Barcelona (UAB), Spain

Sebastian Rieger, University of Applied Sciences Fulda, Germany
Sashko Ristov, "Ss. Cyril and Methodius" University of Skopje, Macedonia
Norbert Ritter, University of Hamburg, Germany
Philip Robinson, SAP Research - Belfast, UK
Ivanm Rodero, NSF Center for Autonomic Computing, Rutgers the State University of New Jersey - Piscataway, USA
Daniel A. Rodríguez Silva, Galician Research and Development Center in Advanced Telecommunications" (GRADIANT), Spain
Kyung Dong Ryu, IBM T.J. Watson Research Center, USA
Majd F. Sakr, Carnegie Mellon University in Qatar, Qatar
Iñigo San Aniceto Orbegozo, Universidad Complutense de Madrid, Spain
Elena Sanchez Nielsen, Universidad de La Laguna, Spain
Volker Sander, FH Aachen University of Applied Sciences, Germany
Gregor Schiele, Digital Enterprise Research Institute (DERI) at the National University of Ireland, Galway (NUIG), Ireland
Igor Sfiligoi, University of California San Diego-La Jolla, USA
Alan Sill, Texas Tech University, USA
Raül Sirvent, Barcelona Supercomputing Center, Spain
Luca Spalazzi, Università Politecnica delle Marche - Ancona, Italy
George Spanoudakis, City University London, UK
Jie Tao, Steinbuch Centre for Computing/Karlsruhe Institute of Technology (KIT), Germany
Orazio Tomarchio, University of Catania, Italy
Stefano Travelli, Developer at Cyntelix Corporation BV, Netherlands
Matteo Turilli, University of Oxford, UK
Raul Valin, Swansea University, Spain
Geoffroy R. Vallee, Oak Ridge National Laboratory, USA
Luis Vaquero, HP Labs Bristol, UK
Michael Gr. Vassilakopoulos, University of Central Greece - Lamia, Greece
Jose Luis Vazquez-Poletti, Universidad Complutense de Madrid, Spain
Salvatore Venticinquè, Second University of Naples - Aversa, Italy
Mario Jose Villamizar Cano, Universidad de los Andes - Bogotá, Colombia
Salvatore Vitabile, University of Palermo, Italy
Eugen Volk, High Performance Computing Center Stuttgart (HLRS) - Stuttgart, Germany
Andy Ju An Wang, Southern Polytechnic State University - Marietta, USA
Cho-Li Wang, University of Hong Kong, China
Lizhe Wang, Center for Earth Observation & Digital Earth - Chinese Academy of Sciences, China
Zhi Wang, Florida State University, USA
Martijn Warnier, Delft University of Technology, Netherlands
Mandy Weißbach, University of Halle, Germany
Philipp Wieder, Gesellschaft fuer wissenschaftliche Datenverarbeitung mbH - Goettingen (GWDG), Germany
John Williams, Massachusetts Institute of Technology, USA
Yong Woo Lee, University of Seoul, Korea
Christos Xenakis, University of Piraeus, Greece
Hiroshi Yamada, Tokyo University of Agriculture and Technology, Japan
Chao-Tung Yang, Tunghai University, Taiwan R.O.C.
Hongji Yang, De Montfort University (DMU) - Leicester, UK
Yanjiang Yang, Institute for Infocomm Research, Singapore

Jinhui Yao, CSIRO ICT Centre, Australia
Ustun Yildiz, University of California, USA
Qi Yu, Rochester Institute of Technology, USA
Jong P. Yoon, Mercy College - Dobbs Ferry, USA
Jie Yu, National University of Defense Technology (NUDT), China
Massimo Villari, University of Messina, Italy
Baokang Zhao, National University of Defence Technology, China
Zibin (Ben) Zheng, Shenzhen Research Institute, The Chinese University of Hong Kong, Hong Kong
Jingyu Zhou, Shanghai Jiao Tong University, China
Hong Zhu, Oxford Brookes University, UK
Wolf Zimmermann, University of Halle, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission to reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

An Architecture for a Heterogeneous Private IaaS Management System <i>Rodrigo Garcia-Carmona, Mattia Peirano, Juan C. Duenas, and Alvaro Navas</i>	1
A Look at Cloud Architecture Interoperability through Standards <i>Claus Pahl, Li Zhang, and Frank Fowley</i>	7
K Means of Cloud Computing: MapReduce, DVM, and Windows Azure <i>Lin Gu, Zhonghua Sheng, Zhiqiang Ma, Xiang Gao, Charles Zhang, and Yaohui Jin</i>	13
A Novel Cloud Hybrid Access Mechanism for Highly Sensitive Data Exchange <i>Elhadj Benkhelifa and Dayan A Novel Cloud Hybrid Access Mechanism for Highly Sensitive Data Exchange</i>	19
Eliciting Risk, Quality and Cost Aspects in Multi-cloud Environments <i>Victor Munte ?s-Mulero, Peter Matthews, Aida Omerovic, and Alexander Gunka</i>	25
Moonstone: A Framework for Accelerating Testing of Software <i>Atsuji Sekiguchi, Tomohiro Ohtake, Toshihiro Shimizu, Yuji Hotta, Taichi Sugiyama, Takeshi Yasuie, and Toshihiro Kodaka</i>	31
Challenges with Tenant-Specific Cost Determination in Multi-Tenant Applications <i>Anna Schwanengel and Uwe Hohenstein</i>	36
Community Clouds a centralized approach <i>Claudio Giovanoli and Stella Gatzju Grivas</i>	43
Using MapReduce to Speed Up Storm Identification from Big Raw Rainfall Data <i>Kulsawasd Jitkajornwanich, Upa Gupta, Ramez Elmasri, Leonidas Fegaras, and John McEnery</i>	49
A RESTful Approach for a Cloud Gateway <i>Chang Ho Yun, Jong Won Park, Hae Sun Jung, Yong Woo Lee, and Haeng Jin Jang</i>	56
Adaptive Multimedia Learning Delivered in Mobile Cloud Computing Environment <i>Aleksandar Karadimce and Danco Davcev</i>	62
Digital Signature as a Cloud-based Service <i>Wojciech Kinastowski</i>	68
Cloud-Enabled Scaling of Event Processing Applications <i>Irina Astrova, Arne Koschel, and Ahto Kalja</i>	73

Cloud Computing Services Potential Analysis <i>Giuseppe Ercolani</i>	77
Context-Aware Data-Flow in the Cloud <i>Mandy Weissbach, Wolf Zimmermann, and Welf Lowe</i>	81
A Coordinated Reactive and Predictive Approach to Cloud Elasticity <i>Laura Moore, Kathryn Bean, and Tariq Ellahi</i>	87
Elastic-TOSCA: Supporting Elasticity of Cloud Application in TOSCA <i>Rui Han, Moustafa M. Ghanem, and Yike Guo</i>	93
OpenStack Cloud Security Vulnerabilities from Inside and Outside <i>Sasko Ristov, Marjan Gusev, and Aleksandar Donevski</i>	101
Seamlessly Enabling the Use of Cloud Resources in Workflows <i>Michael Gerhards, Volker Sander, and Adam Belloum</i>	108
Collaborative Autonomic Resource Management System for Mobile Cloud Computing <i>Ahmed Khalifa and Mohamed Eltoweissy</i>	115
SLA Template Filtering: A Faceted Approach <i>Katerina Stamou, Verena Kantere, and Jean-Henry Morin</i>	122
CPU Utilization while Scaling Resources in the Cloud <i>Marjan Gusev, Sasko Ristov, Monika Simjanoska, and Goran Velkoski</i>	131
Evaluating Computation Offloading Trade-offs in Mobile Cloud Computing: A Sample Application <i>Jorge Luzuriaga, Juan Carlos Cano, Carlos Calafate, and Pietro Manzoni</i>	138
Massively Scalable Platform for Data Farming Supporting Heterogeneous Infrastructure <i>Dariusz Krol, Michal Wrzeszcz, Bartosz Kryza, Lukasz Dutka, and Jacek Kitowski</i>	144
Fuzzy Controlled QoS for Scalable Cloud Computing Services <i>Stefan Frey, Claudia Luthje, Vitali Huwwa, and Christoph Reich</i>	150
Scalable Store and Forward Messaging <i>Ahmed El Rheddane, Noel De Palma, and Alain Tchana</i>	156
Towards a Method for Decision Support in Multi-cloud Environments <i>Aida Omerovic, Victor Munte-Mulero, Peter Matthews, and Alexander Gunka</i>	162
A DSL For Logistics Clouds	169

Bill Karakostas and Takis Katsoulakos

Deploying a Multipoint Control Unit in the Cloud: Opportunities and Challenges 173
Alvaro Alonso, Pedro Rodriguez, Joaquin Salvachua, and Javier Cervino

An Approach to Assure QoS of Machine Translation System on Cloud 179
Pawan Kumar, Rashid Ahmad, Banshi D Chaudhary, and Mukul K Sinha

Defining Intercloud Federation Framework for Multi-provider Cloud Services Integration 185
Marc X. Makkes, Canh Ngo, Yuri Demchenko, Rudolf Stijkers, Robert Meijer, and Cees de Laat

A Cloud Platform to support User-Provided Mobile Services 191
Vincenzo Catania, Giuseppe La Torre, Salvatore Monteleone, and Daniela Panno

CloudState: End-to-end WAN Monitoring for Cloud-based Applications 195
Aaron McConnell, Gerard Parr, Sally McClean, Philip Morrow, and Bryan Scotney

Transparent Access on Encrypted Data Distributed over Multiple Cloud Infrastructures 201
Luca Ferretti, Michele Colajanni, Mirco Marchetti, and Adriano Enrico Scaruffi

Forensics-as-a-Service (FaaS): Computer Forensic Workflow Management and Processing Using Cloud 208
Yuanfeng Wen, Xiaoxi Man, Khoa Le, and Weidong Shi

Fuzzy Subtractive Clustering Based Prediction Approach for CPU Load Availability 215
Kadda Beghdad Bey, Farid Benhammadi, and Faouzi Sebbak

On the analytical characterization of a real life Virtual Network Function: the Italtel Virtual Session Border Control 221
Sergio Montagna and Pietro Paglierani

Trusted Computing on Heterogeneous Embedded Systems-on-Chip with Virtualization and Memory Protection 225
Marcello Coppola, Miltos Grammatikakis, George Kornaros, and Alexander Spyridakis

Using Cloud-based Resources to Improve Availability and Reliability in a Scientific Workflow Execution Framework 230
Sergio Hernandez, Javier Fabra, Pedro Alvarez, and Joaquin Ezpeleta

Eliciting Risk, Quality and Cost Aspects in Multi-cloud Environments 238
Victor Munte ?s-Mulero, Peter Matthews, Aida Omerovic, and Alexander Gunka

Towards a Method for Decision Support in Multi-cloud Environments 244
Aida Omerovic, Victor Munte-s-Mulero, Peter Matthews, and Alexander Gunka

An Architecture for a Heterogeneous Private IaaS Management System

Rodrigo García-Carmona, Mattia Peirano, Juan C. Dueñas, Álvaro Navas

Departamento de Ingeniería de Sistemas Telemáticos

ETSI Telecomunicación, Universidad Politécnica de Madrid

Madrid, Spain

rodrigo@dit.upm.es, peirano.m@gmail.com, jcduenas@dit.upm.es, anavas@dit.upm.es

Abstract—Cloud computing and, more particularly, private IaaS, is seen as a mature technology with a myriad solutions to choose from. However, this disparity of solutions and products has instilled in potential adopters the fear of vendor and data lock-in. Several competing and incompatible interfaces and management styles have given even more voice to these fears. On top of this, cloud users might want to work with several solutions at the same time, an integration that is difficult to achieve in practice. In this paper, we propose a management architecture that tries to tackle these problems; it offers a common way of managing several cloud solutions, and an interface that can be tailored to the needs of the user. This management architecture is designed in a modular way, and using a generic information model. We have validated our approach through the implementation of the components needed for this architecture to support a sample private IaaS solution: OpenStack.

Keywords—private IaaS; cloud management; management architecture; cloud interoperability; OpenStack.

I. INTRODUCTION

Cloud computing has, during recent years, gained traction both in the enterprise world and the academia. Among all possible cloud service models, one in particular, the private IaaS, has experienced an exceptional growth in the number of solutions available [1]. Several competing products, both open-sourced and proprietary, are contending for attaining relevance and are constantly trying to surpass each other. This fact creates a climate in which the user has the possibility of choosing among a huge array of possible solutions.

However, this ample offer of private IaaS cloud technologies also involves an important drawback: each one is managed using different abstractions (sometimes for the same concepts) and through different management interfaces. This is aggravated by the use of different technologies for these interfaces. This presents problems for a more widespread adoption of private IaaS cloud computing, since potential users fear of being locked-in with a particular solution that falls behind the others in terms of features or support. The infrastructure's owner should be able to change his previously chosen technology for private IaaS without having to modify the management interfaces, a fact that sometimes incur in expensive retraining and even more expensive errors during production deployments. Vendor and

data lock-in are considered two of the bigger factors that hinder the development of cloud computing [2] [3].

Moreover, an enterprising private IaaS user could have the desire of deploying two or more different cloud offerings, leveraging the strong features of each for a solution better tailored to his or her specific needs. In this situation the user would benefit greatly from an integrated management interface that could wrap this mixture of products in a uniform whole. Another reason for deploying two private IaaS solutions at the same time is to compare their performance side to side or ease the migration from one to the other.

With this problem in mind, we propose a generic management system for private IaaS clouds, decoupled from any particular solution but able to work with all of them, and using a set of common abstractions that could be translated to the specifics of each targeted product. This management system should also be able to provide its interface through the use of different technologies (like a REST web service, a command line, or a web page), to better suit the user's needs.

To achieve this goal, we have defined a modular architecture, in which components for both different private IaaS technologies and interfaces can be developed and plugged as needed. In this paper, we present this architecture, validating it through the implementation of the components needed for the management of OpenStack clouds.

The next section of this paper features a brief view of existing private IaaS management solutions and related research. After it, in Section 3, we show the general architecture of the proposed management system. Section 4 covers the specifics related to the OpenStack implementation of the management system. Finally, the last section of this paper summarizes our achievements and what was learned in the experience, while also exposing some possible future lines of work.

II. PRIVATE IAAS MANAGEMENT SOLUTIONS

There are a multitude of management interfaces for cloud infrastructure and storage services. Every solution has at least one, and they can be found in multiple shapes: command-line tools, locally installed management applications with a GUI, web browser extensions, online tools, etc.

All private IaaS solutions offer their own management interfaces, tailored to its specific needs and features, and rarely able to interact with other solutions, or even other cloud deployments of the same solution. The only exceptions to this fact are not by design: it is just that some private IaaS solutions try to replicate the same capabilities and abstractions offered by more popular public offerings, like Amazon AWS. And, in doing so, they develop very similar or even identical interfaces. Among these the more extended are Eucalyptus, Nimbus, OpenNebula and OpenStack. Comparisons between the IaaS solutions usually include a comparison between their management interfaces [4]–[6]. All considered, these management systems are usually solutions particularized to work only with a specific cloud technology, and they are not compatible with others.

Third party solutions for managing private IaaS clouds also exist, some of them suited to just one cloud solution [7], while others support several technologies. KOALA (Karlsruhe Open Application (for) cLoud Administration) is a web based application able to manage and control AWS compatible cloud services [8]. It allows to work with a large variety of services of various public and private cloud providers in a seamless and transparent way [9]. KOALA innovative characteristic is that it does not require a local installation since itself could be deployed in the cloud. The user interface allows customers to start, stop and monitor their instances or volumes in various cloud infrastructure regions, and have access to the console output of virtual machines. KOALA supports S3, Google Storage and Walrus storage services.

Scalr is a cross platform, cloud management software that provides auto scaling disaster recovery and server management [10]. It is open source, available at Google Code but a hosted version is available as paid service. The manager is able to scale the virtual infrastructure according to the load. Scaling strategies could be based on CPU, RAM, disk, network or date. The latter can be useful in case of an increase in traffic is expected, like during scheduled public events. The code is distributed under Apache 2 license.

Puppet is an IT automation software that helps system administrators manage infrastructure throughout its lifecycle, easing the automation of the repetitive tasks [11]. This configuration management tool is written in Ruby and provides some specific modules for cloud management. The software is distributed for free with some utilization restrictions. The paid version offers a solution without limits.

Finally, there are open source initiatives like Libcloud [12], jcloud [13] or deltacloud [14], but they are more concerned with the management of public IaaS providers, even if they include support for some private IaaS solutions. They are centred in the management of virtual instances, and do not give much attention to the physical underlying infrastructure while doing so. Also, they are limited to a specific programming language or interface.

As can be seen, none of them offers a true solution-agnostic view. The academia has produced systems that offer a generic management interface that could potentially be adapted to any particular product. However, they present an important drawback: their interface is fixed, since their generic model and interface are built with a specific technology in mind, like REST [15] or SOAP [16] web services. Therefore, it is difficult to extend them to provide other kind of interfaces, like command-line tools or web pages.

In the end, the fact that the existing management interfaces are focused in exposing low-level infrastructure elements makes working with several solutions or migrating from one to another a complex affair, since there is no exact match between the features and abstractions being used by each one [17]. This is mainly because they are focused in the management of resources, not applications.

III. MANAGEMENT SYSTEM ARCHITECTURE

The first step for developing a technology-independent management system is to have a generic information model that covers all the elements that need to be managed in a private IaaS. We have developed a model that bridges the gap between the applications deployed into a cloud and the actual resources allocated to them. This way the user of the management infrastructure can have a view of the big picture. This model has already been accepted for publication [18] and, therefore, will not be the topic of this paper. However, its elements related to the IaaS resources are summarized in Figure 1. They are mostly self-explanatory, but it is important to note the fact that all elements are *Resources* (and because of that are managed uniformly), and that there are relationships between the physical and virtual *Resources*, enabling traceability. The *VirtualMachine* element represents the VMs managed.

This model is able to cover every solution, but none of them are able to work with it without modification; a transformation from this model to the internal representation specific for each cloud technology is necessary. Similarly, a translation between the generic management actions and the operations enabled by each cloud technology must be provided. This two tasks are fulfilled by the management system, and these needs determine its architecture, which is shown in Figure 2. The management system is divided in three separate layers:

- The topmost layer, named *Control Layer*, is responsible of providing an interface to the outer world, and makes use of the set of services provided by the *Management Layer* to execute the management actions.
- Under it lies the *Management Layer*, which is the heart of the system. This layer is composed by a set of interfaces that define the capabilities of the system and one or more implementations of them, providing management over specific functional areas of a particular private IaaS cloud technology.

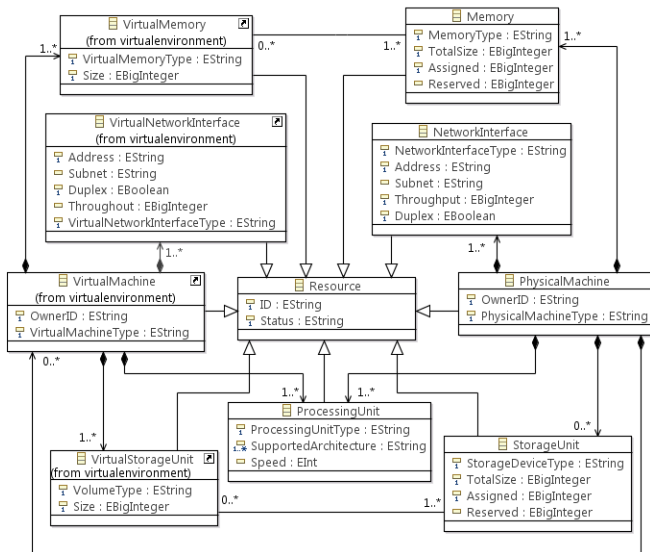


Figure 1. Generic Information Model

- The lowest layer is the *Client Layer*, which connects the system to the cloud solution itself.

The target of this division is to support a) several interfaces for the same management system, b) several cloud solutions, and c) several client technologies for the same cloud solution. We will analyse each of this layers in detail in the following subsections.

A. Control Layer

This layer is divided in 2 other: the *Control Interface* and the *High-Level Managers*.

The *Control Interface* is the outward interface that connects the system to the manager. This layer is designed to be interchangeable and support several interface implementations at the same time. The motivation for this schema is that different users could prefer different management interfaces. Moreover, the user does not need to be a human operator at all, it can be other system, and therefore there is a need for interfaces more suited to this task. Samples of these interfaces could be command-line tools or an administration web page for a human operator, and a web services interface for an autonomic system that keeps care of the private cloud infrastructure.

Under the *Control Interface* lies the *High Level Managers*, which intermediate between the aforementioned interface implementations and the *Management Layer*. The *High Level Managers* sublayer provides to the *Control Interface* two components: an *Infrastructure Manager* for controlling the cloud through a set of management actions defined in our information model (and therefore technology-agnostic), and an *Authentication Manager* in charge of monitoring and enforcing the security model for the private cloud. This component deserves to be separated from the *Infrastructure*

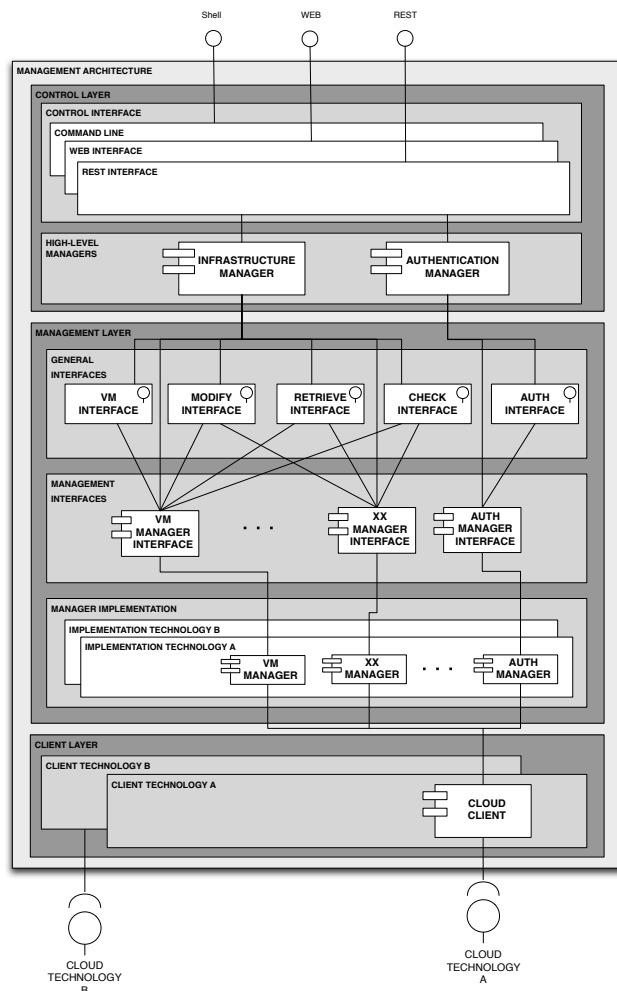


Figure 2. Management System Architecture

Manager because of the high importance of security in a cloud environment, where there could be multiple tenants, and the different implementations of the security system that each cloud solution features. It is necessary to have a common interface that abstracts from this differences and complexities.

B. Management Layer

This layer is again divided in 3 other: *General Interfaces*, *Management Interfaces* and *Manager Implementations*.

The *General Interfaces* sublayer offer three interfaces which provide management primitives that can be applied to every *Resource* as defined in our information model. These primitives are very simple, and the more complex management activities are built upon them:

- *Retrieve*, which encompasses the actions to obtain data from the cloud: *getList* and *getSpecific*.
- *Modify*, which includes the actions tailored to modify

the state of the cloud: *create* and *delete*. Modification is just a deletion followed by a creation.

- *Check*, used to check if an element already exists in the cloud. It includes just one action of the same name: *check*. This feature is needed to be kept up to date with the state of the environment.

The *General Interfaces* also include two other interfaces:

- *VM*, for controlling Virtual Machines, one of the most important elements of our model. It provides the actions *Suspend*, *Resume*, *Resize* and *Migrate*.
- *Authentication*, which does not perform actions over entities at all. Instead, it controls who can perform them and under what circumstances.

Under the *General Interfaces* lie the *Management Interfaces*. This sublayer particularises the primitives of the upper level to specific *Resources* of the infrastructure. We have defined the following interfaces, whose responsibility can be easily inferred from their names: *VM Manager*, *Image Manager*, *Virtual Appliance Manager*, *Compute Manager*, *Network Manager*, *Host Manager*, *Key Manager*, *Group Manager*, *Tenant Manager* and *Authentication Manager*. These interfaces extend one or more of the *General Interfaces* level interfaces.

These interfaces are in turn implemented in the remaining sublayer: *Manager Implementation*. Since this implementation has to be tailored for each cloud technology, it is here where the adaptation between our generic information model and the solution’s specific one is performed. Therefore, we must implement the *General Interfaces* for each cloud technology we desire to support. If this is done correctly several solutions could be used at the same time, without each one being conscious of the others.

C. Client Layer

Finally, in the *Client Layer* is where the adaptation between the management system and the real infrastructure is realised. This is done through one or more modules, each designed to work with a particular access technology and cloud solution. These modules connect each with the appropriate *Manager Implementation* and interact with the private IaaS itself.

IV. OPENSTACK MANAGEMENT

To validate our proposal, we decided to create an implementation of the management architecture able to interface with one of the existing private IaaS solutions: OpenStack. We chose this product because its open sourced nature would help us in solving any problems that might arise. On top of that, it is a relatively mature solution that is seeing intense development at the moment.

To adapt a cloud solution to our proposal, we had to complete three tasks: 1) specify a translation between our

TABLE I. MAPPING BETWEEN INFORMATION MODELS

Generic Model	OpenStack Model
Virtual Instance	Virtual Machine (<i>Server</i>)
Virtual Appliance	Image
	Flavor
	Name
	Security Group
	Metadata
Virtual Memory	Flavor (<i>RAM</i>)
Virtual Storage	Flavor (<i>Disk</i>)
	Volume
Processing Unit	Flavor (<i>CPU</i>)
Virtual Network Iface.	Virtual Network
Owner	Tenant
Physical Machine	Host
-	User
Initial Configuration	Key Pair
-	Floating IPs

generic information model and the solution’s own, 2) develop a corresponding set of *Manager Implementations*, and 3) create at least one interface for the *Client Layer*.

Table I shows the mapping between our generic information model and the OpenStack representation that we developed. Each column includes some elements that are not present in the other. For example the *Virtual Appliance* element is not defined in the OpenStack environment. However, a correspondence between several disparate elements of the OpenStack model to it can be made. Even if they are placed in different relative places inside their own model, most components of every private IaaS can be traced to elements of our information model. There are exceptions, though, like the *User* and *Floating IPs*. But in these cases the culprits are always relevant to specifics of each implementation, and can be managed inside the *Manager Implementation* sublayer without hampering the view of the environment.

After the mapping is defined the grunt work of the adaptation to OpenStack lies in the development of the *Manager Implementations* and *Client Layer*. Most of this work is just programming and is no relevant to this text, but one aspect of it took a special importance during the process: an OpenStack component named Quantum. Quantum provides advanced high level network management, enabling the definition of L2 and L3 network topologies and multiple networks across different VMs and tenants. Quantum was still in its early stages of development while we were validating our proposal and, in fact, there was no complete support for it in the OpenStack interface. Therefore, the development of our *ClientLayer* involved modifying the OpenStack code itself. Also, Quantum forced us to rethink some aspects of the network-related elements in our model, to support its more advanced capabilities. The *Client Layer* was designed to use the OpenStack REST interface.

To illustrate how the different managers interact among themselves, encompassing both operations over the generic information model and the actual infrastructure, we repro-

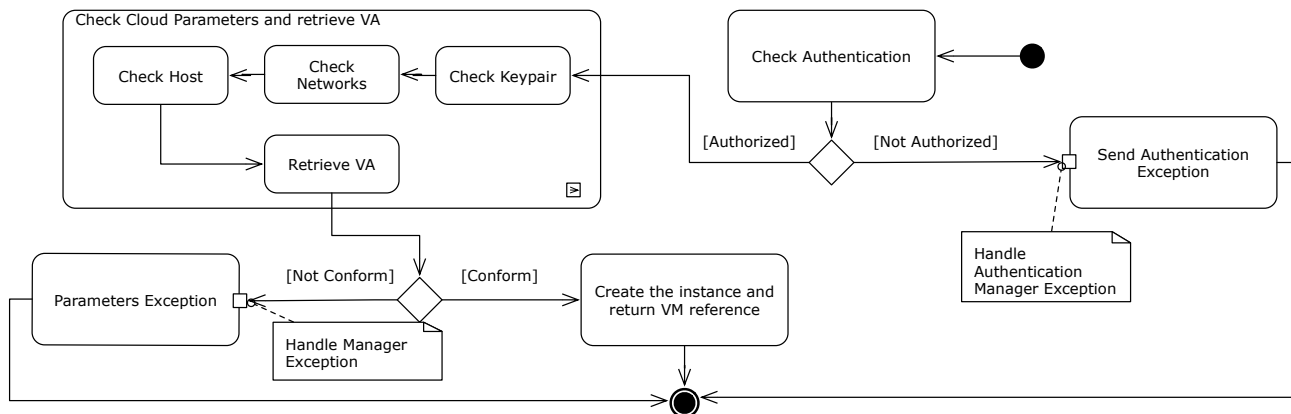


Figure 3. Starting a Virtual Machine

duce here a sample activity, the process of starting a VM (see Figure 3). This activity involves an authentication check (to be sure that the tenant can perform the creation), a state of the environment check (to ensure that the intended action is feasible), and the start of the VM action itself.

To complement our work with OpenStack and have a complete and usable management system, we have also developed two *Control Interfaces*: a REST service and a web page. The latter is intended to be used by a human operator and the former is connected to an autonomic system of our own creation. The whole system was written in Java.

V. CONCLUSION AND FUTURE WORK

In this paper, we have established the need for a management architecture for private IaaS clouds that support several solutions (to avoid data and vendor lock-in), working alone or together, and several user interfaces. To this end, we have proposed the use of a generic information model that captures all the relevant information for the infrastructure, and a modular architecture that can be adapted to fit several IaaS products and needs. We have detailed this architecture, making a special emphasis in how it achieves the desired results. In doing that, we have explained its three layers and how they fit inside the big picture.

To validate our approach, we have developed and tested a sample implementation with support for one cloud solution (OpenStack) and two interfaces (REST services and a web page). In this text, we have explained the aspects of this implementation more relevant to the development of the modules needed to support other cloud technologies.

In this process, we had to confront the realities of actual products and how to apply our proposal to them. This gave us some interesting realizations, like the pressing need for a more fine-grained network configuration support in clouds (already established in the literature [19]), and how to use

the improved network customization features offered by solutions like Quantum to achieve this end.

Therefore, our next efforts will be focused on this topic. In the future, we also want to develop support for at least another cloud solution: This way we will be more able to test a federation of several private clouds and achieve a true working common management interface for multiple technologies. This matter will include the difficult topic of deciding where to physically put the management interface itself when working with several infrastructures. This integration will definitely test if our generic approach to security is suitable for use with different cloud solutions at the same time.

Finally, another line of work we want to follow is the application of our architecture to the management of public cloud offerings, since the interest in hybrid clouds that mix public and private IaaS is steadily growing.

REFERENCES

- [1] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet Computing*, IEEE, vol. 13, no. 5, sept.-oct. 2009, pp. 14–22.
- [2] N. Leavitt, "Is cloud computing really ready for prime time," *Computer*, vol. 42, no. 1, 2009, pp. 15 –20.
- [3] M. Armbrust et al., "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, Apr. 2010, pp. 50 –58.
- [4] A. Lonea, D. Popescu, and O. Prosteian, "A survey of management interfaces for eucalyptus cloud," in *Applied Computational Intelligence and Informatics (SACI)*, 2012 7th IEEE International Symposium on, may 2012, pp. 261 –266.
- [5] S. Wind, "Open source cloud computing management platforms: Introduction, comparison, and recommendations for implementation," in *Open Systems (ICOS)*, 2011 IEEE Conference on, sept. 2011, pp. 175 –179.

- [6] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, "Comparison of open-source cloud management platforms: Openstack and opennebula," in *Fuzzy Systems and Knowledge Discovery (FSKD)*, 2012 9th International Conference on, may 2012, pp. 2457–2461.
- [7] L. Xu and J. Yang, "A management platform for eucalyptus-based iaas," in *Cloud Computing and Intelligence Systems (CCIS)*, 2011 IEEE International Conference on, sept. 2011, pp. 193–197.
- [8] C. Baun, M. Kunze, and V. Mauch, "The koala cloud manager: Cloud service management the easy way," in *Cloud Computing (CLOUD)*, 2011 IEEE International Conference on, july 2011, pp. 744–745.
- [9] C. Baun and M. Kunze, "The KOALA cloud management service: a modern approach for cloud infrastructure management," in *Proceedings of the First International Workshop on Cloud Computing Platforms*, ser. CloudCP '11. New York, NY, USA: ACM, 2011, p. 1:1–1:6.
- [10] "Scalr," <http://code.google.com/p/scalr/>, retrieved: 28th March 2013. [Online]. Available: <http://code.google.com/p/scalr/>
- [11] "Puppet labs: IT automation software for system administrators," <http://puppetlabs.com/>, retrieved: 28th March 2013. [Online]. Available: <http://puppetlabs.com/>
- [12] "Apache libcloud," <http://libcloud.apache.org/>, retrieved: 28th March 2013. [Online]. Available: <http://libcloud.apache.org/>
- [13] "jcloud," <http://www.jclouds.org/>, retrieved: 28th March 2013. [Online]. Available: <http://www.jclouds.org/>
- [14] "Apache deltacloud," <http://deltacloud.apache.org/>, retrieved: 28th March 2013. [Online]. Available: <http://deltacloud.apache.org/>
- [15] H. Han et al., "A restful approach to the management of cloud infrastructure," in *Cloud Computing, 2009. CLOUD '09*. IEEE International Conference on, sept. 2009, pp. 139–142.
- [16] Z. Lu, J. Wu, and W. Fu, "A novel cloud-oriented ws-management-based resource management model," in *Web Services (ICWS)*, 2010 IEEE International Conference on, july 2010, pp. 676–677.
- [17] T. Harmer, P. Wright, C. Cunningham, J. Hawkins, and R. Perrott, "An application-centric model for cloud management," in *Services (SERVICES-1)*, 2010 6th World Congress on, july 2010, pp. 439–446.
- [18] R. García-Carmona, F. Cuadrado, A. Navas, A. Celorio, and J. Dueñas, "Multi-level monitoring approach for the dynamic management of private iaas platforms," *Journal of Internet Technology*, vol. Special Issue on Dynamic Intelligence for Sustainable Computing, no. to appear, 2013, p. to appear.
- [19] J. Wickboldt, L. Granville, F. Schneider, D. Dudkowski, and M. Brunner, "A new approach to the design of flexible cloud management platforms," in *Network and Service Management (CNSM)*, 2012 8th International Conference on, oct. 2012, pp. 155–158.

A Look at Cloud Architecture Interoperability through Standards

Claus Pahl, Li Zhang and Frank Fowley

School of Computing, Dublin City University, Dublin, Ireland

Email: [cpahl|lzhang|ffowley]@computing.dcu.ie

Abstract—Enabling cloud infrastructures to evolve into a transparent platform while preserving integrity raises interoperability issues. How components are connected needs to be addressed. Interoperability requires standard data models and communication encoding technologies compatible with the existing Internet infrastructure. To reduce vendor lock-in situations, cloud computing must implement universal strategies regarding standards, interoperability and portability. Open standards are of critical importance and need to be embedded into interoperability solutions. Interoperability is determined at the data level as well as the service level. Corresponding modelling standards and integration solutions shall be analysed.

Keywords - Cloud Architecture; Interoperability; Standards.

I. INTRODUCTION

Enabling cloud infrastructures to evolve into a transparent platform while preserving integrity raises interoperability issues [2], [5]. Interoperability requires standard data models and communication encoding technologies compatible with the existing Internet infrastructure. The need to scale and provide cost-effective time-to-market. Public cloud services are available to the public and owned by an organisation selling cloud services, e.g. Microsoft or Amazon are major providers. Hybrid clouds are an integrated cloud services arrangement that includes provision of compute resources from more than one source (e.g. either private or public). Hybrid architectural models may be vertically partitioned (e.g. data stored privately) or horizontally partitioned (e.g. using public cloud to prototype a new device view of a service in parallel with an existing implementation). These architectural scenarios define the need for interoperability solutions if flexible composition, migration and portability are sought [1], [3].

To reduce vendor lock-in situations, cloud computing must implement universal strategies regarding standards, interoperability and portability. Open standards are of critical importance and need to be embedded into interoperability solutions [9], [10]. Standardisation efforts linked with intelligent processing techniques shall be given particular attention. Interoperability is determined at the data level as well as the service level [11], [4], [13]. Corresponding modelling standards and integration solutions shall be analysed.

The objectives of this investigation include the review of relevant standards for cloud architecture interoperability (looking at their background, usage and analysing their importance for this context) and analysing the overall maturity of the technology and determining current trends and shortfalls.

We start with an architectural scenario, the definition of stakeholders and interoperability concerns in Section 2. In

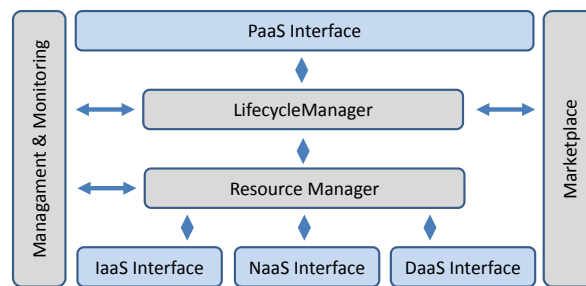


Fig. 1. Layered Architecture for Cloud-based Software Components.

Section 3, we categorise existing standards and review a selection, before ending with some discussions.

II. CLOUD ARCHITECTURE AND INTEROPERABILITY

A. Cloud Architecture

A cloud architectural framework consists of the classical three cloud layers infrastructure (IaaS), platform (PaaS and software (SaaS) as service-oriented offerings [2], [5], [6]. In addition, we can differentiate between (hardware or software) resources provided in a traditional way, and the . . . as a Service version of them, which considers virtualization, multi-tenancy and elasticity as the concerns [3]. A platform product can be deployed over IaaS (or a network provided as a service NaaS) or over real hardware infrastructure. A platform product can be offered as a Service (PaaS) for the application layer. The application software can be deployed either on top of a platform product (cloud-less), or making use of platform services (PaaS). Finally, an application product can be offered as a Service (SaaS) for external customers.

Different usage models can be derived from the combinations of the layers. We take into account that some software (or hardware) is provided as a service, other components are directly interfaced: application over a platform, SaaS over a platform, or pure platform over IaaS. These different usage models rely on interoperability solutions. Some are service-based abstractions (APIs) that need, consequently, to be aligned with common (Internet/Web-based) service description, modelling and composition standards. However, also more technology (or layer) specific standards are also important.

A stakeholder can play more than one role within the platform scenario according to the usage. For instance, a Software Provider in the Application Software layer can be at the same time a PaaS or Platform Software customer. Service providers

and users rely more on service-related standards, whereas non-service interfaces for software providers and developers might be more layer-specific.

B. Stakeholder Roles

Different stakeholders can be associated with the architectural scenarios. These can be categorised into roles that reflect their activities and needs. These roles are based on suggestions from the EU FP7 Projects SLA@SOI (for general roles in the services context) and 4CaaS (for cloud-specific roles).

A Service Provider (or application provider) supplies services to one or more internal or external customers. A Software Manager defines software-based services, takes care of their management (business focus) and administration (technical focus) A Service Aggregator is a reseller that contracts and aggregates services or applications by third parties in order to create a new ones A Service Maintainer maintains a service after it has been deployed A Context Provider provides context information about the underlying infrastructure components (e.g. telecoms or sensor network) A Data Provider owns, manages and provides data to a service

A Cloud Provider is a resource provider that provides an integrated platform or infrastructure services based on possibly heterogeneous cloud offerings. A Platform Provider offers a technical platform to Service Provider to host their software services. A Platform Manager manages a platform from a business perspective. An Infrastructure Provider is a cloud IaaS provider. An Infrastructure Manager measures and controls infrastructure properties.

A Software Provider produces software which might be used by a Service Provider to assemble services. In this context, a Software Designer designs/develops the architecture and components of a specific SLA-based application.

A Service Customer orders services and defines and agrees service-level targets (SLA). A Service Consumer is the person who actually consume/use the provided services.

C. Interoperability Concerns

Interoperability concerns arise in different situations. Interoperability between cloud layers needs standardised APIs to allow higher cloud layers to link to a range of services provided at the lower layers, e.g. platform implementations to uniformly link to IaaS offerings. Roles of importance are service provider and service user. Interoperability within layers needs suitable standards to allow components in a layer to interact and be exchangeable. Non-service interactions need to be supported, e.g. where, as explained in the third scenario above, a Software Developer combines different platforms in the development of a new system. Figure 1 indicates some of these components and their connectivity for the infrastructure and platform concerns. The architecture in Figure 1 should only be indicative of these concerns, but captures some agreed components in this context.

III. INTEROPERABILITY-RELATED STANDARDS

Standards are necessary to consolidate efforts in a technology domain and to enable interoperability. An overview and

a categorisation of standards relevant to interoperability in the cloud computing context that we cover here is:

- Web services: WSDL (description), SOAP (protocol), WS-BPEL (composition), UDDI (repository)
- Service modelling: Open-SCA (service composition and interaction), USDL/SoaML/CloudML (multi-view services), EMMML (mashups)
- Service interfaces: OCCI (infrastructure management), CIMI (infrastructure management), EC2 (de-facto standard), TOSCA (portability), CDMI (data)
- Infrastructure: OVF (virtual machines); specific concerns: memcached (data caching), VEPA (network)
- Security: OAuth, SCAP

For each standard, we provide background about origins, support and purpose, the intended usage, and an analysis of the relevance for interoperability considerations. Providing a comprehensive overview of all standards is not the objective. We have singled out those that represent specific aspects well.

A. Core Web Services Standards

Service-based provision needs Web services alignment. Thus, relevant standards are SOAP, WSDL, WS-BPEL and UDDI (not discussed due to space considerations). As all cloud layers (infrastructure, platform, software, processes) can be provided in an ...-as-a-service form, these classical services standards form the foundation of cloud interoperability. Although not standardised as such (and thus not covered here), RESTful services have become a similar, more lightweight major architectural style for services.

B. Service Modelling and Interface Standards

We separate more advanced modelling and interface description standards from the core Web services standards in this context, i.e. WSDL and WS-BPEL, as the ones covered here have not gained as much recognition.

Open Composite Services Architecture (Open-CSA)

The OASIS Open Composite Services Architecture (CSA) specifications provide standards to simplify SOA application development [21]. OASIS brought together vendors and users from to collaborate on the development and adoption of the Service Component Architecture (SCA) and Service Data Objects (SDO) families of specifications. **Usage:** As an example, the SCA Assembly Model is a framework to describe service coordination and interaction that ties in service composition with common software architecture concerns. **Analysis and Recommendation:** The CSA standards can be utilised as is or can serve as input for any composition and assembly language for interoperability concerns. It can serve a guide for the specification of services and cloud configurations. It can also facilitate the development of visual tools for assembling of components and service references during application design.

The specifications on the SCA Assembly Model are very relevant for interoperability. Application development using SCA should result in the following advantages. It promotes decoupling of application business logic from the details of the invoked services. Target services in a range of languages

(like C++, Java, and PHP) are supported. Different communications constructs including One-Way, Asynchronous, Call-Return, and Notification are considered. Legacy components or services, accessed as Web Services, EJB, JMS, JCA, RMI, or CORBA can be included. Quality of Service requirements are considered, such as security, transactions and the use of reliable messaging

Data interoperability is an equally important concern in Open-CSA. Data could be represented in Service Data Objects.

The value proposition of SCA is, therefore, the flexibility for composite applications to incorporate reusable components in an SOA programming style. The overhead of business logic concerns regarding platforms, infrastructure, plumbing, policies and protocols are removed.

Enterprise Mashup Markup Language (EMML) Enterprise Mashups combine and remix data from databases, spreadsheets, websites, Web Services, RSS/Atom feeds, and unstructured sources that deliver actionable information. The Open Mashup Alliance (OMA) is in charge of the Enterprise Mashup Markup Language (EMML) [20]. It can support enterprise mashup implementations, improve mashup portability of mashup designs, and increase the interoperability of mashup solutions. OMA provides an EMML schema and a reference runtime environment as the technology framework. **Usage:** EMML is a Domain Specific Language (DSL) designed to address the creation and reuse of mashups. EMML is, however, not a general-purpose language - EMML was designed to be complimentary to and integrated with languages like JavaScript, Java, Groovy, and Ruby via scripting. EMML is a declarative XML-based language and, as such, leverages and complements existing XML capabilities inherent in XQuery, XPath, and XSLT. EMML is an open language specification. This free-to-use language (and technologies that embed or use it) have a much better chance of meeting the needs of enterprise developers than a proprietary language. **Analysis and Recommendation:** It is particularly suited for interoperability issues related to mashup creation. It is supportive of a strong trend towards lightweight and integrative content and service assembly and is therefore representative of a specific modelling and integration concern.

Unified Service Description Language (USDL) The aim of the Unified Service Description Language (USDL) team, an W3C Incubator Group, is to define a language for describing general, generic parts of technical and business services to allow services to become tradable and consumable [19].

- Technical services are considered software services based on WSDL, REST or other specifications.
- Business services are defined as business activities that are provided by a service provider to a service consumer to create value for the consumer.

The business services are more general and comprise manual and technical services. The USDL definition aims at complementing the technical language stack by adding required business and operational information. The targeted cloud stakeholders for USDL are service providers, infrastructure providers, service assemblers and service consumers. Industry-specific and general-purpose attributes of a service are derived based on use cases, taking into account the target

groups. The USDL group aims to derive best practices and learning from testing cycles that can then be deployed in a number of use cases. These use cases serve as references and proof-of-concept of USDL. **Usage:** The language is usable for any purpose and implementation scenario of business services on a general level. However, it is also extendable for industry-specific aspects. USDL defines an interoperability-centric language that enables its users to model arbitrary services and to integrate with existing standards. **Analysis and Recommendation:** Particularly the aim to address service modelling and support this with mappings to different standards makes this a worthwhile framework for interoperable cloud service modelling. This enables new business models in the field of service brokerage because services can automatically be offered, delivered, executed, and composed from services of different providers. Business-IT alignment is an ongoing concern. Another development to be considered in this context includes SoaML (standardised by OMG, see <http://www.omg.org/spec/SoaML/>), which falls into the same category as USDL in our categorisation as a service description and modelling language. While still under development (and thus far from being standardised), CloudML (<http://www.cloudml.org/>) is a language more specific to clouds, developed by the same group as SoaML.

Open Cloud Computing Interface (OCCI)

OCCI (infrastructure lifecycle management) is now the first of four cloud-specific standards, also including CIMI (like OCCI on infrastructure management), TOSCA (portability and cloud-bursting), and CDMI (data management).

Cloud computing currently is organised into three models or layers offering Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), which all involve the on-demand delivery of computing resources. Providers offer IaaS solutions to enhance elastic capacity, where server instances are executed in their proprietary infrastructure and billed on a utility computing basis. For the infrastructure layer this means that typically virtual machines on a per-instance per-hour basis are the units. For the software (SaaS) layer, software application instances are the corresponding units, managed and billed with similar mechanisms. There are also both commercial and open source products that replicate this functionality in an in-house setting, but also exposing compatible interfaces as a hybrid cloud environments can be realised. The OGF OCCI working group provides an API specification for the management of cloud computing infrastructure [18]. **Usage:** The scope is a comprehensive range of high-level functionality for life-cycle management of virtual machines (or workloads) running on virtualization technologies (such as containers). OCCI provides an API for interfacing IaaS cloud computing facilities, which is sufficiently complete to facilitate the implementation of interoperable implementations:

- Consumers to interact with cloud computing infrastructure (e.g. deploy, start, stop, restart)
- Integrators to offer advanced management services
- Aggregators to offer a single common interface to multiple providers
- Providers to offer a standard interface that is compatible with available tools

- Vendors to offer standard interfaces for dynamically scalable service delivery in their products.

Analysis and Recommendation: OCCI is a step towards matching cloud-specific interoperability needs through standards. While targeting IaaS concerns, it can foster interoperability endeavours at higher levels. The scope of OCCI is high-level functionality for lifecycle management. This is in part realised through coverage of existing proprietary APIs. Storage details beyond creation and mapping of mount points is specifically excluded. Networking details are similarly excluded beyond creation and mapping of interfaces, assignment of these to public or private networks and assignment of dynamic or static IPs. While the focus is on the upper cloud stack layers for the section presented in Figure 1, it is nonetheless a suitable framework for interoperability at the interface of infrastructure services. OCCI allows, as an additional interoperability concern, the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring.

Cloud Infrastructure Management Interface (CIMI)

Similar to OCCI, the CIMI - Cloud Infrastructure Management Interface from DMTF addresses infrastructure management. CIMI which addresses the runtime maintenance and provisioning of cloud services. The scope of the CIMI standard covers core IaaS functionality, addressing deploying and managing virtual machines and other artifacts such as volumes, networks, or monitoring. Once interfaced to the IaaS provider, the information that needs to be processed to manage a cloud service can be discovered iteratively, including the metadata describing capabilities and resource constraints. **Usage:** The model behind CIMI describes resources (systems or collections of resources managed as a whole, e.g. as an OVF file - which is covered below): machines (resource with CPU and memory), volumes (storage), and networks (representing layer 2 broadcasts). It also describes meters, which are metrics for some property, and event logs. Most developers use with the CIMI REST/HTTP-based protocol, the current interface binding to the model (others are expected later). This delivers standard HTTP status codes and supports JSON and XML serialization formats. **Analysis and Recommendation:** CIMI, if widely used, would allow organisations to design cloud-based business solutions being assured that management (and governance) processes will not be compromised if the business solution is moved to another (standards-based) IaaS provider.

Amazon Elastic Compute Cloud (EC2) While OCCI and CIMI are similar standards, in a wider context, Amazon EC2 as a proprietary solution and OpenStack as an open-source solution need to be considered in this context as well. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity servers in Amazon's data centers. These can be used to build and host software systems. EC2 follows a pay-as-you-go for the capacity that is needed. **Usage:** They allow access to components and features using a web-based GUI, command line tools, and APIs. At the core is an Amazon Machine Image (AMI), which is a template that contains a software configuration (operating system, application server, and applications). An AMI is used to instantiate (create) a virtual machine; it is an AMI is a filesystem image which includes an operating system (e.g., Linux, UNIX, or Windows) and any additional software required to deliver a service. From an AMI, instances

are launched, which are running copies of the AMI. You can launch multiple instances of an AMI. Instances run until you stop or terminate them, or until they fail. **Analysis and Recommendation:** EC2 is a de-facto standard and comes with a rich ecosystem, including for instance monitoring tools such as CloudWatch or libraries for a range of programming languages. Some open-source standards are pushed by Amazon AWS competitors to gain market shares.

Topology and Orchestration Specification for Cloud Applications (TOSCA) Supported by OASIS, the TOSCA framework aims to enhance the portability of cloud applications and services. TOSCA enables interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behaviour of these services (such as deploy, patch, shutdown) independent of the supplier creating the service, and any particular cloud provider or hosting technology. TOSCA also aims to support higher-level operational behaviour to be associated with cloud infrastructure management. **Usage:** Through service and application portability in vendor-neutral settings, it enables:

- Portable deployment to any compliant cloud
- migration of existing applications to the cloud

thus adding to consumer choice and dynamic, multi-cloud provider applications. **Analysis and Recommendation:** The core concept behind TOSCA is cloud bursting, which is the ability to move workloads between public and private cloud infrastructures in a transparent way. There seems to be some discussion, with large IaaS providers not having joined the consortium yet. The core to the solution would be a hypervisor-agnostic portability mechanism, which requires IaaS compliance. TOSCA also needs to be observed as a vendor initiative in the context of open-source activities like OpenStack gaining momentum.

Cloud Data Management Interface (CDMI) The CDMI, the Cloud Data Management Interface by SNAI (see <http://www.snia.org/cdmi>), targets cloud storage. The Cloud Data Management Interface is a standard for self-provisioning, administering and accessing cloud storage. CDMI defines RESTful HTTP operations for accessing the capabilities of the Cloud storage system, including allocating and accessing containers and objects, managing users and groups, implementing access control, attaching metadata, making arbitrary queries, using persistent queues, specifying retention intervals and holds for compliance purposes, logging, billing, moving data between Cloud systems, and exporting data via other protocols. Transport security is via SSL/TLS. **Usage:** CDMI defines the functional interface that applications use to create, retrieve, update and delete data elements from the cloud. As part of this interface, a client can discover the capabilities of the cloud storage offering and use this interface to manage containers and the data that is placed in them. In addition, metadata can be set on containers and their contained data elements through this interface. **Analysis and Recommendation:** Compared to OCCI and OVF, CDMI specifically targets data moving and format immigration. Although CDMI can also be used for task management, this would need more extensive rules to be defined.

C. Infrastructure Standards

The OCCI is cloud-specific and addresses interoperability and interface concerns for the infrastructure level. We include here three further cloud standards that are specific to the cloud infrastructure level, of which OVF is the most critical and successful so far. This reflects the current activity and maturity in this context. Again, these are representative of a number of concerns and this selection is not meant to be exhaustive. Memcached and VEPA are representative of other concerns, but for instance as far as protocols are concerns, AMQP (Advanced Message Queuing Protocol) or STOMP (Simple (or Streaming) Text Orientated Messaging Protocol) could have been included.

Open Virtualization Format (OVF) The Open Virtualization Format (OVF), submitted to DMTF as a standard, describes an open, secure, portable, efficient, and flexible format for the packaging and distribution of one or more virtual machines [14], [16]. OVF features include:

- It enables optimized distribution and portability of virtual appliances.
- It aims to support robust installation. Compatibility with the local virtual hardware is also verified.
- It supports both single and multi-virtual machine configurations. With OVF, Software Developers can configure complex multi-tiered services consisting of multiple interdependent virtual appliances.
- It enables portable VM packaging. OVF is virtualization platform independent.
- It supports a wide range of virtual hard disk formats used for virtual machines today, and is extensible to deal with future formats that are developed.

It supports vendor and platform independence as it does not rely on the use of a specific host platform, virtualization platform, or host/guest operating system. It is also designed to be extended as the industry moves forward with virtual appliance technology. **Usage:** VMDK is a popular file format that encodes a single virtual disk from a virtual machine. However, a VMDK does not contain information about the virtual hardware of a machine, like CPU, memory, disk, and network information, making manual configuration costly. OVF provides a complete specification of a virtual machine. This includes the full list of required virtual disks plus the required virtual hardware configuration, including CPU, memory, networking, and storage. An administrator can quickly provision a virtual machine into virtual infrastructures with little or no manual intervention. **Analysis and Recommendation:** OVF is a portable format that allows users to deploy virtual machines in any hypervisor that supports OVF. As such, it sits at the core of resource management in the infrastructure provisioning layer, overcoming previous deficiencies in standardised solutions such as VMDK. Despite, supporting interoperability as a standard for this specific technical context, other features of OVF are important for cloud architecture. For instance, the localisation support is important for cloud services to be offered across different locales. If these locales can be defined and adapted supported by standards, a hurdle for exploitation is overcome.

D. Security Concerns

Authentication and identity management is a primary concern for controlling access to cloud resources. Thus, OAuth shall be covered in a brief overview of security concerns. SCAP is a protocol to deal with downloading security content. OAuth as an identity management and SCAP as a security content related standard are covered in the security context.

OAuth is an open standard for authorization. It allows users to share resources across sites. (e.g. photos, videos, contact lists) stored on one site with another site without having to hand out their credentials, typically supplying username and password tokens instead. OAuth uses username and password tokens. A token grants access to a specific site for specific resources and for a specified duration. OAuth runs on top of HTTP or HTTPS. The OAuth mechanism allows users to grant a third party access to their resources (information) stored with another service provider (which could be a cloud provider), but without sharing access permissions. Twitter is one of the users of OAuth, as is Facebook. OAuth is a service complementary to other identity management mechanisms such as OpenID.

SCAP is the Security Content Automation Protocol. It is a protocol to enable automated vulnerability management, measurement, and policy compliance evaluation. It actually combines a number of open standards that deal with software flaws and configuration issues related to security. NIST is in charge of SCAP. SCAP validation focuses on evaluating versions of vendor products, based on the platforms they support. Validation certificates will be awarded on a platform-by-platform basis for the version of the product that was validated. As it attempts to standardize the automation of the linkage between computer security configurations, it is interesting from a cloud interoperability perspective. Trustworthy cloud systems is the aim within which SCAP can be applied. SCAP provides tools that can, e.g., help determine compliance of security requirements implemented in Cloud provider OS images.

IV. DISCUSSION AND CONCLUSIONS

Interoperability between clouds, cloud services and components is vital for the further development of the cloud ecosystem and market. While standards for the Web Services context are abundant, more specific standards for the cloud computing domain reflect the current maturity. Firstly, a number of standards for the lower infrastructure layers apply to respective cloud computing technologies. They address interoperability solutions for specific aspects like virtual machine management or data management. It reflects initiatives for interoperability for large offerings provided by multinational organisations. Secondly, for platforms and services, the respective (Web) service standards are still of relevance. Standards exist, beyond the core Web services platform, that can further the development of platform and software services from existing offers. Generic service solution can provide a starting point where cloud-specific standards are lacking. This indicates more development in the second category. In addition, it is worth looking at a number of different concerns that help us to judge the state of standardisation and its impact on interoperability: (i) organisations behind standards and their domain, (ii) stakeholders involved through standards and (iii) standards and open-source/proprietary solutions.

Firstly, by looking at the organisations behind the standards, we can also observe that while the Web services domain is primarily dominated by W3C and OASIS in terms of standardisation, the situation in cloud computing is more diverse. Some of the organisations active include DMTF (management of distributed IT systems), the OGF (grid computing), the OMG (middleware), SNIA (storage), OASIS (services), OCC (cloud), as well as national (e.g. NIST) and sector-specific (e.g. ETSI - telecoms) organisations. Currently, there is a dominance of infrastructure and lower-level management, i.e. enabling concerns for cloud computing, reflecting predictions made in reports such as the EU report on cloud computing and its development time lines [3].

Secondly, stakeholders are yet another perspective that we can look at. We have referred to stakeholders in the review and discussion of standards where relevant to differentiate the different interoperability needs of stakeholders in clouds as multi-organisational, multi-role environments. While the infrastructure standards target clearly software developers, the more generic service-oriented standards are more at the interface (as-a-service) level, targeting service providers and consumers. Particularly combined roles, such as prosumers or aggregators that are providers and consumers by combing and brokering between more basic offerings and somehow extended or advanced needs of end-users, benefit from the recent service description and modelling standards.

Thirdly, while standards can achieve interoperability, often de-facto standards emerge from open-source or proprietary solutions. We discussed OCCI and CIMI as standards in a context where OpenStack is a strong open-source framework, all competing with Amazon EC2 as the dominant solution.

Our observations do not reflect to a full extent concerns raised by actual and potential cloud users, such as security, privacy and trust [1], but rather indicate more technology concerns in relation to development and deployment activities. By looking at the standards we reviewed here for indications of future standardisation needs, emerging from the categorisation of standards are the following observations:

- Modelling under incorporation of a variety of standards can support migration and, consequently, the uptake of cloud computing solutions.
- Composition, e.g. mashups, is becoming of importance to provide a market for basic and composite offering where providers and aggregators compete.
- Quality of Service and Service Level Agreement standardisations beyond security concerns in the cloud are actually largely lacking.

Open-SCA and other standards in this context are examples of the emergence of programming and interoperation models for services, which will be instrumental for the composition and customisation of cloud services. Adding more semantics to service descriptions is a direction that can further the composition and brokerage in cloud architectures. Interoperability is, once platform stability has been reached, of increasing concern. Migration and interoperability for service offerings are considered for instance in modelling frameworks such as USDL. The need to support composition, brokerage and mediation is also reflected by EMMML, which addresses mashups.

ACKNOWLEDGMENT

A number of research project, particularly the EU FP7 projects SLA@SOA, 4CaaS and Remics, provided invaluable input for this investigation.

REFERENCES

- [1] 451 Group. *Report on Cloud Computing 'As-a-service' market sizing - Report II*. 2010.
- [2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica. A view of cloud computing. *Communications of the ACM*, 53(4):50–58. 2010.
- [3] EU Commission. *Report on The Future of Cloud Computing - Opportunities for European Cloud Computing Beyond 2010*. EU. 2010.
- [4] K. Boukadi, C. Ghedira, S. Chaari, L. Vincent and E. Bataineh. How to employ context, web service, and community in enterprise collaboration. *Proceedings of the 8th Intl Conference on New Technologies in Distributed Systems*. ACM, 1-12. 2009.
- [5] R. Buyya, J. Broberg, and A. Goscinski. *Cloud Computing - Principles and Paradigms*. Wiley. 2011.
- [6] P. Fingar. Cloud computing and the promise of on-demand business innovation. *Intelligent enterprise*. 2009.
- [7] C. Pahl, S. Giesecke and W. Hasselbring. An Ontology-based Approach for Modelling Architectural Styles. *European Conference on Software Architecture ECSA 2007*. Springer. 2007.
- [8] M.X. Wang, K.Y. Bandara and C. Pahl. Integrated constraint violation handling for dynamic service composition. *IEEE Intl Conf on Services Computing*. pp. 168-175. 2009.
- [9] DMTF Distributed Management Task Force: Interoperable Clouds. http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0101_1.0.0.pdf. Accessed April 2013.
- [10] GICTF Global Inter-Cloud Technology Forum: Use cases and functional requirements for inter-cloud computing. http://www.gictf.jp/doc/GICTF_Whitepaper_20100809.pdf. Accessed April 2013.
- [11] OMG Object Management Group: Cloud Interoperability Roadmaps Session. http://www.omg.org/news/meetings/tc/ca/special-events/Cloud_Interop_Roadmaps.htm. Accessed April 2013.
- [12] CloudCom 2011 Workshop: Market Implementation of Cloud Interoperability and Portability Research in IaaS and PaaS. <http://www.cloud4soa.eu/workshop2011>. Accessed April 2013.
- [13] Cloud Standards Overview. http://cloud-standards.org/wiki/index.php?title=Main_Page. Accessed April 2013.
- [14] DMTF Distributed Management Task Force. Open Virtualization Format Specification Version 1.0.0. http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf. Accessed April 2013.
- [15] Memcached Project web site. <http://memcached.org/>. Accessed April 2013.
- [16] OVF Open Virtualization Format. http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.0.pdf. Accessed April 2013.
- [17] VEPA Virtual Ethernet Port Aggregator. <http://www.ieee802.org/1/files/public/docs2008/new-congdon-vepa-1108-v01.pdf>. Accessed April 2013.
- [18] OCCI Open Cloud Computing Interface. <http://occi-wg.org/>. Accessed April 2013.
- [19] USDL Unified Service Description Language. <http://www.w3.org/2005/Incubator/usdl/>. Accessed April 2013.
- [20] EMMML Enterprise Mashup Markup Language. <http://www.openmashup.org/omadocs/v1.0/index.html>. Accessed April 2013.
- [21] Open CSA - Open Composite Services Architecture. <http://www.oasis-openpsca.org/>. Accessed April 2013.

K Means of Cloud Computing: MapReduce, DVM, and Windows Azure

Lin Gu Zhonghua Sheng Zhiqiang Ma
 Xiang Gao Charles Zhang
 Department of Computer Science and Engineering
 Hong Kong University of Science and Technology
 Kowloon, Hong Kong SAR
 Email: {lingu,szh,zma,xgaoaa,charlesz}@cse.ust.hk

Yaohui Jin
 State Key Lab of Advanced Optical Communication
 Systems and Networks, Shanghai Jiaotong University
 800 Dongchuan Road, Minghang District
 Shanghai, China
 Email: jinyh@sjtu.edu.cn

Abstract—Cloud-based systems and the datacenter computing environment present a series of challenges to system designers for supporting massively concurrent computation on clusters with commodity hardware. The platform software should abstract the unreliable but highly provisioned hardware to provide a high-performance platform for a diversity of concurrent programs processing potentially very large data sets. Toward this goal, a number of solutions are designed or proposed. Among these products and systems, we elect three technologies, MapReduce/Hadoop, DVM, and Windows Azure, as representatives of three different approaches to constructing the infrastructure and instructing the programming in the cloud. We empirically study these technologies using a well-known and widely used application, k-means, and analyze their performance data in relation with the abstraction layers they establish. The implementations of k -means on the three platforms are presented with sufficient details to show the design patterns with these technologies. We analyze the evaluation results in the context of the design goals and constraints of the technologies, and show that the instruction-level abstraction can provide flexible programming capability as well as high performance.

Keywords—Cloud computing; k -means; parallel programming; MapReduce; DISA; big data processing

I. INTRODUCTION

We entered the cloud computing era without a consensus on how large-scale distributed computing systems should be constructed. As many problems remain unsolved for systems with hundreds of loosely-coupled nodes, leading Internet firms have constructed datacenters orders of magnitude larger than typical “large-scale” systems around 2000’s. To system designers, datacenter systems present new technical challenges for the following reasons.

- First, the scale of a datacenter can reach hundreds of thousands of compute servers, which is out of the scope of many distributed algorithms.
- Second, constructing a loosely coupled system at such a scale with commodity hardware inevitably introduces faults in the system to the extent that failures of components are “norm” [1]. This design context departs significantly from traditional high-performance computing systems.
- Third, the applications in datacenters typically require extremely high availability and process very large

data with high throughput [2]. Moreover, a number of computing tasks require deterministic output to ensure correctness, which is well accepted practice in computations of smaller scale but turns out to be very difficult in datacenter systems without noticeably affecting performance.

- Finally, a datacenter is a shared environment where a number of applications run concurrently and may interact with each other. In contrast, a typical high-performance computing (HPC) environment can run in a dedicated or isolated manner. In fact, many HPC users desire to have their application run in relatively isolated resource compartments.

In this context, the cloud computing infrastructure should abstract the unreliable but highly provisioned hardware to provide a high-performance platform for a diversity of concurrent programs processing potentially very large data sets. The programs should be easy to write, worry-free to deploy, and fast to execute.

Several technologies are developing towards this goal—besides earlier solutions developed by Google, Yahoo!, and other industry firms, integrated solutions start to emerge and combine existing software development practices. In addition, a few academic research systems exhibit excellent performance and potentially indicate future directions of innovation in this area.

Among these products and systems, we elect three technologies, MapReduce/Hadoop [3], DVM [4], and Windows Azure [5], as representatives of three different approaches to constructing the infrastructure and instructing the programming in the cloud. We empirically study these technologies using a well-known and widely used application, k-means, and analyze their performance data in relation with the abstraction layers they establish. The implementations of k -means on the three platforms are presented with sufficient details to show the design patterns with these technologies. Our study reveals some characteristics of the design space of cloud computing, and sheds light onto how to construct and program cloud-based systems and applications.

The rest of the paper is organized as follows. Section II introduces the background of the technologies discussed in this paper. Section III presents the k-means programming on

MapReduce/Hadoop, DVM and Windows Azure. Section IV evaluates the performance of k-means computation on the three platforms, and analyzes the experimental results. The related work is discussed in Section V, and we provide concluding remarks in Section VI.

II. BACKGROUND

When Internet datacenters were first multiplexed to conduct serious data-intensive processing, it became obvious that there lacked a method to orchestrate the numerous compute nodes in such systems to conduct effective computation. In spite of immense work on distributed computing and parallel processing, traditional approaches are ill-suited for the new computing platform. Consequently, several technologies have been developed to enable large-scale distributed processing in datacenters, pioneered by Google’s MapReduce [3]. Recently, Microsoft’s Windows Azure integrated a full set of cloud-related technologies, including not only distributed execution but also programmable resource provisioning and data-layer abstractions, in the existing development frameworks [5], [6]. Another important trend is to conduct in-memory computation on commodity-hardware-based clusters. As one of the earliest approaches in this category, the DVM technology constructs an instruction-level abstraction to enable programs distribute computation in a large shared memory space [4].

A. MapReduce-style computation

MapReduce is perhaps the most widely recognized cloud computing technology. It simplifies the data dependence and regulates the semantics of the tasks (e.g., tasks should be idempotent) so that it is easy to implement “embarrassingly parallel” programs and utilize the large number of processor cores in a cluster [3]. Although multiple implementations extend the MapReduce framework to multicore and GPGPU processing [7], [8], MapReduce is mainly design for massively parallel data-intensive processing on a cluster of compute nodes.

While MapReduce is a computational framework, its design is highly dependent on the underlying filesystem abstraction, GFS [1]. First, the replicated data chunks in the filesystem effectively enhance the scheduling efficacy and the I/O bandwidth. Second, the filesystem provides a means of maintaining very large program state and providing a “global” namespace. Finally, atomic operations (e.g., rename) in the filesystem ensures the correctness of the MapReduce computation. The performance of MapReduce computation also relies on a datacenter-wide “meta-scheduler”. The open source variant of MapReduce, Hadoop, has implemented a filesystem, HDFS, with similar semantics to those provide by GFS and a application-level task scheduler.

B. Languages, virtual machines, and DVM

Virtualization is considered part of the technical foundation of cloud computing. In fact, virtualization can take place at several different system layers, and the level of abstraction makes significant difference in generality, expressiveness, and performance. X10 represents an approach of abstracting computation at the language level [9]. Similar approaches include

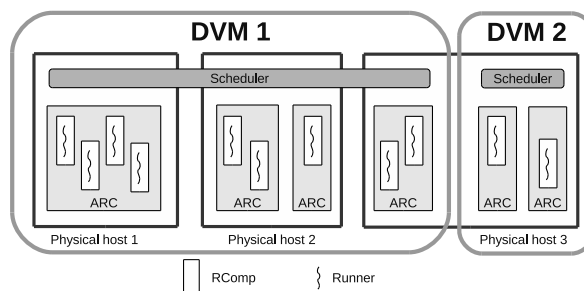


Fig. 1. Organization of two DVM virtual machines on three computers. Each DVM virtual machine utilizes computing resources provided by one single computer or many computers.

Fortress, Google App Engine, and Chapel. The language-level approach gives the programmers more precise control of the semantics of parallelization and synchronization, and X10’s PGAS approach (Partitioned Global Address Space) can potentially support very large data for sophisticated processing. Amazon EC2, on the other hand, abstracts the platform at the instruction level, and builds on existing VMM (Virtual Machine Monitor) technology.

The DVM technology represents a new virtualization function, which provides a low-level abstraction but enables it to support large-scale clusters and sophisticated parallelizable processing [4]. It introduces a new ISA, Datacenter Instruction Set Architecture (DISA), which can be easily emulated on existing hardware. Above such a general architecture, a large virtual machine, DVM, can coordinate resources from a large number of physical hosts and support various programming languages.

Fig. 1 shows the organization of a DVM system composed of two virtual machines spanning three physical hosts. The instruction-level abstraction provided by DISA is very close to typical machine instructions directly supported by processor hardware. However, the semantics of the instructions and the memory model enable multiple tasks, each called a runner, to reside in a large shared memory space, conduct computation and orchestrate massively parallel processing in the abstraction of a “single computer”. The runners resolve their dependence upon each other through a watcher mechanism provided by the DVM. In particular, a latter task depending on an earlier task’s output can be implemented as a “watcher” that monitors the memory area where the former task writes the output data. Once the watched area is modified, the latter task is activated and allowed to proceed with its computation.

C. Windows Azure

The Windows Azure Platform, developed by Microsoft, provides a full set of abstractions and programming tools for developing cloud-based applications. It also uses several related technologies, e.g., VMM, scalable key value store, and datacenter-oriented programming languages, to construct a fairly complete solution.

An application materializes as services hosted in Windows Azure, consisting of one or multiple web roles together with a set of optional worker roles. The program running in the roles may invoke distributed key value store or database

services via well-defined APIs to meet the requirements of a spectrum of applications, including commercial applications requiring strict transactional semantics. Similar to DVM, Azure allows programmers to use programming languages of their choice, given that the language is supported by the program development environment (e.g., Microsoft Visual Studio) and the Windows Azure SDK.

III. K-MEANS PROGRAMMING AND SYSTEM SUPPORT

k -means is a well-known data clustering application used in many areas such as data mining, computing vision and information retrieval. It partitions a data set into k clusters iteratively, and has been implemented in various software systems and applications. Moreover, k -means is widely used for evaluating cloud-based technologies, and, with its clear algorithmic design and adjustable problem size, presents a manageable workload with which various cloud-related technologies can be studied [10], [11].

The k -means process starts with k initial cluster centroids and iteratively refines the clusters by reassigning points to the closest centroids and updating the clusters' centroids. We implement the k -means algorithm with a similar iterative workflow to the one used in Mahout and X10, and optimize the algorithm to achieve better performance in the distributed environment. Similar optimizations are also used in some prior work [10], [12].

A. MapReduce and Hadoop

The iterative computation of k -means does not directly fit into the MapReduce framework, which mandates a reduce stage following a map stage. However, the computation in each iteration is similar with different cluster centroids and the two phases (assigning points to clusters and calculating the new centroids) in each iteration can be expressed as one MapReduce job—we use the map tasks to perform the distance computation and point assignment to clusters as the distance computation between one point and the centroids is irrelevant to the computation for other points in one iteration, and the distance computation can be executed in parallel. The calculation of the new centroids can be performed by the reduce tasks. Hence, we can iteratively run MapReduce jobs and each MapReduce job performs the computation in each iteration of the k -means algorithm. As the distance computation is the most intensive calculation in k -means algorithm, the computation is effectively parallelized using the MapReduce programming model.

Alg. 1 shows the k -means clustering algorithm on Hadoop. The input data are initially stored in files of roughly equal sizes. The input files contain data points' coordinates as a sequence of $\langle \text{key}, \text{value} \rangle$ pairs where the coordinates are stored in the value field. To share the centroids which are read and updated by each MapReduce job, we store the centroids in files in HDFS so that they are read by the map tasks for distance computation and are updated by reduce tasks with the new centroids. Hence, the final output of the k -means cluster program is the centroid files after the last iteration.

The combine function minimizes the communication among map and reduce tasks. Using multiple MapReduce jobs, we are able to implement the iterative computation required by

Algorithm 1 k -means clustering using Hadoop

```

1: create current_centroids and new_centroids in the file system
2: write new_centroids with the first  $k$  points in the input files
3: repeat
4:   delete current_centroids, rename new_centroids to current_centroids, and create empty new_centroids
5:   for all map tasks do
6:     read the data points from the input files
7:     read current_centroids
8:     for all data points do
9:       calculate the distances between the data point and each centroid
10:       $n =$  the identity of the cluster with the closest centroid
11:       $v =$  coordinates of the data point
12:      output the  $\langle n, v \rangle$  (assign data point  $v$  to cluster  $n$ )
13:    end for
14:  end for
15:  Run the combine function to sum the values of data points assigned to the same cluster and output  $\langle n, V \rangle$  for each distinct  $n$  where  $V$  is a composite value of the coordinates of the centroid of the data points being combined and the number of data points associated with  $n$ 
16:  for all reduce tasks do
17:    sum all the intermediate values generated by the combine functions and compute the new cluster centroids.
18:    write the new centroids to file new_centroids
19:  end for
20: until the difference between the centroids in current_centroids and new_centroids is less than a threshold or the number of iterations reaches the maximum value

```

k -means. However, the transition between successive MapReduce jobs cannot be expressed inside the MapReduce framework itself, and external “glue” language must be employed to make such transition happen. It is also noteworthy that the external logic forces the program to use the distributed file system as the media for recording program state. These issues, although tolerable in “embarrassingly parallel” programs, result in non-trivial burden in programming and performance for this slightly sophisticated application.

B. DISA and DVM

DISA presents a generic programming platform, and DVM is constructed above this generic abstraction layer. Hence, it is not difficult to implement the k -means algorithm on a DVM. The program flows are instantiated to runners in DVM and the dependence between the iterations and phases inside each iteration is expressed with watchers. The k -means program on DVM reads its input from disks through one of its I/O channels. Alg. 2 shows the k -means clustering algorithm on DVM.

It may appear to be an unnecessary overhead that the program creates M `dist_cal_runners` in each iteration. In fact, this design results from the snapshotted memory semantics in DISA—the runners see data in its snapshot created upon the runner's instantiation, and the new centroids created at the end of one iteration are visible to runners created at the beginning of the next iteration. It is very efficient to spawn new runners on a DVM, and this makes the overhead of creating runners practically negligible. In comparison, the MapReduce-style programming also requires the program to

Algorithm 2 *k*-means clustering on DVM

```

1: read data points from the I/O channel and store them in designated memory areas
2: new_centroids = the first k data points
3: repeat
4:   current_centroids = new_centroids;
5:   create M (a program parameter) dist_cal_runners, each responsible for one partition of data points, and one dist_cal_runner_watcher.
6:   for all dist_cal_runner and its associated partition P do
7:     for all data point p in P do
8:       calculate the distances between p and each centroid in current_centroids
9:       assign p to the closest centroid n
10:      add p to the sum for centroid n in P – store n, the sum with p included and the number of data points, including p, associated with n in P
11:     end for
12:     dist_cal_runner_watcher is activated each time a dist_cal_runner exits and commits.
13:     dist_cal_runner_watcher checks whether all dist_cal_runner runners have completed
14:     if all dist_cal_runner runners have completed then
15:       the watcher creates the centroid_cal_runner
16:       centroid_cal_runner sums all the intermediate values generated by dist_cal_runners for each centroid, computes the new cluster centroids and assigns them to new_centroids
17:     else
18:       exit the watcher
19:     end if
20:   end for
21: until the difference between the centroids in current_centroids and new_centroids is less than a threshold or the number of iteration reaches the maximum value

```

create numerous map and reduce tasks in each iteration, but the tasking overhead is very heavy in the current implementations.

C. Windows Azure

The web and worker roles in Azure are general enough to implement almost any computational jobs with Windows Azure-enabled languages, with web roles incorporated with built-in web servers. However, it is still a technical challenge to use the distributed data services to construct a reliable mechanism for recording program state and enabling web and worker roles to exchange intermediate data. Alg. 3 shows the design of *k*-means on Windows Azure.

To implement *k*-means on Windows Azure, we use the Windows Azure blob storage to store the input dataset and the output results. The communication between different roles relies on the Windows Azure queue service. We build two types of worker roles – a *master* role and a *slave* role. There is only one master worker role (henceforth called *master*) instance which is responsible of partitioning the dataset, assigning tasks, and collecting results. There are one or multiple slave worker role (henceforth called *slave*) instances. They consume the tasks in the task queue, generate the intermediate results in its data partition, and write back to the result queue.

Algorithm 3 *k*-means clustering on Windows Azure

```

1: new_centroids = the first k data points in the input
2: repeat
3:   current_centroids = new_centroids
4:   master partitions the dataset
5:   master writes centroids together with the task control information (e.g., the number of concurrent tasks) into the task queue
6:   for all slaves do
7:     retrieve the tasks from the queue and compute the intermediate results consisting of the centroid assignment, the sum of the coordinates of the data points assigned to a cluster in its partition and the number of data points in the corresponding cluster and partition
8:   end for
8:   master collects the intermediate results, computes the new centroids, and assigns them to new_centroids
9: until the difference between the centroids in current_centroid and new_centroids is less than a threshold or the number of iteration reaches the maximum value

```

IV. PERFORMANCE, PROGRAMMABILITY, AND EMPIRICAL EXPERIENCE

With *k*-means implemented on Hadoop, DVM, and Azure, we conduct an empirical study on these implementations to study the performance of these solutions, and link the observed performance data to the design choices in cloud computing technologies. To ensure the applicability of our observations, we run the experiments on both research testbeds and industrial platforms such as industrial computing clusters and the Windows Azure platform.

Fig. 2 presents the execution time for *k*-means on DVM and Hadoop on 16 working nodes. From the results, we can see that DVM is at least 13 times faster than Hadoop. We believe this indicates that instruction-level abstractions can lead to more efficient computation and less tasking overhead. While an optimized language-layer construct, such as a MapReduce implementation using memory as the main data storage, can significantly increase the performance, such optimization is unlikely to close the gap between the language and instruction-layer abstractions.

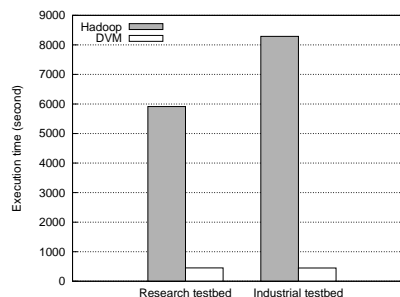


Fig. 2. Execution time for *k*-means on 16 nodes

Illustrating the speedup, Fig. 3 shows the relative performance for *k*-means on DVM and Hadoop on the research testbed (“R” in the figure) and industrial testbed (“I” in the figure) as we scale the number of compute nodes. The relative performance is calculated with respect to execution time on Hadoop with one node. Fig. 4 presents the execution time and

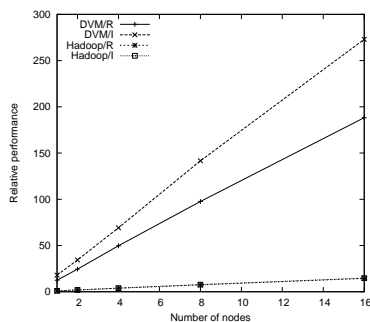


Fig. 3. Relative performance of *k*-means

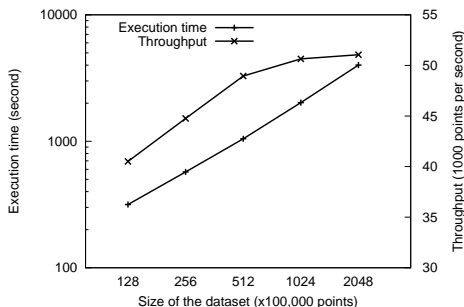


Fig. 4. Execution time and throughput of *k*-means

throughput of *k*-means on DVM with 50 compute nodes as we scale the dataset from 12,800,000 to 204,800,000 points. The throughput is calculated through dividing the number of points by the execution time. The result shows that the throughput increases with the data size, which reflects that DVM scales well with the data size.

Since DVM shows excellent scalability and efficiency, it may appear that the instruction-level abstraction represents the best choice for constructing the cloud technology. However, similar to the situation with traditional ISAs, the instruction layer is mainly defining the interface between hardware and software, and may not provide a complete solution to programming. In fact, our experience of developing programs in the DISA assembly language verifies the challenge of developing programs at a level close to the instruction set, and has prompted us to start developing a compiler for DISA. The goal of the DVM is to provide a powerful foundation, rather than the completion, of the cloud computing technology, and new software tools and supportive routines shall be added to the platform to fully utilize its capability and enhance productivity.

To provide a complete programming environment, Windows Azure integrates the distributed data and processing services with the familiar Visual Studio based development environment. Fig. 5 shows the performance of *k*-means on Windows Azure platform with 1 *master* and 1 to 4 *slaves*. The worker roles reside on small instances in Windows Azure, and both the roles and the Azure storage service are located in the “South Central US” region. We also uses the local emulator to evaluate and compare the execution of *k*-means. The Windows Azure emulator runs on a server with 4 CPU cores and 6GB of memory.

We observe that, when the number of *slaves* increases

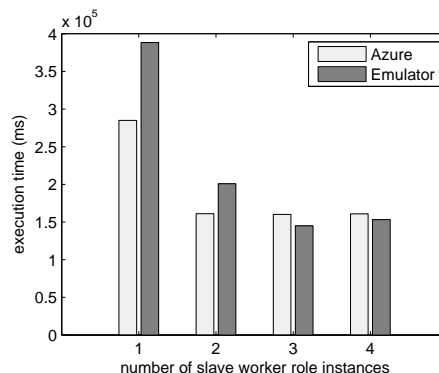


Fig. 5. Execution time of *k*-means computation on Windows Azure

from 3 to 4, the speedup becomes low or even negative. Based on our study, this is likely due to the sharing of CPU resources— 3 *slaves* and 1 *master* can each use 1 CPU core almost exclusively. However, when we have 1 *master* and 4 *slaves*, totally 5 worker role instances share the 4 CPU cores on the physical host, and this serializes a significant part of the computation. Similarly, on the Windows Azure platform, we also observed the same phenomenon. Such hardware-coupling overhead can be mitigated by better scheduling or more resources. Looking at the performance data, we can also conclude that, without the hardware-coupling overhead, *k*-means exhibits obvious speedup on Windows Azure. This verifies that the Windows Azure platform, although designed to provide an easy-to-program methodology in a familiar development environment, can potentially support parallelized scientific computing with the worker roles.

The evaluation clearly shows that the instruction-level abstraction, DISA and DVM, exhibits superior performance in the computation. More importantly, the Turing-complete instruction set of DISA presents a model that can express a wide range of applications. We believe that such generality is a key advantage in the future design of cloud computing systems. Meanwhile, MapReduce has been proved an effective solution to data-intensive computing when the processing logic and data dependence relation fit its specific computation model. Windows Azure, although optimized for Web-based applications, also exhibits a significant amount of flexibility in supporting scientific computation.

V. RELATED WORK

Many programming frameworks and languages are proposed and designed to exploit the computing power of the large number of compute servers inside today’s gigantic datacenters. Dean et al. have created the MapReduce programming model for Google’s datacenter environment [3]. Dryad takes a more general approach, using a “communication DAG (directed acyclic graph)” to depict the dependency among multiple task instances [13]. While these frameworks are successful in large data processing, the restricted programming model makes it difficult to design sophisticated and time-sensitive applications [11], [14], [15], [16].

DVM, on the other hand, allows programmers to easily design general-purpose applications running on a large number

of compute nodes by providing a more flexible programming model [4]. DVM and DISA, the instruction set of DVM, represent a virtualization technology different from widely used virtualization systems, such as Xen and VMware on the x86 ISA [17], [18]. As comparison, VMware pioneered the virtualization of the x86 ISA, and vNUMA extends IA-64 to multiple hosts connected through an Ethernet network that provides “sender-oblivious total-order broadcast” [19]. The new DISA instruction set allows programs to scale up to much larger clusters. Currently, optimization of programs running on a DVM is programmer-driven. Although it has been shown that the performance of DISA programs are very high, optimizing compilers will make it much easier to harness such optimization techniques.

In the meantime, the instruction-level abstraction must combine with high-order languages and compilers to fully release its capability. High-level languages, such as X10, Sawzall and DryadLINQ, which are implemented on top of programming frameworks (MapReduce and Dryad), make the data-processing programs easier to design [9], [20], [21]. To implement their linguistic features efficiently, the language-level approach gives also calls for an underlying computing infrastructure that is capable of supporting general-purpose programs, follows a storage-computing coupled architecture, and provides measures for system-wide optimization. Towards these requirements, DVM provides a foundation upon which language-level instruments can be built.

VI. CONCLUSION

Our study shows that DVM has the best performance for computation in a datacenter, but higher-order languages and compilers must be used to make instruction-level abstraction easy to program. Meanwhile, Windows Azure provides a relatively full set of data services, language support, and development and deployment tools. The role constructs together with the queue-based communication can support a variety of applications, including scientific workloads, on Windows Azure.

ACKNOWLEDGMENT

This work was supported in part by the Huawei Technologies research grant HUAW17-15G00510/11PN, HKUST research grants REC09/10.EG06, DAG11EG04G and SJTU research grant 2011GZKF030902. We thank Microsoft for providing computing resources and support for the research on the Windows Azure platform.

REFERENCES

- [1] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” In Proc. of the 9th ACM Symposium on Operating Systems Principles (SOSP’03), 2003, pp. 29–43.
- [2] L. Barroso, J. Dean, and U. Hoelzle, “Web search for a planet: The Google cluster architecture,” *IEEE Micro*, vol. 23, no. 2, pp. 22–28, 2003.
- [3] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” In Proc. of the 6th Symp. on Operating Systems Design & Implementation (OSDI’04), Berkeley, CA, USA, 2004, pp. 137–149.

- [4] Z. Ma, Z. Sheng, L. Gu, L. Wen, and G. Zhang, “DVM: Towards a datacenter-scale virtual machine,” in Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments, 2012, pp. 39–50.
- [5] “Windows Azure,” <http://www.windowsazure.com/>, [last access: 3/31/2013].
- [6] D. Ongaro, S. M. Rumble, R. Stutsman, J. Ousterhout, and M. Rosenblum, “Fast crash recovery in ramcloud,” in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, ser. SOSP ’11, 2011, pp. 29–41.
- [7] B. He, W. Fang, Q. Luo, N. Govindaraju, and T. Wang, “Mars: a MapReduce framework on graphics processors,” in Proceedings of the 17th international conference on parallel architectures and compilation techniques, 2008, pp. 260–269.
- [8] R. Yoo, A. Romano, and C. Kozyrakis, “Phoenix rebirth: Scalable mapreduce on a large-scale shared-memory system,” in Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on. IEEE, 2009, pp. 198–207.
- [9] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioğlu, C. Von Praun, and V. Sarkar, “X10: an object-oriented approach to non-uniform cluster computing,” in ACM SIGPLAN Notices, vol. 40, no. 10, 2005, pp. 519–538.
- [10] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, “Map-Reduce for machine learning on multicore,” in Proc. of NIPS’07, 2007, pp. 281–288.
- [11] J. Ekanayake, S. Pallickara, and G. Fox, “MapReduce for data intensive scientific analysis,” in Fourth IEEE International Conference on eScience, 2008, pp. 277–284.
- [12] W. Zhao, H. Ma, and Q. He, “Parallel k-means clustering based on mapreduce,” in Proceedings of the First International Conference on Cloud Computing (CloudCom), 2009, pp. 674–679.
- [13] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad: distributed data-parallel programs from sequential building blocks,” in Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, 2007, pp. 59–72.
- [14] Z. Ma and L. Gu, “The limitation of MapReduce: A probing case and a lightweight solution,” in Proc. of the 1st Intl. Conf. on Cloud Computing, GRIDs, and Virtualization, 2010, pp. 68–73.
- [15] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, “Evaluating MapReduce for multi-core and multiprocessor systems,” in Proc. of the 2007 IEEE 13th Intl. Symposium on High Performance Computer Architecture, 2007, pp. 13–24.
- [16] H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, “Map-Reduce-Merge: simplified relational data processing on large clusters,” in SIGMOD ’07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007, pp. 1029–1040.
- [17] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” in Proceedings of the 19th ACM symposium on Operating Systems Principles, 2003, pp. 164–177.
- [18] C. A. Waldspurger, “Memory resource management in VMware ESX server,” *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 181–194, 2002.
- [19] M. Chapman and G. Heiser, “vnuma: A virtual shared-memory multiprocessor,” in Proceedings of the 2009 USENIX Annual Technical Conference, June 2009, pp. 15–28.
- [20] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, “Interpreting the data: Parallel analysis with Sawzall,” *Sci. Program.*, vol. 13, no. 4, pp. 277–298, 2005.
- [21] Y. Yu, M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. K. Gunda, and J. Currey, “DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language,” in the 8th Conference on Symposium on Operating Systems Design & Implementation, 2008, pp. 1–14.

A Novel Cloud Hybrid Access Mechanism for Highly Sensitive Data Exchange

Elhadj Benkhelifa

Faculty of Computing, Eng and Sciences
 University of Staffordshire
 Staffordshire, UK
 e-mail: e.benkhelifa@staffs.ac.uk

Dayan Abishek Fernando

Faculty of Computing, Eng and Sciences
 University of Staffordshire
 Staffordshire, UK
 e-mail: d.fernando@staffs.ac.uk

Abstract— This paper presents a research contribution from a significant Business-University collaborative Project. The aim of the project is to develop a new disruptive approach for Digital Forensics service provision to enable the creation of new value chains via the Cloud technology. The project is highly complex and multidimensional. The project is concerned with the manipulation and service provision for highly sensitive data via a secure Cloud Service Delivery Platform. This paper reports on one aspect of a long running research program, concerned with Security. The paper presents a relatively novel solution adopted in the project for enhanced security to be implemented as part of the intended Cloud Service Delivery Platform. This solution is a hybrid approach between a Single-Sign-On and Multi-Factor Authentication in Federated Settings. Consideration of implementing this solution in the presence of Multi-Tenancy is also discussed in this paper, An aspect which has not been attempted yet, to the best of the authors’ knowledge.

Keywords- Cloud Computing, Single-Sign-on; Multi-Factor Authentication; Cloud Federation; Cloud Security, Multi-Tenancy

I. INTRODUCTION

Cloud computing is fast becoming a mainstream technology replacing the current practices in IT resource provisioning. The Cloud technology is a disruptive model as it represents a major change to the IT services landscape. Cloud Computing describes a new way of delivering IT services based on Internet protocols, and it typically involves provisioning of dynamically scalable and often virtualized resources [1]. Cloud services offer great benefit to organizations by eliminating complexity of service designing, deploying and configuring. Cloud Computing enables the delivery of services through the on-demand service-provisioning model to end users on a pay as you go basis over a network such as the Internet [1, 2].

Using the Cloud, companies can drive a more efficient, effective, and consumer led commercial that helps them continually reinvent and transform the way they do business, focusing on what makes sense from a business delivery, consumer satisfaction and growth model [1]. Enabling the underlying IT allows businesses to rapidly deliver services, integrate across technological divides, and increase

efficiencies; where cost reduction and increased efficiency is a major feature; along with the ability to affect reach, reliability and availability no matter where you are or what time it is. In short, Cloud technology offers a wide spectrum of new digital value chains. However, security is often cited as one of the major concerns in adopting the technology.

Cloud services are mainly delivered through three main delivery models Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [1, 2, 3]. Table 1 summarises the main security concerns for each of the delivery models.

TABLE I. MAIN SECURITY CONCERNS ON CLOUD DELIVERY MODELS

Delivery Models	Security Concerns	Examples
SaaS	Data Security App Security Identity Authentication	Google Apps, Oracle SaaS, NetSuite Salesforce
PaaS	Data & Computing Availability Data Security Disaster Recovery	Google App Engine, RedHat, Microsoft Azure Heroku
IaaS	Data center construction Physical Security Network Security Transmission Security System Security	Amazon EC2, Verizon, IBM, Rackspace, Nimbus

As defined by the American National Institute of Standards and Technology (NIST), Cloud can be deployed in four models: Public and Private Clouds together with less commonly used models, Community and Hybrid Clouds; private Cloud; community Cloud; and hybrid Cloud; [3, 4].

This paper presents a relatively novel solution adopted in a real business case project; funded by a UK research Council to develop a complex Cloud Service Delivery Platform for Digital Forensics [15]. This solution is a hybrid approach between a Single-Sign-On and Multi-Factor Authentication in Federated Settings.

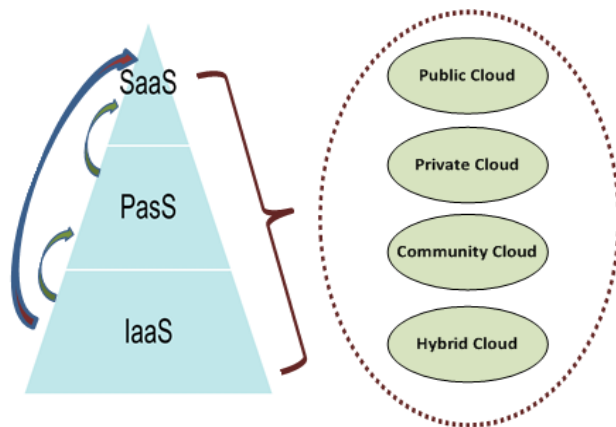


Figure 1. Delivery models vs. Deployment models.

Consideration of implementing this solution in the presence of Multi-Tenancy is also discussed in this paper; an aspect which has not been attempted yet, to the best of the Authors' Knowledge (Section IV). The paper also was an opportunity to review the area of Cloud service provision and reflect on the current practices for Cloud access management (Section III) and classifies Security challenges (Section II). Conclusions and direction for future research are summarised in Section V.

II. CLOUD SECURITY AND PRIVACY CHALLENGES

This section provides a concise summary of the current security and privacy challenges in a Cloud environment based on state of the art classification under five main categories, as illustrated in Figure 2.

There are number of security and privacy concerns for today's cloud computing landscape as it incorporates with various technologies including virtualization (i.e. virtual servers, virtual networks), on demand service provisioning, shared resource pools (i.e. data, memory), concurrent access, load balancing and distributed data are some examples [5]. Also big data in cloud has always been a security and privacy challenge due to the velocity, volume and variety of data. As shown in Figure 2, cloud security and privacy challenges can be classified into five main categories.

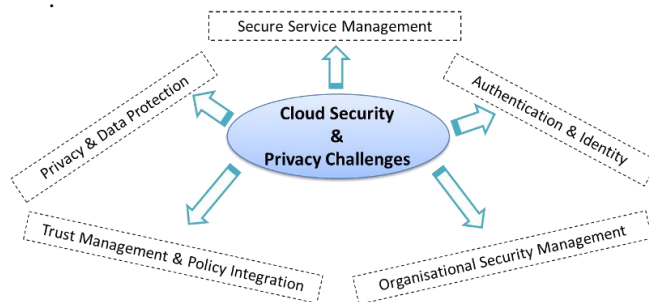


Figure 2. Cloud Security and Privacy Challenges.

Authentication and Identity Management

- Interoperability challenges in between service providers
- Inherent limitations in passwords
- Lack of clarification of multi-tenancy
- Multi-jurisdiction issues

Trust Management and Policy Integration

- Semantic heterogeneity
- Jurisdiction issues
- Trust and interaction/sharing requirements
- Composition of multiple services to enable bigger application services

Secure Service Management

- Issues such as price, QOS, and SLAs
- Automatic and systematic service provisioning; and a composition framework that considers security and privacy issues

Privacy and Data Protection

- Storing data and applications on systems that reside outside of on-premise datacentres
- Shared infrastructure, risk of potential unauthorized access and exposure.
- Privacy-protection mechanisms must be embedded in all security solutions.
- Balancing between data provenance and privacy

Organizational Security Management

- Shared governance
- Dependence on external entities
- Insider threat is significantly extended when outsourcing data and processes to Clouds.

III. CLOUD ACCESS MANAGEMENT

The Cloud Data Management Interface (CDMI) defines the functional interface: to implement strong access controls; provide data encryption; and storage media for secure multi-tenant Cloud environments; [6] CDMI supports most of standard protocols like File Transfer Protocol (FTP), Storage Area Network (SAN), Network Attached Storage (NAS) and Web Distributed Authoring and Versioning (WebDAV) [7].

Preventing un-lawful access to data resources in the Cloud is a key challenging deliberation. The most significant issue is that the digital identification and framework may not naturally extend into a Cloud environment, thus re-engineering the existing framework to support Cloud services may prove to be difficult [8]. Employing two different authentication protocols, one for the internal systems and another for external Cloud-based systems, leads to technical difficulty that can become unusable over time. Identity federation, supported by the introduction of Service Oriented Architectures (SOA), is one solution. Identity federation allows both Cloud service provider and service organisation to trust and exchange digital identities and

attributes across both domains. As shown in Figure 3, for federation to succeed, identity and access management transactions must be interpreted carefully and unambiguously, and protected against attacks [8, 9]. Federation is enabled by an Authorisation Exchange Standard [10].

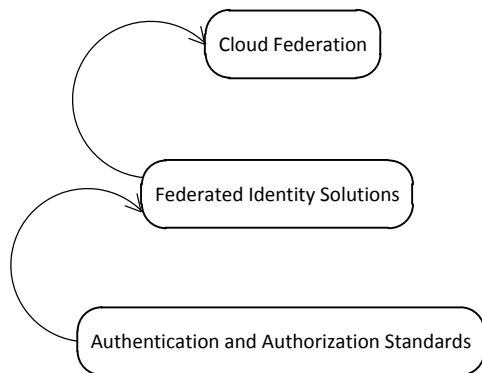


Figure 3. Single Sign-on based Cloud federation framework.

A. Authentication /Authorisation Standards

The Authentication and Authorisation Standard for Cloud computing defines a set of principles for exchanging authentication and authorisation between security domains. There are a number of protocols like OpenID, UMA, Radius and SAML which provide support to build the authentication and Authorisation frameworks [10]. The Security Assertion Markup Language (SAML) is the most widespread standard that integrates digital security tokens containing assertions which pass information about a user, protocols and profiles so as to implement authentication and authorisation scenarios which allow secure data exchange between domains [11].

B. Single Sign-On (SSO)

As shown in Figure 4, Single Sign-On (SSO) is a process that enables a user to have single user credentials to gain access to multiple applications and resources which have been assigned for the user. SSO allows users to switch between different applications more effectively without any additional authentication requests [11].

Numerous researches have shown the prompt impact of SSO within Cloud industry. Shibboleth IDP, oxAuth OP, UMA PDP and LDAP cache are some of the architectures that refer to frameworks to build SSO environments [10, 11].

According to JANET, Shibboleth is the most widely adopted open source federated identity solution developed by the Internet2 middleware group [12]. The latest Shibboleth (V 2.0) builds on top of the SAML 2.0 authentication and authorization standards.

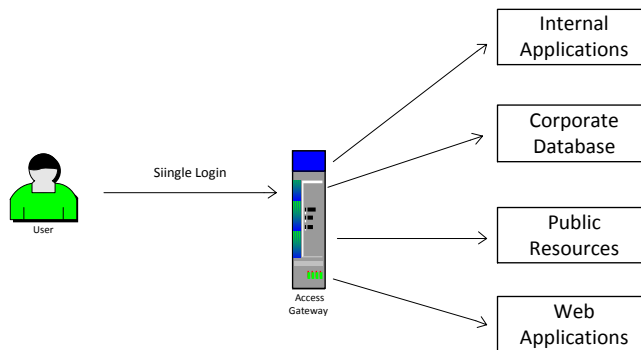


Figure 4. Single Login – Multiple Applications.

C. Multi-Factor Authentication

The multi-factor authentication is an authentication method, which requires two or more authentication factors to allow access to the IT resources [13]. As shown in Figure 5, there are three factors involved in establishing the multi-factor authentication framework.

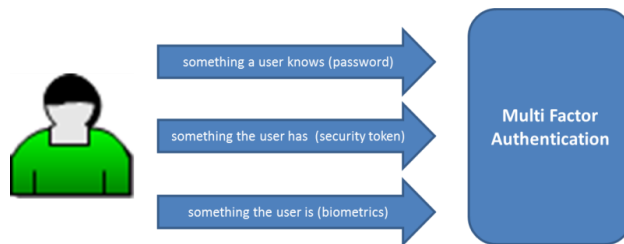


Figure 5. Multi-Factor Authentication.

Research studies have shown that a traditional username and password based Single-factor authentication is no longer strong/scalable enough to support the present security demands of the cloud. This is because compromise of a single factor results in a breach; whereas multi-factor authentication decreases the chance of subversion by having an increase in likelihood of correct identification with every additional factor. The current and widely used trend is two-factor authentication; it is widely spread amongst large financial institutions in Europe [8]. Two-factor authentication can also be found in a number of user-facing applications, such as social networks (Facebook) and Google Applications. Although the two-factor is currently considered the most efficient and very secure scheme, research continues to explore other more effective solutions as Two-factor authentication is already seen being breached. Therefore, an increased factor authentication (eg. three-factor) or hybrid approaches [9, 10], are currently being tested and researched.

This paper is an attempt to implement a hybrid authentication approach, using Single-Sign-On together with two-factor authentication whilst considering the multi-Tenancy Scenario (Section IV). This approach is proposed for direct application in a real business case.

IV. SECURE CREDENTIAL FEDERATION FOR HIGHLY SENSITIVE DATA EXCHANGE

Finding a balance between security and simplicity/accessibility is very important [13]. The proposed multi-level, hybrid authentication mechanism based on Single Sign-on (SSO) and two-factor authentication, enables not only Cloud federated access among multiple applications and organizations but also allowing sensitive data exchange between different domains (Figure 7).

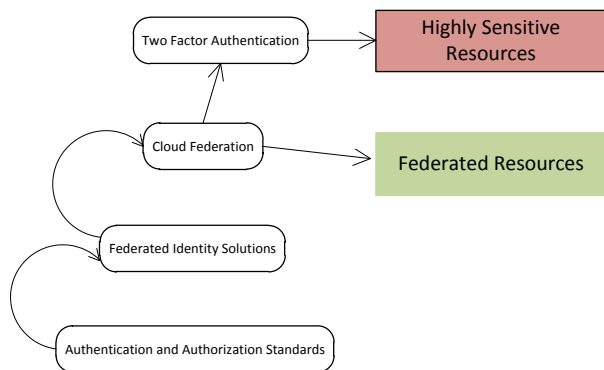


Figure 6. Proposed - Multi-Level Hybrid Authentication Mechanism

Single-Sign-On provides a unified mechanism to manage and monitor user interactions and business rules, determining user access to Cloud applications and data resources through the internet. Some industries require extra levels of security and identity protection over SSO settings in order to precede some specific secured tasks such as extremely sensitive application/data access, cross-border investigations, and remote data manipulation activities. Therefore, this paper is inspired to introduce a hybrid Cloud access framework by combining multifactor authentication with SSO in order to protect enterprise identities and thus enable a strong authentication method. Figure 6 and 7 illustrate the basic idea of the proposed solution.

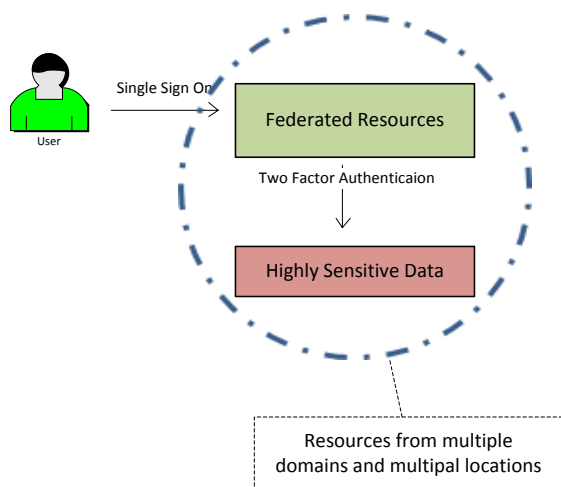


Figure 7. Resource Access through Multi-Level Hybrid Authentication.

The proposed hybrid solution comprises of two levels of security access layers providing access to federated resources and further sensitive resources within the federation agenda. The proposed solution defines federated resources as combined Cloud environments and applications for the purpose of resource sharing using single sign on to access multiple applications from multiple locations. Also, this solution allows access to highly sensitive resources within federation settings. Table 2 displays the access matrix vs. authentication of the proposed framework.

TABLE II. DIFFERENT RESOURCE ACCESS THROUGH MULTI-LEVEL HYBRID AUTHENTICATION

Data Access \ Authentication	Federated Resources	Highly Sensitive Resources
Single Sign-on	√	√
Two Factor Authentication	X	√

Figure 8 shows the high level access flow of the proposed hybrid access mechanism. The cloud access gateway acts as the doorman at the enterprise perimeter to cloud services and service users. The users can gain access to federated resources simply by providing SSO credentials. If the user need to access sensitive data within the federated settings, then they will be diverted to the two factor authentication for further credentials in order to gain access. Figure 9 illustrates the interaction between cloud components and users, showing how processes operate with one another and the direction in which federated resources are accessed. In a similar manner, Figure 10 displays the enhanced version of accessing sensitive data through two factor authentication.

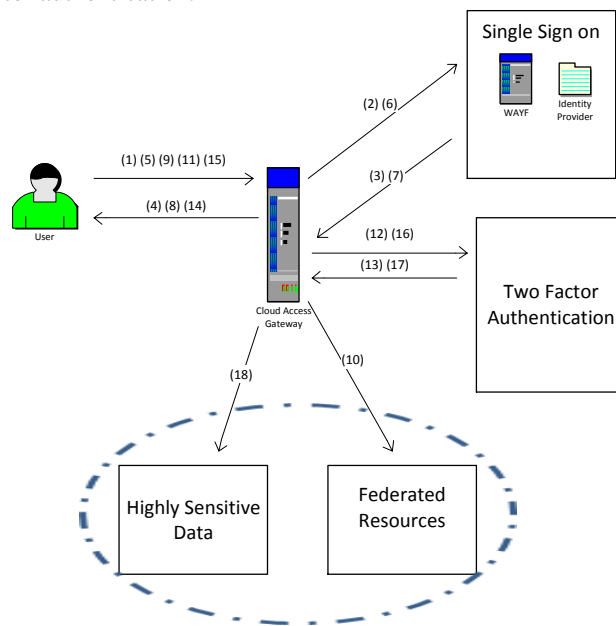


Figure 8. Hybrid SSO-Two-Factor Authentication Framework.

A. Cloud Access Process flow

1. Request to access federated resources
2. Redirected to the SSO WAFS (Where are you from)
3. IDP Request
4. Request User Credentials
5. Provide Credentials
6. Authentication
7. If success – Generate SSO User Session and pass into the Access Gateway
8. Prompt Authentication Status and redirect to federated resource pool
9. Access to Federated Resources
10. Federated Resources
11. User Request to access Highly Sensitive Data
12. Redirected to the Two Factor Authentication Service
13. Generate Security Token and pass into the Access Gateway
14. Send Security Token via SMS / Email
15. Enter Security Token
16. Pass token for verification
17. Update SSO session
18. Access to Highly Sensitive Data

B. Considering Multi-Tenancy with the Proposed Solution

Multi-tenancy is a method of sharing a single instance of data and applications among multiple customers (tenants) by allocating a unique profile for each tenant. Multi-tenancy presents a number of benefits such as: reduced operation cost by sharing resources (software/hardware); increased utilisation /optimisation rate in data centres and instant service provisioning for new clients [14]. However, despite the above-mentioned benefits, multi-tenancy is not widely deployed in the Cloud industry. The balance between resource sharing and security is very constrained and conflicting within a multi-tenancy framework. Also, the present multi-tenancy delivery models (Dedicated resource model, Metadata map model) are either less flexible or less secure (Table 3). The future developments of the proposed hybrid authentication solution will attempt to embed the multi-tenancy architecture where it is believed a mix of dedicated resources and metadata map architectures will deliver stronger security and greater flexibility. To the best of the authors’ knowledge, this work is unprecedented, due to its complexity and limitations in the current Cloud Technology.

TABLE III. MULTI-TENANT DELIVERY MODELS

Dedicated resource model	Metadata map model
Increased Security	Increased Flexibility
Lower Flexibility	Lower Security

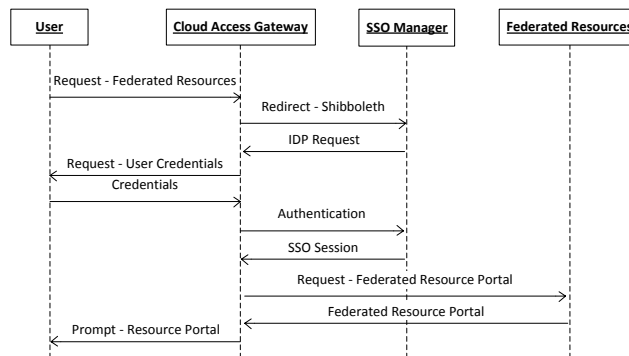


Figure 9. Sequence Diagram – Access to Federated Resources.

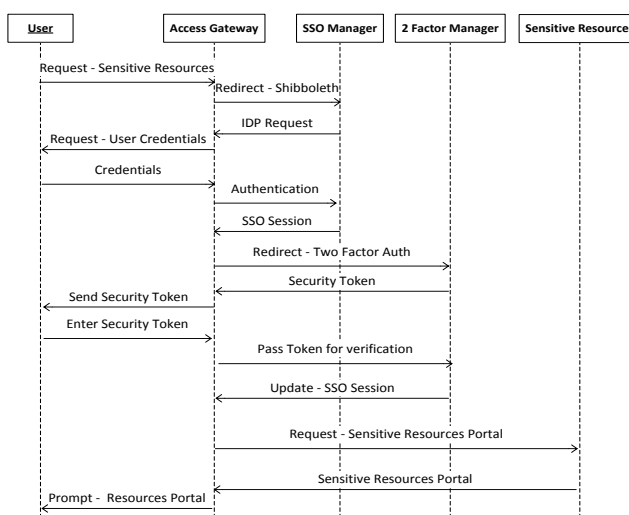


Figure 10. Sequence Diagram – Access to Sensitive Resources.

V. CONCLUSION

This paper proposes a novel hybrid solution for increased security to be implemented as part of a real business case project. The project is concerned with highly sensitive data, hence a more complex security approach is needed. The proposed hybrid solution, Single-Sign-On and two-factor authentication, is accepted by the project consortium and end-users to be a state-of-the-art and highly secure authentication approach. The proposed framework is currently being tested as part of the project deliverables, and results will be shared in future publications. Immediate future work will investigate the implementation of the proposed framework in the presence of Multi-tenancy, in federated Cloud settings. Another direction for future research is to evaluate the feasibility of implementing more than two factors for authentication. This evaluation will include the readiness of the current Cloud technologies for such enhancement in security and working out the balance under different constraints.

REFERENCES

- [1] C. M. DaSilva, P. Trkman, K. C. Desouza, J. Lindic, Disruptive Technologies: A Business Model Perspective on Cloud Computing, 2013.
- [2] C. Chapman, et al., Software architecture definition for ondemand cloud provisioning, Cluster Computing, 2011: pp. 1-21.
- [3] P. Mell and T. Grance, The NIST Definition of Cloud Computing, 2011
- [4] M. Armbrust, et al., A View of Cloud Computing, Communication of ACM, 2010
- [5] A. Lenk., et al., What is Inside the Cloud? An Architectural Map of the Cloud Landscape, in Workshop on Software Engineering Challenges of Cloud Computing, Collocated with ICSE 2009, IEEE Computer Society: Vancouver, Canada , 2009.
- [6] SNIA Cloud Data Management Interface (online). Available at URL: <http://cdmi.sniaCloud.com> [Retrieved: Feb 2013]
- [7] JSON-RPC project (online). Available at URL: <http://json-rpc.org> [Retrieved: Feb 2013]
- [8] P. A. Boampong, L. A. Wahsheh, Different facets of security, Proceedings of the 15th Communications and Networking Simulation Symposium, 2012.
- [9] L. Peterson et al., Slice-based federation architecture, v2.0. <http://groups.geni.net/geni/attachment/wiki/SliceFedArch/SFA2.0.pdf> [Retrieved March 2013]
- [10] K. D. Lewis, J. E. Lewis, "Web Single Sign-On Authentication using SAML," IJCSI- International Journal of Computer Science Issues, 2009.
- [11] D. Raywood, Google adds two factor authentication to Gmail via SMS one time passwords, 2010.
- [12] L. M. Vaquero, L. Rodero-Merino, J. Caceres, M. A. Lindner, Break in the Clouds: towards a Cloud definition, 2009.
- [13] M. Trojahn and F. Ortmeier, Biometric authentication through a virtual keyboard for smartphones, in International Journal of Computer Science & Information Technology (IJCSIT), 2012.
- [14] S. Walraven, T. Monheim, E. Truyen, W.J. Sdsd, Towards performance isolation in multi-tenant SaaS applications, Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing, 2012.
- [15] E. Benkhelifa and D. Fernando, Developing a Complex Cloud Service Delivery Platform: Practical Lessons From Real Business Case. International Conference of Cloud Computing and Services Science (ICCCSS'13), Dubai. 29-31 Jan 2013.

Eliciting Risk, Quality and Cost Aspects in Multi-cloud Environments

Victor Muntés-Mulero and Peter Matthews

CA Technologies
CA Labs Europe

Email: {Victor.Muntes, Peter.Matthews}@ca.com

Aida Omerovic

SINTEF
Oslo, Norway

Email: aida.omerovic@sintef.no

Alexander Gunka

BOC Information Systems
Austria

Email: alexander.gunka@boc-eu.com

Abstract—With the increasing number of providers offering cloud-based services, new opportunities arise to build applications capable of avoiding vendor lock-in issues. Such applications are developed in multi-cloud environments that allow replacing services with those offered by alternative providers. While this may improve quality and provide independence from a single cloud service provider, it also brings new risks. Being able to assess risks and those quality aspects that are specifically related to multi-cloud environments is essential in order to design reliable applications based on the use of cloud services. Although a lot of work has been done to study risks and quality aspects for cloud services, this is usually focused in single-provider scenarios. In this position paper, we discuss several risks and quality aspects that are specifically related to multi-cloud environments.

Keywords- *Multicloud, Risk assessment, Quality prediction, Cost prediction*

I. INTRODUCTION

Many applications and Cloud Service Providers (CSPs) replicate or combine services from multiple clouds or multi-clouds (also called cloud mashups [9]) to avoid the risk of vendor lock-in. New architectures, technologies, and standards are being proposed to support collaboration among multiple cloud systems [1], [2], [6], [7]. Although direct collaboration among applications hosted by different clouds is still restricted [9], the adoption of these proposals will improve the ease of migration from one provider to another and increase open competition. Nevertheless, the current environment already offers many opportunities for collaboration among services offered by different providers without requiring standards or important changes to the delivery model.

In multi-cloud environments, it is essential to provide tools that guide multi-cloud application architects to choose the services providing the necessary quality and ensuring acceptable level of risk. Previous work has focused on describing quality aspects and metrics to measure the suitability of a cloud service from a multi-dimensional perspective. An example of this is the Service Measurement Index (SMI) [10], a framework designed to allow for quick and reliable comparison of IT business services. SMI establishes the basis for comparing isolated services in regard of several categories such as for instance accountability, agility or assurance. However, they do not explicitly analyze these aspects in a multi-cloud context.

Based on this quality aspects and other factors, model-based decision making system help application designers to choose the cloud components that better fit their needs. Some

of these major factors include functional and non-functional properties, as well as cost and the added value. A trade-off between such factors is the basis for decision making. This trade-off is particularly complex between the non-functional factors, the variable parts of the architecture, and the cost of the selected solutions. The variability, as well as incomplete information or knowledge, are also sources of risk. Since functional requirements are less flexible and specified rather early, and since the added value is strongly related to functional properties, the factors that are tuneable and highly interrelated are risk, quality and cost.

In this paper, we discuss the risks related to cloud services in a multi-cloud environment, the quality aspects that are specific to that environment and make some cost considerations. We analyze three important issues which are essential in multi-cloud environments: interoperability issues between services offered by different providers, the ease of migration from a current service to a new equivalent service, and the security issues that arise from the fact that confidentiality, integrity, availability, etc. does not depend on a single provider.

This paper is organized as it follows. Section II presents related work. Section III briefly describes multi-clouds scenarios and describes the aspects considered in this paper. Section IV presents a summary of quality aspects to be considered. Section V provides a brief description of costs that must be taken into account in this type of environment. In Section VI, we discuss risks that must be considered in a multi-cloud. Finally, Section VII presents the conclusions and draws some future work.

II. RELATED WORK

As a basis for the elicitation of the adequate quality characteristics, the software product quality standard ISO/IEC 9126 defines quality as the totality of features and characteristics of a software product that bear on its ability to satisfy stated and implied needs. The ISO 9126 standard provides an established specification of decomposed quality notions with their qualitative and quantitative definitions. The standard defines a quality model for external and internal quality, and for quality in use. The characteristics of the internal and external quality model are functionality, reliability, usability, efficiency, maintainability and portability. These are in turn decomposed into a total of 34 sub-characteristics.

SMI [10] is a standardization effort from the Cloud Services Measurement Index Consortium (CSMIC) consisting of

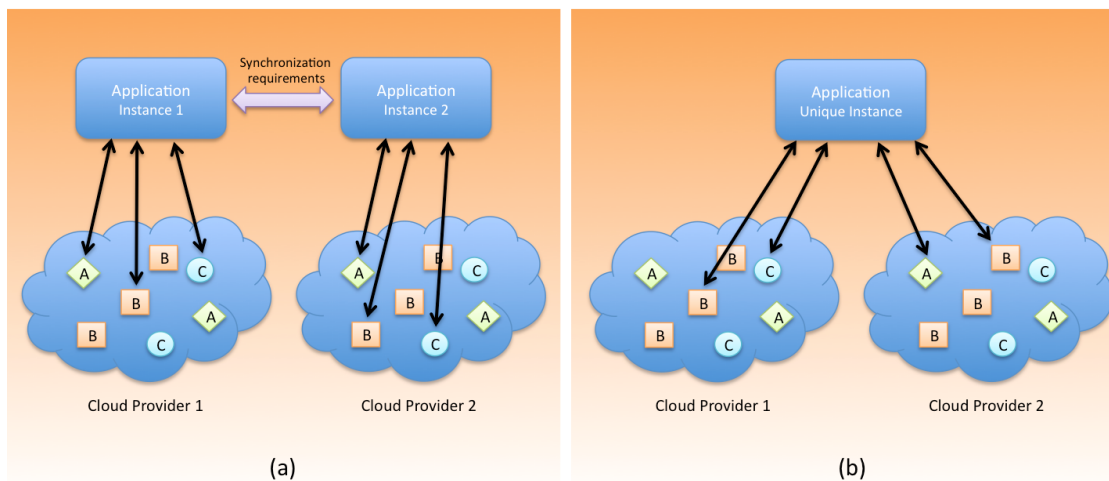


Fig. 1: Examples of two different multi-cloud scenarios

academic and industry organizations. The Service Measurement Index (SMI) uses a series of characteristics and measures to create a common means to compare different services from different suppliers. The characteristics are categorized as Usability, Performance, Agility, Security and Privacy, Financial, Assurance and Usability. Each of these characteristics has a number of measures that can be used to evaluate the risk in using a service. For example in the accountability category one of the measured attributes is Compliance and another is Service-Level Agreements (SLA) verification both of which can be used to create a risk measure for the service and the provider. The work presented in this paper is based both on the ISO standard and SMI conclusions.

In order to enable risk monitoring based on indicators or metrics, there is a need not only to identify the relevant indicators, but also to understand how to relate the indicators to potential risks, and how to aggregate the monitored values into risk levels [5]. In this paper, we identify both risks and quality aspects related to multi-cloud environments. To our knowledge, none of the previous work has been focused on jointly analysing risk, quality and costs in a multcloud environment.

III. MULTI-CLOUD SPECIFIC NEEDS AND CHALLENGES

We define a *multi-cloud application* as any piece of software using several cloud services hosted by two or more different providers. Usually, two different scenarios are considered when referring to multi-cloud environments. Figure 1 depicts these two cases. In the first case (a), an application is replicated to improve resilience, and may also be used to avoid vendor lock-in. This means that the application has two independent instances using the same type of cloud services (A, B, C in the figure) in two different cloud providers. In the second case (b), a single instance of the application runs different cloud services hosted by two or more cloud providers. In this latter case, it is also possible to replicate services to ensure availability. This would also imply synchronization. Because of the need for high interoperability between services offered by different providers, scenario (b) is in general more complex to manage and may potentially involve larger risk compared to (a). In fact,

scenario (a) may be considered a particular case of scenario (b). Because of this, we focus on scenario (b) in this paper.

The use of multiple cloud services from multiple providers adds a new dimension of complexity to an already complex cloud computing scenario. Heterogeneity caused by the existence of independent providers that have created their own business models, protocols, processes and formats generates an increasing number of risks to be taken into account when creating a new application using a multi-cloud strategy. In this paper, we emphasize three essential aspects that must be considered in a multi-cloud environment:

- **Heterogeneity of services offered by different providers results in reduced interoperability:** the lack of standard interfaces for services in different clouds and the creation of independent proprietary systems by each provider, make multi-cloud environments very heterogeneous. Interoperability problems may range from technical issues, such as messaging interfaces or quality of service, to semantic, organizational or legal issues. This heterogeneity is an important risk to consider at design time, since it will influence the capacity of an application architect to decide between one service and another. In terms of quality, a service will be highly interoperable with other systems if it can be combined in collaboration with many other services, from the same or other cloud service providers.
- **Migration between services offered by different CSPs is an essential operation to ensure the compliance with the application requirements:** one of the most common reasons to deploy an application in a multi-cloud environment may include increasing the cloud service catalog and increasing the capacity of users to migrate from one service to another in case the requirements on the application are not fulfilled. We call this capacity *replaceability*, and it represents the ease to migrate from one service to another to replace the first one. It will be essential to decompose migration processes from one cloud service to another into several finer-grained steps, and analyze the quality

aspects to be considered in the process.

- Security threats are increased in multi-cloud computing environments:** increasing the number of services and providers, will increase the complexity of the overall system and the number of potential attacks. Control over customers data decreases, especially because of potential migration between services of different providers. The continuous communication of data between services in different clouds may also result in storing data in intermediary less secure external storage systems, increasing the overall vulnerability and potentially compromising confidential information. In terms of data privacy, multitenancy makes it more difficult to guarantee confidentiality of sensitive information.

These three aspects have been selected and prioritized after several interviews with industrial and academic partners. They have been chosen based on experience and from studying different migration processes. They represent three essential requirements in a multi-cloud environment: coordination between services offered by different providers, capacity to replace a service by another one, and the increase of complexity in the system increasing possible points of failure in terms of security. Note that, we do not claim this to be a comprehensive list of possible aspects to analyze, but we believe they are a good starting point to establish the basis to define risk and quality in multi-clouds.

IV. QUALITY ASPECTS IN MULTI-CLOUD ENVIRONMENTS

In this section, we analyze those quality aspects related to the issues detected in Section III that must be considered in a multi-cloud environment: interoperability, replaceability and security. Figure 2 summarizes the quality aspects considered related to these three issues.

A. Interoperability

The interoperability problems of cloud services in the controlled environment of a single CSP, are exacerbated by mixing services from different providers and may imply incompatibilities in other areas of a mixed service implementation. From the point of view of a developer, it will be important to know the degree of interoperability of a certain service with respect to other services it must interact with. Figure 3 depicts the scenario studied in this case. Figure 2 divides these incompatibilities in four different areas: technical, semantic, organizational and legal. The Technical interoperability quality aspects refer to the capacity of two or more services offered by different providers to communicate through common protocols and to jointly guarantee a certain quality of service. For instance, possible indicators that might be used to evaluate the degree of technical interoperability might be the number of standardized interfaces that can be compared towards the total number of interfaces used by the service, or the average recovery time of the service or other performance aspects. Semantic aspects refer to aspects related to the data syntax consistency and the data quality. These data related aspects are relevant for interoperability since only two or more services offering mechanisms to guarantee global data properties might be combined in the same application. Organizational aspects

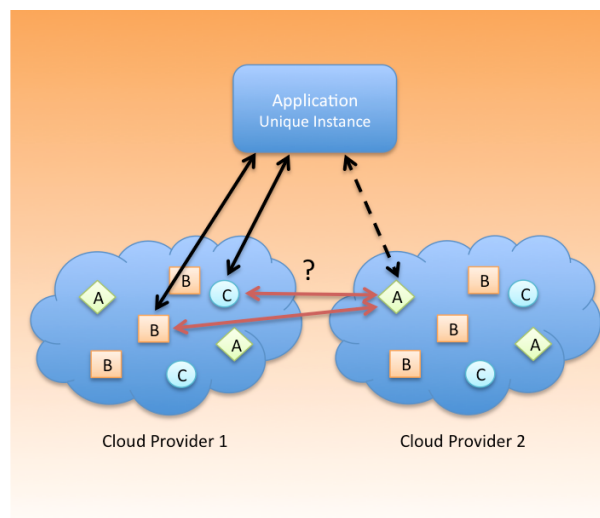


Fig. 3: Interoperability in a multi-cloud environment: services offered by different providers interacting with each other.

indicate how adaptable a service is to several work processes. Since each of these work processes might be established by different providers, it is important that a service in a multi-cloud environment is adaptive to fit the requirements of each work process in each case. Changes in a work process may require changes in a specific cloud service that is already used. In a migration process, choosing a new cloud service candidate to replace an existing service may depend on the capacity of this new service to adapt to the existing work process. Compliance with existing cloud service standards in terms of role and functionality of that specific cloud service will be essential to ensure good organizational interoperability. Regarding legal aspects, we focus on regulatory compliance. Compliance in this case may be understood as a list of laws that are observed by the service provider. Some may be mandated by the customer such as Sarbanes-Oxley [8], some by government, e.g. Data Protection act [3]. It is the presence or absence of compliance that is of interest. A purchasers compliance officer will provide a number of regulations that any service would have to observe and these would be part of the requirements gathering.

Several aspects are likely to be difficult to measure. A good example is the number of standards in the communication capability aspect. Standards for cloud service communications are evolving and several attempts have been made to create an agreed list of them. NIST has a list of recommended standards and the European Commission has created a Cloud Standards Coordination (CSC) that is being administered by ETSI [4]. The requirements of multi-cloud applications may need some or all of the relevant standards to be adhered to.

B. Intercloud Replaceability

Migration is an essential operation linked to multi-cloud environments. The capacity of a software architect to redesign an application and replace existing services by other services with the same or similar functionalities defines in fact the realism of considering cloud mashups. For instance, a cloud database service may integrate application building tools that

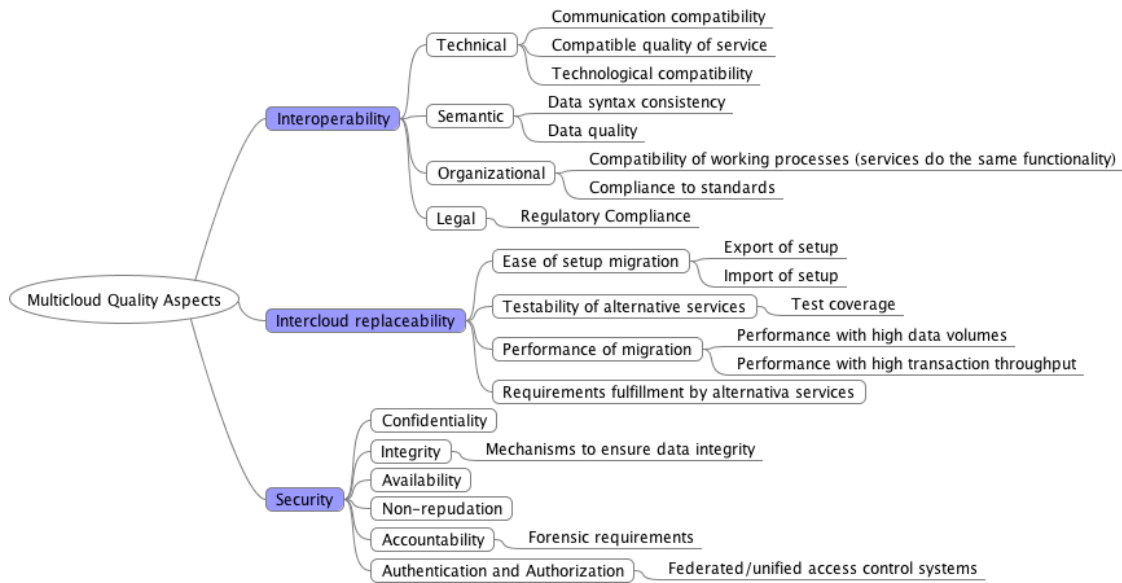


Fig. 2: Quality aspects related to multi-cloud environments

might be used by our system, such as APIs based on web services standards. If the other services interacting with this cloud database service assume that these tools exist, moving to a new cloud database that it does not provide these tools will require reengineering part of our system and it may have an unaffordable cost. In this subsection, we define and analyse the migration process to find the quality aspects that make a service easy to migrate from. We focus on the case where a service is replaced by one or more services offered by a different cloud provider. We consider two situations:

- The current cloud service does not fulfill the requirements of the system: this may happen for instance when the service is updated or modified, when the amount of information handled by the application grows making it impossible to comply with certain pre-established SLAs, etc. Usual examples may range from a variation in the cost that makes the service not competitive compared to other services of the same type, to a change in policies and functionalities that affects security, availability, resilience, or any other important aspect.
- The requirements of the system have changed: one or more cloud services may not fulfill these new requirements and need to be replaced.

Figure 4 depicts a generic process of service-to-service migration. First, a cloud service is selected for migration. Depending on the reason for migration, it may be necessary to review the requirements defined at design-time. After this, one or more new candidate cloud services must be selected. In order to simplify this step, Figure 4 considers a single candidate in the process. Once we have found a candidate target service to migrate to, we can export both data and the configuration from the original service. At this point, it is usually necessary to enter an intentional contract with the new service provider. In some cases, it will be also necessary to inform the old service that we are initiating a process to retire it. In this situation, the old service and the new one

may be active at the same time during the testing and training process. This will depend on the availability requirements of the application migrating one of its cloud-based components. In the next step, it is important to adjust or define a new workflow for the application. This might be necessary if the new service is not perfectly compatible with the old one or if the application was redesigned in a way that the workflow was altered. After this, we can start preparing the testing environment and the new service. Usually, the testing process will be divided in several phases.

In general, it is necessary to carry out functionality and performance testing in a test environment. In this situation, data needs to be kept synchronised. Following successful functionality and performance testing, the service may move to a modification of A/B testing so that the application is tested with the new service in production before switching over completely. In case requirements are not satisfied, we must start the process again. If they are fulfilled, we can start the users training process and eliminate the old service if this is still active. Once this has been done, the application can be deployed again using the new cloud service.

Figure 2 shows several quality aspects related to replaceability. Possible indicators of quality related to intercloud replaceability may include the number of proprietary configurations that can be exported or imported based on a standard format, completeness, precision and relevance of tests, time required to migrate large amounts of data, etc.

C. Security

Preserving security becomes more complex in a multi-cloud environment. Trust among the different cloud service providers is essential. It is difficult to handle the heterogeneity of the different security rules established by each provider, making it complex to monitor security policies in composite services. Besides, an additional challenge involves data and identity privacy preservation when several services from different providers collaborate.

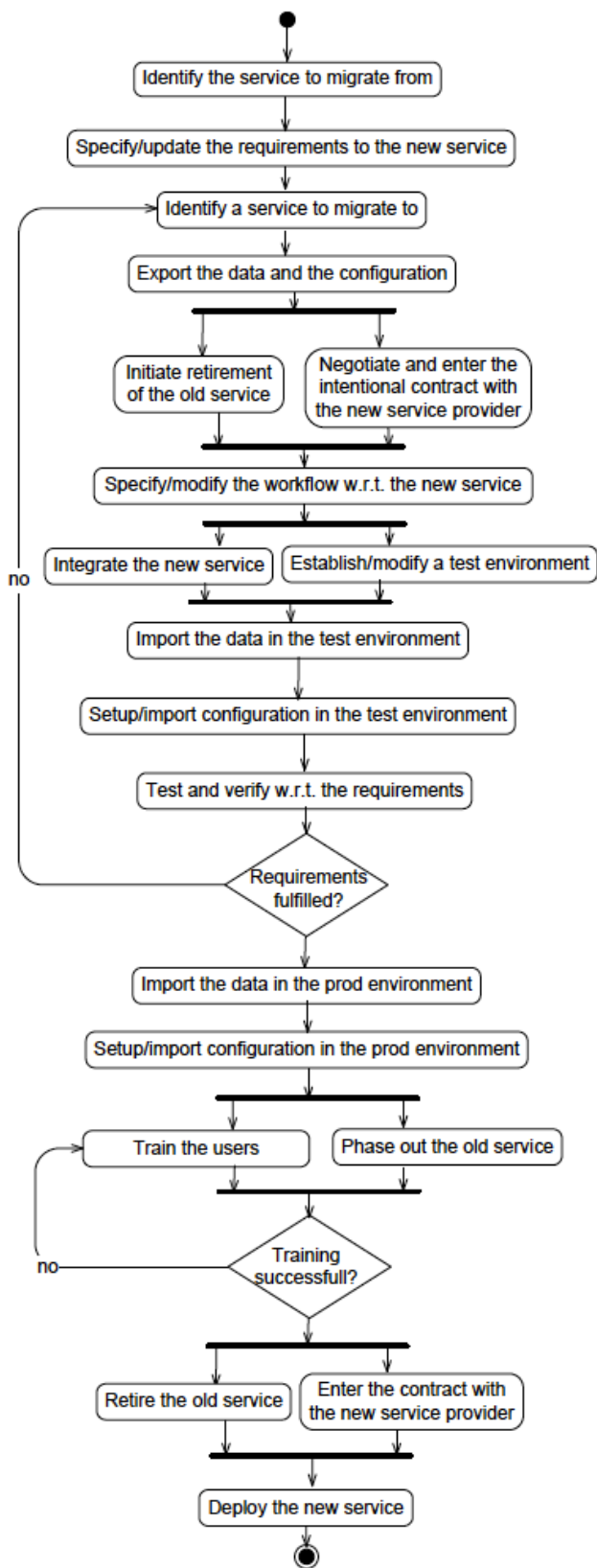


Fig. 4: Description of a generic migration process

In Figure 2, we classify quality aspects related to security in the usual areas: confidentiality, integrity, availability, non-repudiation, accountability and authentication and authorization. In order to preserve data privacy, it is crucial to establish agreements with other providers on the level of privacy of data and identities. Trust in general must be guaranteed by explicit agreements or shared protocols between providers. An alternative solution involves using reliable proxies for communication, but services still need to be able to establish agreements on the fly and secure delegation with these proxies. Finally, it will be important to evaluate services depending on the need to store data in public storage system in order to share this data with other services. In this case, data are exposed to a larger number of threats

V. COST IN MULTI-CLOUD ENVIRONMENTS

Besides risk and quality, we consider another essential dimension: cost. SMI and other previous proposals describe cost-related aspects in cloud computing environments. In a multi-cloud environment, an extra cost appears that may be also considered in the decision-making process: the cost of migration. Migrating from one cloud service to another may involve several economic costs that must be considered at design time. These costs may depend on the personnel involved in the migration process, the cost incurred by keeping the old and the new cloud services running in parallel during the migration process, the cost of the hardware or other resources necessary to perform the migration, or the cost of training the users of the application (note that this cost is also necessary in other situations, but it is usually unavoidable in a migration process).

VI. SPECIFIC RISKS IN MULTI-CLOUD ENVIRONMENTS

In this section, we sketch a list of possible potential risks that may be found in a multi-cloud system. These risks are based on the analysis of the elicited quality aspects that make multi-cloud environments different from clouds provided by a single provider.

1) *Risk of unexpected lack of replacement and consequent vendor lock-in:* a certain cloud service may not fulfill requirements, or requirements may change. In this situation a different service may be needed but it may not be possible to find a new service provided by another vendor which is interoperable with the other services of the system. Two theoretically equivalent services might differ in several relevant aspects. The heterogeneity between different CSPs is usually high as they typically use proprietary interfaces and configurations. Services are also highly integrated with lower-level services offered by the same CSP. Examples of this may be lack of common SLA enforcement systems, use of non-compatible technologies, lack of compatibility in the communications protocol, lack of shared mechanisms to ensure data consistency and quality, the existence of services which are not strictly equivalent and miss some important functionalities, or the lack of services compliant with certain regulations. If this problem appears and the need for migrating from the original service is real, this may even force the migration of other services apart from the service which is not compliant with requirements.

2) *Risk of new security breaches due to the increased complexity of the system and new communications:* data needs to flow from one service to another, hosted by different providers. This creates new points of failure and potential security issues. For instance, this may be caused by the lack of shared security protocols and data integrity mechanisms, lack of forensic mechanisms to be compliant with regulations, the lack of shared authentication systems, etc.

3) *Risk of non-viable migration due to migration costs and complexity:* a developer may not be aware of the cost and complexity of migrating from a certain service chosen to be part of the application to other similar services (see Figure 4). This might become a risk if it is necessary to migrate from that service to another one. As we have discussed, a usual problem in a migration process is the lack of compatible data formats, making it necessary to perform transformations that require time and resources. A related problem might be the lack of information of the new service regarding a certain quality aspect. In this case, uncertainty may also impact a migration process negatively. Note also, that a technical aspect to be considered is whether two services are implemented using the same technology, which might also be a blocking factor for a fast and easy migration. Complexity in the setup migration may also be an important problem. Beyond compatibility in terms of data storage and access, the configuration of a cloud service may also be essential to guarantee the compliance with user requirements. An excessively complex migration of configurations between two services may also result in a time-consuming and expensive migration process. Besides, ease of testing a service and total downtime are two aspects that may largely impact the suitability of a certain migration. Several possible methodologies may be used for developing and support this testing. For instance, modified A/B may be used where only one service is changed and a number of different grades of testing are performed. Finally, depending on the requirements of the application, it might be necessary for the two cloud services, the original one and the replacement, to coexist during a certain period of time, during the testing process of the migration. Complexity to synchronize data between the two services might make the coexistence difficult and using the new service as a hot backup of the first is inefficient.

4) *Risk of costs unpredictability:* by using services from different providers, it may become more and more complex to predict costs.

5) *Risk of lack of provider interest in collaboration:* business agreements are usually required for two CSP to collaborate. For instance, the service delivery model requires customers to register to a service. Because of this, a service in a certain CSP will not allow customers from other CSPs to use it without going through the necessary registration process, unless the right agreements are put in place. Besides, vendors may try to retain customers at any cost to be more competitive. Contracts and other legal issues may be blockers to migrate from one service to an equivalent one. In other words, there is a risk of unfair customer retention and consequent vendor lock-in.

6) *Risk of unavailability of evidences in case of fraudulent actions:* this is a potential risk that may be caused by the lack of forensic tools and global tracking mechanisms.

7) *Risk of lack of negotiation on SLAs:* large organizations using a single supplier can negotiate terms. SMEs or companies using multiple services from multiple vendors are unlikely to have the power or the time to negotiate. This will create an increasingly unstable cost and terms and conditions problem.

Note that a more formal risk analysis might be performed to consider this a final list of risks.

VII. CONCLUSIONS AND FUTURE WORK

In this position paper, we have discussed some essential aspects to establish the necessary baseline for a decision support method aimed at facilitating the selection of cloud services and providers in a multi-cloud environment. In particular, we argue that risk, quality and cost are among the main factors in such a selection process. We believe that a trade-off analysis between risk, cost and quality based on a consolidated view of the three will provide a useful basis for a decision maker in assessing the possible choices through a cost-benefit analysis. For this, we have reported the results of an elicitation of the risk, cost and quality aspects that are specific to multi-cloud environments. We argue that security, interoperability and ease of migration are among the main quality aspects in a multi-cloud environment.

Beyond this initial analysis, we plan to develop a comprehensive study on risk and quality aspects to be considered in a multi-cloud. With this, we aim at creating a decision support tool able to help multi-cloud applications architects to design their systems. This tool will be implemented based on a new methodology that integrates risk, quality and cost dimensions.

ACKNOWLEDGMENT

This work has been conducted as a part of the MODA-Clouds project (Grant Agreement FP7-318484) funded by the European Commission within the 7th Framework Programme.

REFERENCES

- [1] D. Bernstein and D. Vij, Intercloud Security Considerations, Proc. 2nd Intl Conf. Cloud Computing (CloudCom 10), IEEE Press, 2010, pp. 537-544.
- [2] R. Buyya et al., Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility, Proc. 9th IEEE/ACM Intl Symp. Cluster Computing and the Grid (CCGRID 09), IEEE CS, 2009, pp. 599-616.
- [3] Data Protection Act 1998.
<http://www.legislation.gov.uk/ukpga/1998/29/contents>
- [4] European Telecommunications Standards Institute (ETSI).
<http://www.etsi.org>
- [5] Ligaarden, O. S.: A Framework for Analyzing and Monitoring the Impact of Dependencies on Quality. PhD thesis, University of Oslo (2013)
- [6] M.P. Papazoglou and W. van den Heuvel, Blueprinting the Cloud, IEEE Internet Computing, Nov./Dec 2011, pp. 74-79.
- [7] B. Rochwerger et al., ReservoirWhen One Cloud Is Not Enough, Computer, Mar. 2011, pp. 44-51.
- [8] Sarbanes-Oxley Act of 2002 (Pub.L. 107204, 116 Stat. 745, enacted July 30, 2002). <http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/html/PLAW-107publ204.htm>
- [9] Singhal, M.; Chandrasekhar, S.; Tingjian Ge; Sandhu, R.; Krishnan, R.; Gail-Joon Ahn; Bertino, E., "Collaboration in multicloud computing environments: Framework and security issues," Computer, vol.46, no.2, pp.76-84, Feb. 2013
- [10] Cloud Services Measurement Index Consortium: CSMIC Website <http://csmic.org/>. Accessed, March 2013

Moonstone: A Framework for Accelerating Testing of Software

Atsuji Sekiguchi, Tomohiro Ohtake, Toshihiro Shimizu,
Yuji Hotta, Taichi Sugiyama, Takeshi Yasuie and Toshihiro Kodaka
Cloud Computing Research Center
FUJITSU LABORATORIES LIMITED
Kawasaki, Japan

e-mail: {sekia, ohtake.tomohiro, shimizut, yhotter, sugiyamataichi, yasue.takeshi, and tkodaka}@jp.fujitsu.com

Abstract—Enterprises must speed up software development and releases so that they can quickly verify business ideas. We have developed a framework called “Moonstone” that can be used to speed up the testing that is included in a release operation. Moonstone has the following two functions to support testing. 1) Function to construct test environment: this function is used to automatically construct test and production environments on a cloud platform. This function uses hint information of a system configuration included in source code and configuration files, and templates of system configurations. 2) Function to prepare and execute test: this function is used to automatically create and run test scenarios by replaying captured network packets. Because testing in a release operation phase can be done efficiently with these functions, the time required for a release operation can be reduced. We used Moonstone in a trial environment and obtained the following results: 1) a reduction of more than 80% of the time required for the construction of a test environment, 2) a reduction of 33% of the time required for the testing.

Keywords—continuous delivery; software development; software test; cloud platform; traffic replay

I. INTRODUCTION

Global business competition has been intensifying with the spread of Internet and cloud computing [1]. Enterprises must therefore realize business ideas as products and services and then improve them so that they can survive this competition.

Lean Startup [2] describes a method of carrying out product development by verifying business ideas (hypothesis) quickly in the market. In this method, the effects of each hypothesis are quantitatively measured while verifying one hypothesis at a time in the market. An enterprise can learn which hypothesis is effective because the method can individually measure the effect of each hypothesis. Because one hypothesis is verified in the market in a certain period, the amount of development needed to prove the hypothesis decreases, and development can be sped up. Moreover, because the product or service can be quickly released to the market with this method, the enterprise can change its business direction according to the result of the verification.

However, there is the following problem in practicing “lean startup” in software development and release. The release operation contains testing and deployment processes

[3]. Testing is done to inspect software, configurations, and environments from functional [4] and non-functional [5] aspects (such as execution time, performance, quality of service, and security). Deployment is carried out to distribute software and configurations to test and production environments. Even if the system will be slightly changed, testing of the entire system is required to confirm that the changed system will run correctly. Thus, the release operation imposes a constant workload, which is not in proportion to the amount of development. If the amount of development is not changed, the workload increases when the number of releases increases. Thus, the hypothesis cannot be verified quickly.

Therefore, we have developed a framework called “Moonstone” to speed up the release operation. Moonstone has the following two functions to support testing. 1) Function to construct a test environment: this function is used to automatically construct test and production environments on a cloud platform. This function uses hint information of a system configuration included in source code and configuration files, and templates of system configurations. 2) Function to prepare and execute a test: this function is used to automatically create and run test scenarios by replaying captured network packets. Because testing in a release operation phase can be done efficiently with these functions, the time required for the release operation can be reduced. Furthermore, reproducibility of testing can be increased.

The rest of this paper is organized as follows. First, we describe the Moonstone architecture in Section II. Next, we explain the method and result of an evaluation in a trial experiment that uses Moonstone in an environment on a cloud platform in Section III. In Section IV, we discuss related work. Finally, we end with the conclusion and future work in Section V.

II. MOONSTONE ARCHITECTURE

To support phases from development to release of applications, Moonstone can cooperate with various functions such as an issue tracker (Redmine [6]) for development task management, a version control system (Subversion [7] or git [8]) for management of an application's source code, and a continuous integration [10] tool (Jenkins [9]) as shown in Figure 1. In addition, to speed up the testing, Moonstone has two functions: a function to construct a test environment, and a function to prepare and

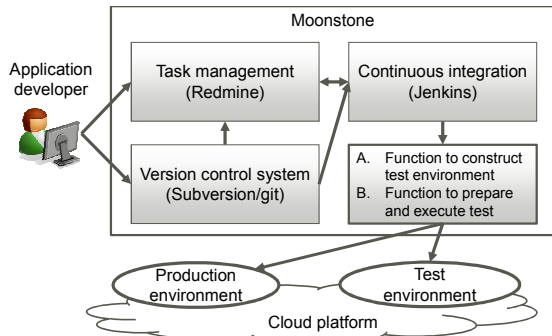


Figure 1. Moonstone architecture

execute a test. Moonstone can realize continuous delivery [11] with these cooperative actions and functions.

A. Function to construct test environment

Application developers generally develop their application in consideration of the characteristics of a system (such as a web server type, an application server type, a database server type, or their connectivity), but they cannot change configurations of the system without permission (such as changing the configuration of a database server from one server to a master-slave configuration). System construction operators are responsible for changing the system. To cooperate with the system construction operators, the application developers must make many preparations such as documentation. Thus, system configuration cannot be changed quickly when a change of system configuration is needed because of the growth of the application.

We have developed a function that extracts the characteristics of an application from its source code and configuration files, and automatically constructs a system by using the most suitable template for the characteristics. The proposed function enables the system to be changed quickly because the application developers can change the system by themselves without needing to have cooperation from the system construction operators.

We will explain the flow of processing on the basis of Figure 2.

1. Application characteristics extraction: In this process, the proposed function extracts the characteristics of an application from its source code and configuration files by using extraction patterns. In many cases, when middleware is used in a system, descriptions for the middleware exist in the source code and the configuration files (e.g., if JDBC (Java Database Connectivity) [12] and MySQL [13] is used in the system, the name of JDBC driver for MySQL appears in the source code or the configuration files, such as “com.mysql.jdbc.Driver”). So, to detect the used middleware, we wrote a pattern beforehand to detect the middleware as an extraction pattern (e.g., detection of “com.mysql.jdbc.Driver” in the source code or the configuration files). As a result, the proposed function can detect the characteristics of middleware composition such as a mail server, a message queuing server, and a cache server. Moreover, we wrote hint information in the source code beforehand for the

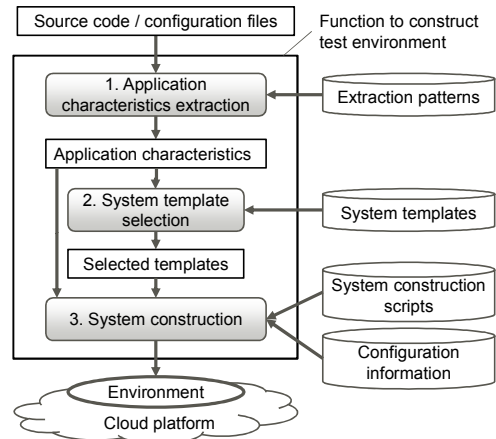


Figure 2. Function to construct test environment

characteristics that could not be detected by the above-mentioned process. The proposed function only supports the Java language currently. In Java, the function utilizes some of the annotations [14] in the source code as hint information. For instance, the hint information is described as the ratio of reading and writing in a class that operates the database, such as “@ReadWriteRatio”. The proposed function extracts the middleware used and the hint information as characteristic information of the application.

2. System template selection: We defined system configuration patterns (such as AWS Cloud Design Pattern [15]) on a cloud platform as “system templates” beforehand. Each system template contains its characteristic information. In this process, this function selects a template that matches the characteristics of the application. For instance, when a name of a database is detected from the configuration files, the proposed function selects a template that installs and sets up the database (e.g., “MySQL”). When hint information such as “@ReadWriteRatio (value = 5.0)” (means “the reading frequency of the database is five times the writing”) is extracted from the source code, the proposed function decides on a template of a composition that suits such purpose (e.g., a database template for a master-slave configuration).

3. System construction: The proposed function constructs a system on the basis of the selected system template. In each system template, we associated the template with scripts that automatically construct the system beforehand. We wrote construction scripts for each server type such as Web, application, and database using Chef [16]. For example, in a case of Web server, our construction script prepares virtual machines (VMs) through the cloud platform’s API, and deploys Web server program (e.g., httpd) and contents to each VM.

This function is useful not only for constructing a test environment for confirming an application behavior but also for constructing a production environment. Furthermore, this function can increase the recyclability of the automation scripts because the function calls the scripts to suit the application’s characteristics.

B. Function to prepare and execute test

A system that has begun providing services for customers should be stable. When an application or its system will be changed, the correctness of the behavior of the application should be checked (tested). To test operations in the entire system, load test tools [17] or devices [18] are usually used to run a test efficiently. The test contains two steps: preparation and execution. In the preparation step, these tools or devices require preparation of test scenarios. In the execution step, the test is executed on the basis of the test scenarios, and success or failure is judged from the results of the test scenarios. The test scenario consists of definitions such as access patterns to the system, request messages for access to the system, and response messages that the system should make in response to the requests. There is a problem that making test scenarios requires a great deal of skill and much time.

Therefore, we have developed a function that automates the test scenario making and the test execution [19]. In the preparation step, a module of this function captures network packets of the request/response messages that are exchanged between clients and servers in the production environment. The request/response messages and their access timing are used as the test scenarios. In the test execution step, the proposed function replays the request messages in the captured packets and compares response messages between the captured packets and the test environment. While replaying, the proposed function translates network/application data of the packets from data in the original environment to data in the test environment. Thus, because the test scenario can be made without a need for much skill or time, the test can be sped up. We implemented the proposed function as a C program on Linux.

We will explain the flow of processing on the basis of Figure 3.

1. Packet capture: In this process, the proposed function captures packets between clients and servers of the system using a packet capture tool (such as Wireshark [20]). At this time, the environment of servers is the production or the test. Users of IaaS (Infrastructure as a Service) [1], which is one of the cloud platforms, cannot capture the packet in the network layer because the network layer is usually hidden in

IaaS. Thus, the proposed function captures the packets in each server and collects them. The packet capture tool can capture packets on various networks (such as Ethernet or Infiniband) because it captures packets OS obtained.

2. Traffic replay: The proposed function generates access loads for the test in the test environment. The access loads are generated based on the test scenario that consists of the request/response messages and their access timing on the captured packets. While sending the packets, the proposed function translates the packets from environment-dependent data in the captured packets to data for the test environment. The translation is based on translation rules. This rule is a definition of how to rewrite data. The data are of the network layer such as Ethernet/IP headers, TCP connections, and HTTP headers, and of the application layer such as session information and authentication tokens. When a response of a server contains session information, the proposed function dynamically rewrites sending packets with the information. We made rules beforehand to match the captured packets to the test environment.

3. Result comparison: The proposed function compares two response messages of the test environment and the response messages in the captured packets. The function regards the test as a success when these responses are the same, and regards the test as a failure when these responses are different. The function can also compare both of access timing of the response messages, and can regard the test as a success when the response time of the test response is less than a threshold time.

When these two functions are combined, the test can be executed on demand. Thus, when testing is needed, the proposed functions construct the test environment on a cloud platform, execute the test, and return the environment after finishing the testing.

III. EVALUATE

To evaluate the two functions described in Section II, we tried these functions in the following environment.

A. Evaluation environment

This environment is a system on a cloud platform to provide a service for consumers through the Internet. The system was constructed for starting the service. The system uses FGCP/S5 (Fujitsu Global Cloud Platform) [21], which is one of IaaS. The system is a typical three-tier one that consists of tens of VMs such as Web servers, application servers, database servers, and load balancers.

B. Evaluation method

To examine the proposed function described in II.A, we constructed a test environment of the same composition as a production environment with the proposed function and via manual operation respectively, and measured the respective elapsed times. Manual operation was executed by experts, who are the system construction operators. We calculated the reduction time and reduction rate by using the proposed function from the result. The construction targets were Web servers, application servers, and databases. Strictly, installed software and settings of each server were a little different

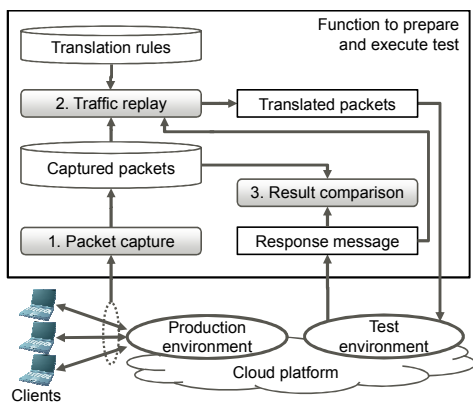


Figure 3. Function to prepare and execute test

from each other. Then, we grouped the tens of servers to three types with the rough role such as Web. Constructing these servers involves starting VMs, making settings for an OS, installing and making settings for middleware such as HTTP server, Java EE server and database server, and installing Web contents and applications. We measured the elapsed time of construction in each type of server. We calculated the average value for the same type of server. The elapsed time of the system was calculated from the type of server and each number.

To examine the proposed function described in II.B, we made and ran the test with the function and via manual operation respectively, and measured the respective elapsed times. Manual operation was also executed by the experts. We calculated the reduction time and reduction rate by using the proposed function from the result. The test contained many test scenarios to confirm the correctness of software, configurations, and environments from functional [4] and non-functional [5] aspects (such as execution time, quality of service, security, usability, and safety). For the manual operation, we made the test scenarios for JMeter [17]. To use the proposed function, the function makes it possible to make the test scenarios automatically by capture and replay of the traffic between clients and servers in the production environment. However, in this trial, we made the test scenarios as follows to clarify a comparison in these two cases: we manipulated a Web browser through each test scenario in manual operation, captured packets through the manipulation, and treated the packets as each test scenario.

C. Evaluation result

The results of the evaluation of II.A are shown in Table I. "Server type" means the type of server such as Web, application, or database. "Elapsed time (manual) [A]" means the elapsed time in the case of manual operation. "Elapsed time (trial) [B]" means the elapsed time in the case of using the proposed function. "Reduction time" is the difference between the elapsed time (manual) and the elapsed time (trial), and is calculated as A-B. "Reduction rate" is the ratio of the reduction time to the elapsed time (manual), and is calculated as (A-B)/A.

An elapsed time of 80% or more was able to be reduced in any server type as shown in Table I. The reduction time of one server was between 4.5 and 9 hours. Because the environment consisted of tens of servers, we were able to reduce the elapsed time of hundreds of hours in total.

The result of the evaluation of II.B is shown in Table II.

TABLE I. SERVER CONSTRUCTION TIME, REDUCTION TIME, AND REDUCTION RATE

Server type	Elapsed time (manual) [A]	Elapsed time (trial) [B]	Reduction time [A-B]	Reduction rate [(A-B)/A]
Web	5h 40m	1h 7m	4h 33m	80%
Application	10h 46m	1h 48m	8h 58m	83%
Database	8h 17m	0h 55m	7h 22m	89%

TABLE II. ELAPSED TIME OF TEST, REDUCTION TIME, AND REDUCTION RATE

Operation type	Elapsed time (manual) [A]	Elapsed time (trial) [B]	Reduction time [AB]	Reduction rate [(A-B)/A]
Test scenario making	40h	16h	24h	60%
Test execution	32h	32h	0h	0%
Total	72h	48h	24h	33%

"Operation type" means an operation of the test such as test scenario making and test execution, and also contains their total. "Elapsed time (manual)", "Elapsed time (trial)", "Reduction time" and "Reduction rate" mean the same as those in Table I.

An elapsed time of 60% (24 hours) was able to be reduced in the making of test scenarios. The elapsed time of the test execution was the same as the manual operation, and the reduction rate was 0%. As a result, the elapsed time of 33% (24 hours) was able to be reduced in total.

IV. RELATED WORK

Continuous integration [10] is the practice of enhancing the quality of source code by automatically integrating, compiling, and testing the source code every day during development. Continuous delivery [11] is a practice that automates deployment of the application in addition to continuous integration. Tools that support these practices exist [9][22]. However, activities such as constructing a test environment and making a test scenario are not being offered by those tools.

There are several approaches to constructing an environment. Tools to support automation of the construction exist [16][23], and a method to automatically construct an environment from a policy-based environment definition is also known [24]. Those approaches require a great deal of skill and much time, because the user has to describe the definition of the composition of the environment correctly. Our method detects the characteristics of the application by using information on system configuration contained in source code and configuration files, and hint information written in the source code as annotations [14]. Then, the method determines the most suitable system template for the characteristics, and automatically constructs an environment based on the selected template. In the case of our method, the user does not need to describe the definition of the composition of the environment.

An approach of the abstraction of clouds' API [25], and an approach of the definition of common data model [26] exist. Though we used the cloud's own API and data model, these approaches can be helpful for portability.

Tools to help automate the making of the contents of the testing exist [17][27][28]. These require a great deal of skill and much time to create a test scenario for the automation. Moreover, it is difficult to imitate real clients' traffic load patterns. Therefore, because some problems are often

overlooked, serious troubles occur when the application runs on the production environment. Traffic replay is an approach that captures packets in the production environment and uses the captured packets for testing [29][30][31][32][33]. However, it is not easy to conduct traffic replay in an ad hoc test environment on a cloud platform. Because the test environment is different from the environment in which the packets are captured, various parameters such as MAC addresses, IP addresses, or TCP port numbers are different. Packets cannot reach servers if the captured packets are simply replayed. In the case of testing an application, application-specific information such as HTTP session IDs and timestamps should be also adjusted to the environment and the time of the testing. Our method can test a long transaction by carrying out a traffic replay with a packet conversion by using not only the difference between these environments but also application-specific information.

V. CONCLUSION AND FUTURE WORK

We have developed a framework called "Moonstone" for speeding up testing that is included in a release operation. Moonstone has two functions: 1) a function to construct a test environment, and 2) a function to prepare and execute a test. In our trial, we confirmed that these functions make it possible to reduce the elapsed time for the testing.

We will tackle the following problems in the future. In 1), when a production environment has a lot of servers, a long elapsed time is required to construct a test environment that is similar to the production environment. In 2), when the proposed function replays captured packets, we must synchronize databases of the test environment beforehand. When the size of the databases is large, the synchronization requires a long elapsed time.

REFERENCES

- [1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition", ACM SIGCOMM Computer Communication Review, Vol. 39 Issue 1, Jan. 2009, pp. 50-55.
- [2] E. Ries, "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses", Crown Business, 2011.
- [3] Cabinet Office, "ITIL Service Transition 2011 Edition (Best Management Practices)", The Stationery Office, 2011.
- [4] W. E. Howden, "Functional Program Testing", Software Engineering, IEEE Transactions on, Vol. SE-6, Issue 2, Mar. 1980, pp. 162-169.
- [5] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties", Information and Software Technology, Vol. 51, Issue 6, Jun. 2009, pp. 957-976.
- [6] Redmine, Available: <http://www.redmine.org/>, retrieved: Mar. 2013.
- [7] Subversion, Available: <http://subversion.apache.org/>, retrieved: Mar. 2013.
- [8] git, Available: <http://git-scm.com/>, retrieved: Mar. 2013.
- [9] Jenkins, Available: <http://jenkins-ci.org/>, retrieved: Mar. 2013.
- [10] P. M. Duvall, S. Matyas, and A. Glover, "Continuous Integration: Improving Software Quality and Reducing Risk", Addison-Wesley Professional, 2007.
- [11] J. Humble and D. Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation", Addison-Wesley Professional, 2010.
- [12] JDBC (Java Database Connectivity), Available: <http://www.oracle.com/technetwork/java/javase/jdbc/index.html>, retrieved: Mar. 2013.
- [13] MySQL, Available: <http://dev.mysql.com/>, retrieved: Mar. 2013.
- [14] Java Annotations, Available: <http://docs.oracle.com/javase/1.5.0/docs/guide/language/annotations.html>, retrieved: Mar. 2013.
- [15] AWS Cloud Design Pattern, Available: <http://en.clouddesignpattern.org/>, retrieved: Mar. 2013.
- [16] Chef, Available: <http://www.opscode.com/chef/>, retrieved: Mar. 2013.
- [17] JMeter, Available: <http://jmeter.apache.org/>, retrieved: Mar. 2013.
- [18] Avalanche, Available: http://www.spirent.com/Products/Avalanche/Avalanche_Latest_Release, retrieved: Mar. 2013.
- [19] T. Sugiyama, T. Yasuie, and Y. Nomura, "A Study for System Verification with Captured Packets" [in Japanese], Proc. of the Society Conference of IEICE 2011 Communication (2), Aug. 2011, p.392.
- [20] Wireshark, Available: <http://www.wireshark.org/>, retrieved: Mar. 2013.
- [21] FGCP/S5 (Fujitsu Global Cloud Platform), Available: <http://welcome.globalcloud.global.fujitsu.com/>, retrieved: Mar. 2013.
- [22] IBM SmarterCloud Continuous Delivery, Available: <http://www-142.ibm.com/software/products/us/en/continuousdelivery/>, retrieved: Mar. 2013.
- [23] Puppet, Available: <https://puppetlabs.com/puppet/puppet-open-source/>, retrieved: Mar. 2013.
- [24] A. Sahai, S. Singhal, R. Joshi, and V. Machiraju, "Automated Policy-Based Resource Construction in Utility Computing Environments", Proc. of Network Operations and Management Symposium, Vol. 1, Apr. 2004, pp. 381-393.
- [25] Deltacloud, Available: <http://deltacloud.apache.org/>, retrieved: Mar. 2013.
- [26] Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol, Available: http://dmf.org/sites/default/files/standards/documents/DSP02_63_1.0.1.pdf, retrieved: Mar. 2013.
- [27] JUnit, Available: <http://junit.org/>, retrieved: Mar. 2013.
- [28] Selenium, Available: <http://seleniumhq.org/>, retrieved: Mar. 2013.
- [29] Tcpreplay, Available: <http://tcpreplay.synfin.net/>, retrieved: Mar. 2013.
- [30] S. Hong and S. Wu, "On Interactive Internet Traffic Replay", Proc. Eighth International Symposium on Recent Advances in Intrusion Detection, Sept. 2005, pp. 247-264.
- [31] IBM Rational Test Virtualization Server, Available: <http://www-01.ibm.com/software/rational/products/rtvs/>, retrieved: Mar. 2013.
- [32] CA LISA, Available: <http://www.ca.com/us/products/detail/CA-LISA.aspx>, retrieved: Mar. 2013.
- [33] Oracle Enterprise Manager: Application Quality Management, Available: <http://www.oracle.com/technetwork/oem/app-quality-mgmt/index.html>, retrieved: Mar. 2013.

Challenges with Tenant-Specific Cost Determination in Multi-Tenant Applications

Anna Schwanengel and Uwe Hohenstein
 CT RTC ITP SYI-DE, Siemens AG
 Munich, Germany
 {anna.schwanengel.ext, uwe.hohenstein}@siemens.com

Abstract— One key element to make Software-as-a-Service (SaaS) successful is so called multi-tenancy, which refers to an architecture model where one software instance serves a set of multiple clients of different organizations (tenants). Hence, it reduces the number of application instances and, in that way, operational costs in a Cloud. The problem SaaS providers are faced within everyday's business is how to define a billing model that has the chance to make profit in a public Cloud. Being profitable with SaaS, the art is to bill tenants in such a way that covers the costs for resources for the underlying PaaS/IaaS provider. This paper discusses some challenges with metering the consumption of tenants as a prerequisite for defining a profitable billing model.

Keywords - Software-as-a-Service; Multi-Tenancy; Billing; Costs; Resource Utilization

I. INTRODUCTION

Since many years a paradigm shift how software is delivered to customers occurs. It changed from installing developed software applications at the customer in-house and operating it on-premise, to a more consumer-based model. Software became an on-demand service drawn from the Internet, i.e., Software-as-a-Service (SaaS) [1]. SaaS is a delivery model that enables customers to rent services without local installation and license costs.

In this context, multi-tenancy is a key element to achieve a successful SaaS business, though not being the guarantor for more revenue. Multi-tenancy means multiple tenants from different organizations share a system operated by one company. The respective application is used by several tenants of a SaaS provider [2]. Thereby, each tenant serves plenty of users who actually use the software. A multi-tenant architecture postulates that the application is able to partition its data and procedures virtually. Each tenant gets a virtual instance, which can be customized according to his wishes, running on the same physical instance, while not being influenced or even aware of the other tenants working concurrently.

In single-tenant systems, each tenant obtains its own instance running the application (or database), which reduces management efforts regarding the mapping of the resources to each tenant. However, looking at the overall efficiency, one can observe some drawbacks, as in a lot of cases many server instances will be low utilized at most time points [3]. This system utilization can be improved by operating a multi-tenant service, where fewer instances are used to serve tenants in a shared environment. Moreover,

operational costs can be saved when the SaaS provider deploys an application on the PaaS or IaaS layer of a Cloud provider. A SaaS provider pays for the resources his SaaS application uses. That means being charged by CPU time, number of transactions, database space etc. The more payable resources are shared, the less costs an application produces. One important aspect is to design the architecture in a way that uses the resources efficiently [4].

In this paper, we focus on another economical problem of SaaS providers, which has been paid less attention in the research area. On the one hand, we have *cost models* defined by IaaS/PaaS providers, a SaaS provider has to pay for when running applications. But a SaaS provider has also to define a *billing model* to charge his tenants for application usage. Both models have to be balanced in a way that SaaS providers obtain a suitable return of investment and are able to make profit while having an attractive billing model for tenants. The investment covers both, the Cloud operational costs and costs for application development or SaaS-enabling of existing applications.

We are approaching this aspect from a technical view. A lot of billing methods have been discussed in the literature such as pay-as-you-go, pay-per-user, pay-per-feature, or a fixed monthly fee [5]. All have in common that a SaaS provider has to keep an overview over total costs and tenant-specific costs in order to offer a profitable billing model. Section II stresses this point and motivates the need for tenant-specific metering of resource consumption.

We present challenges for SaaS providers to balance outgoing costs for the underlying PaaS/IaaS provider and ingoing revenue from the tenants. We choose Windows Azure for this investigation because of its PaaS offering that ships with a complete development and deployment environment. There are no problems with product licensing, as this is part of the platform and the cost model, which makes the cost calculation easier – see Section III.

Section IV gives some insight into cost reasoning for multi-tenancy within Azure. Section V discusses what technical concepts of Azure can be used to monitor tenant-specific resource consumption. A prerequisite, how tenants can be identified, is explained in Section VI. Section VII provides an overview of related work in the multi-tenancy area before Section VIII concludes and names future work.

II. PROBLEM SPACE

It is commonly agreed that a well-economical SaaS provider has to support multi-tenancy, i.e., giving tenants a tailored, best-fitting application satisfying their specific

requirements by customization, while sharing as much resources as possible to achieve higher capacity utilization. Thereby, SaaS provider have to reflect upon easy implementation (as in single-tenant systems, where every tenant holds its own application) and costs (which is more adjusted in multi-tenant systems serving all tenants by one instance). That is in accord with economy of scale sharing both the underpinning infrastructure as well as the hereon running software. This point can also be seen in [6] and [7], where several architectures are distinguished regarding what is shared by tenants: the topmost web frontend, middle tier application servers, and underlying database. Nevertheless, when supporting all tenants by one instance in a multi-tenant system, the question is how to charge each tenant, while targeting at profit. Defining a billing model is easy but how to monitor whether it is reasonable?

Several billing models have been proposed. Most of them are post-paid models. Thereby the tenant receives a bill and pays for usage periodically [8]. To invoice the consumption costs, usage of each tenant is observed and aggregated [9]. The safest method from a SaaS provider's perspective is to charge tenants the same pay-as-you-go way as PaaS/IaaS providers do for their resources, i.e., OPEX are directly forwarded to tenants, plus an additional charge. Such a model is very technical and not cost-transparent for tenants. From a SaaS provider's view, this situation is complicated when several tenants are served by one instance. Therefore, it is important to estimate or even compute the resource costs (e.g., for consumed storage, or CPU), in particular how many resources one tenant uses. This implies the monitoring of each tenant and logging the way they use the application. More precisely, it requires observing the resource usage of the applications for each tenant, and raising an invoice based on usage metrics.

Alternatively, billing models can be based upon factors that are better understandable by tenants, like usage time. The problems for SaaS providers remain the same, and the Cloud cost model must be transformed to a billing model.

A SaaS provider can also charge its tenants by a fixed rate, e.g., per month. However, it is difficult to predict the costs a tenant's usage will produce. Moreover, exhaustive usage by one tenant could reduce the SaaS providers' revenue, even to minus. On these grounds, a precise cost control of each tenant can be used to throttle frequent users to reduce this risk – if SLAs are defined accordingly.

In a pay-per-user billing model, users must be registered and the number is then known. However, there is again a risk of undercharging over-utilizing tenants.

Billing may also be conducted in a pre-paid method. Pre-paid clients load a deposit onto their accounts previous to any consumption. During the usage, this credit is debited and in case of reaching a limit, the tenant has to reload money for service use. Although the pre-paid model sounds promising to SaaS providers offering profit-ability, the post-paid model is more common. Anyway, one has to check whether a tenant's limit has been reached.

All this comes along with a big problem for the SaaS provider: he has no clue whether his offering is profitable. A detailed monitoring of costs produced by tenants is

necessary, independent of the billing method. Besides, cost models of IaaS/PaaS providers are quite complex and take technical parameters into account. This makes it not only difficult to estimate the costs for a given application [10], but also to derive costs for each tenant. The different cost factors that PaaS/IaaS providers charge (which differ enormously from provider to provider) make it difficult to run a clear-cut course. Several systems (e.g., EC2) bill according to a usage-of-instance charge and raise the price additionally based on the absolute number of transferred bytes and not adapted on duration or network activity [11].

Another aspect, which requires closer attention, is that an overview of the total amount of used resources and resulting costs is usually only given on a monthly basis. With only getting a monthly bill from a PaaS/IaaS provider with an aggregated cost report over the consumed resource capacity for his tenants, a SaaS provider could not get any detailed data about the cash accounting. Thus, a SaaS provider could not counteract in time, when his service is getting unprofitable by tenants with frequently active users. There is a strong need for a tenant-specific accurate cost model, which is required for:

- a consumption-based model that charges back tenants for their consumed resources;
- a tenant-specific profit-making check, which illustrates, whether the chosen business model for one/all customer(s) is appropriate to make profit;
- a timely reaction in order to throttle frequent and too expensive tenants; Throttling just at the end of a month will be too late to compensate losses.

This paper deals with these challenges of estimating costs on a per-tenant basis. In particular, costs have to be conducted in an efficient manner that only means a minimal amount of extra burden, to avoid latency and costs.

III. MICROSOFT AZURE AND ITS COST FACTORS

Since we base our investigation on a concrete PaaS platform, Microsoft Azure, we here briefly present basic concepts and the cost model of the Azure Cloud platform according to the status quo when writing this paper [12].

Compute instances (VMs including equipment), called Web and Worker Roles, are charged for the number of hours they are deployed. As seen in Table I, there are several instance categories: A *small* instance (default) costs \$0.12 per hour; the more powerful *medium*, *large*, and *extra large* instances have twice the price as the preceding category, i.e., an extra large instance is charged for \$0.96 per hour (i.e., factor 8 compared to a small instance). The instance categories scale in a linear manner with regard to equipment. That is, a medium instance (M)

TABLE I: PRICES FOR COMPUTE INSTANCES

	CPU	RAM	HDD (GB)	MBps	\$/h	I/O performance
XS	Shared	768MB	20	5	0.04	Low
S	1,6GHz	1,7 GB	225	100	0.12	Moderate
M	2 x	3,5 GB	490	200	0.24	High
L	4 x	7 GB	1000	400	0.48	High
XL	8 x	14 GB	2040	800	0.96	High

has double of CPU, disk etc. than a small instance (S) resulting in a double price. The exception is an *extra small* instance (XS) category. The prices are taken on an hourly basis. Even if a compute instance is used for only 5 seconds, a full hour has to be paid.

For Azure table, blob and queue storages, costs depend on bandwidth, storage consumption and transactions.

Storage is billed based upon the average usage during a billing period. If, e.g., 10 GB of storage are used for the first half of a month and none for the second half, 5 GB of storage are billed for average usage. Azure measures the consumption at least once a day. Each GB of storage is charged with \$0.07. Please note that storage consumption takes into account the physical storage, which consists not only of raw data; the length of property names, and the property data types also affect the size of actual data [13].

Any access to storage by transactions has to be paid: 100,000 transactions cost \$0.01. Bulk operations, which bundle inserts, count as one transaction.

The outbound transfer to the North America and Europe regions is charged with \$0.12 per outgoing GB, the Asia Pacific Region is more expensive. It is important to note that the transferred data has some typical XML overhead according to the protocol. Data transfer is for free within the same affinity group, e.g., for compute instances that run in the same data center. All inbound data transfers to the Azure Cloud are also at no charge.

The costs for an Azure SQL Database, a virtualized SQL Server, are also based on monthly consumption. Up to 100 MBs are charged with \$4.995 a month. Up to 1 GB, the overall price is \$9.99. Any GB exceeding 1 GB costs \$3.996. Having reached 10 GB, the prices again decrease to \$1.996 per additional GB, and beyond 50 GB, a GB costs only \$0.999. This means, a 10 GB is charged with \$45.954: \$9.99 for the first GB, and $9 * \$3.996$ for the remaining 9 GB. Azure instance is charged monthly for the number of databases and amount of data used a day. Further charged services exist, e.g., for authentication by Azure Access Control, but they are out of scope here.

These cost factors are important for SaaS providers to determine the price for a deployed application in a rented PaaS/IaaS environment. Knowing the precise costs for the SaaS application is the core element when a SaaS provider forms a billing model for its tenants. Only in this case, the SaaS provider can create an economical billing method of accounting with high profit.

IV. REASONING FOR MULTI-TENANCY

Multi-tenancy is often presented as a solution to make profit or to deploy SaaS applications economically. The statement is more or less generally accepted. Anyway, we want to provide some calculations to show the effect of multi-tenancy in case of Microsoft Azure.

At first, we consider storing data in an Azure SQL Database. The costs are primarily based on storage consumption. But there is no cost difference between storage in one or in several databases, no matter whether placed on one database server. Hence, there seems to be no cost-benefit for sharing one database or server between

several tenants. Hence, a question is arising: Are several databases (one per tenant) really more expensive than keeping all tenants' data in one large database?

First, pricing in Azure occurs in increments of 1 GB. Thus, four 1.1 GB databases are charged with $4 * 2 = 8$ GB, i.e., $8 * \$9.99 = \79.92 a month, while a single database of 4.4 GB is charged with 5 GB. Next, the storage price decreases with the size. Assume there are 4 tenants with databases à 3.1 GB, 4.3 GB, 38.3 GB, and 87.2 GB respectively. The monthly storage costs for having for each tenant a database of its own are:

$$\begin{aligned} 3.1 \text{ GB: } & 1 * \$9.99 \text{ (1st GB)} + 3 * \$3.996 & = \$ 21.978 \\ 4.3 \text{ GB: } & 1 * \$9.99 \text{ (1st GB)} + 4 * \$3.996 & = \$ 25.974 \\ 38.3 \text{ GB: } & \$45.954 \text{ (1st 10 GB)} + 19 * \$1.996 & = \$ 83.878 \\ 87.2 \text{ GB: } & \$125.874 \text{ (1st 50 GB)} + 38 * \$0.999 & = \$163.836 \end{aligned}$$

This is in total \$295.666. In contrast, a single database for all the 132.9 GB costs

$$\$125.874 \text{ (1st 50 GB)} + 83 * \$0.999 = \$208.791.$$

This means a 26% cost reduction of \$87. However, that rough comparison does not take into account that record sizes increase slightly for the one-in-all database due to the *tenantID* for distinguishing tenants. Keep also in mind that there is a limitation of 150 GB per database, which hinders putting a higher amount of tenants with larger storage consumption in one database!

The constellation is similar for table storages, albeit, the cost decrease is much lower: Here, 1 GB costs 7ct. Any additional GB exceeding 1 TB is charged with 6.5ct. Beyond 50 TB, the price is 6ct. Storing 10 TB in ten 1-TB tables (\$700) makes a difference to one 10-TB table (\$655). This plays a role only for larger data volumes.

For compute instances, 12 ct per hour are charged for a small instance, i.e., \$1,051.20 per year. Saving instances by sharing services is, therefore, reasonable. There is a real cost difference when the provider could serve ten tenants with one instance (\$1,051.20) instead of giving each tenant an instance of its own ($10 * \$1,051.20$).

We are faced with an additional hard decision in determining whether to rent a higher amount of less capable computing instances or to take rather fewer high performing instances. From the cost's view point at a first glance, it makes no difference whether a SaaS provider rents four small (S) instances or one large (L) instance; the SaaS provider has to pay the same price for the same capacity. However, if additional instances are required, due to heavy load by tenants or serving an increasing number of new tenants, a SaaS provider has to add extra instances. In this process, however, the type of instance (XS, S etc.) is already determined at deployment of the application. If applications are designed for L instances, the SaaS provider has to start a further L instance, even though a cheaper S instance would have been sufficient to serve the additional tenants' users. That will result in a less profitable service provisioning and in less revenue. Generally, constant system utilization is improbable and high variations in service usage often occur [14].

V. CHALLENGES FOR TENANT-SPECIFIC COST ESTIMATION

In Section II, we motivated why monitoring tenant-specific consumption costs are useful. In this section, we discuss the features Microsoft Azure provides to this end, thereby concentrating on real multi-tenant systems with tenants sharing instances. We give some insight in what support is available, to what extent, and what is missing.

A SaaS provider has only some basic support by Azure. He gets a bill once a month for the monthly consumption of all cost factors: CPU time, storage, database units, outgoing data transfer, and number of transactions. Furthermore, there is a management API giving access to the recent deployment including information about the number of instances of what size and the starting time. Enabling performance counters allows for tracking aggregated usage for Blobs, Tables and Queues [15]. Please note all this is consolidated for one storage account; the limit is 5 for Azure subscribers. Hence giving each tenant a subscription of his own is unfeasible.

The structure of this section follows the Azure cost factors and distinguishes some multi-tenancy approaches.

A. Azure SQL Database

1) Each tenant obtains a physical DB of its own

In this case, the database size can easily be determined by means of a SQL query using the dictionary information. However, one important question remains: When should be the consumed storage measured?

The cost model says that the storage consumption is measured once a day by Azure, but the time point is unknown, because Azure argues that the charge amortizes during the month. However, the storage consumption might vary a lot day by day and in fact within one day. According to this, even if we periodically check the consumption each day, we do not know when Azure is measuring, and this is relevant for our bill. If we take the values for the consumption at noon, the consumption might be completely different to midnight; maybe this is the time Azure measures our occupancy. To solve this point, Azure's internal measuring must be laid open.

In addition to the storage consumption, Azure also charges for the outgoing data transfer. Outgoing means leaving the data center. This cost fact can be ignored unless the SaaS application offers tenants a direct access to the database, which is albeit rather unusual, e.g., due to isolation and security issues [16].

2) Tenants share a common database

If a common database is shared by multiple tenants, it is more difficult to determine a tenant's part of the database. Assuming that each tenant is maintained by a unique tenant identifier (*tenantID*), it is possible to count the number of records in each table in order to get a rough impression. Nevertheless, this number does not reflect the storage consumption since the length of records might vary from tenant to tenant. A more complex and time-consuming query can sum up the length of all values. Furthermore, the storage for indexes remains unknown.

Moreover, the same questions as above remain about when to measure the numbers for database consumption; we again do not have any information at which time point Azure's measurement takes place.

B. Azure Table Storage

The table storage usage is charged by outgoing data transfer, memory usage, and the number of transactions.

1) Each tenant obtains a physical table set of its own

Unfortunately, there is no efficient way to measure the physical table size. The management API does not yield concrete measurements or consumption numbers, but only a monthly summary for a complete storage account (with several tables). To counteract this lack in tenant-specific billing, some solutions are possible, even though problems remain. First, tenant records can be counted, which means accessing the complete table. This can raise transactional costs, and performance impacts may occur. Besides, still some uncertainty remains due to unknown record sizes.

A more efficient approach is to enumerate records during insert. Then, we are able to ask for the latest record by a timestamp-query; this is approximately the number of records. However, we have no numbers for already deleted records. More cost-intensive is to maintain two counters for insert and delete operations, which doubles the transactional costs. Nonetheless, the number of records is only a rough estimation, and the problem how to compute the specific record sizes still remains.

Consequently, there is a strong need to add further tracing for tenant-specific storage actions. A modular possibility may be to use aspect-orientation to intercept operations [17], however, we are then only able to measure accesses via the C# storage library, but cannot quantify REST calls to the storage. A simpler form is to register event handlers for inserts, which is a rather rudimentary, limited mechanism. When implementing event handlers to observe storing and deleting operations in the table storage, the event handler requires the *tenantID* as a prerequisite for enabling a tenant-specific billing. Anyway, the best way is to add some kind of monitoring in the application, whereby one important problem still remains: When to measure the tenant's consumption?

2) Tenants share a common table storage

In the case of tenants sharing a table, we find the same problem as above. We have to query complete tables to count records, now for one tenant. The counting can be conducted more efficiently if the *tenantID* is taken as the *PartitionKey*. Then, calculation can be done in one partition, reducing search space and raising performance.

3) Transactions

Another cost factor is the number of transactions on the table storage. Charging 100,000 transactions with 1ct appear like micro-costs at a first glance. But investigations show that transactions could be the dominating cost factor in Azure [10]. Moreover, the term transaction must be taken carefully. Every operation to the storage, even asking for the list of tables, is considered as a transaction. Some operations can be performed in bulks; each bulk is

then a transaction. And finally, each query is a transaction whereby a continuation token is returned if the result is too large or runs too long. Then, successive queries become necessary, which are counted as transactions as well.

There exist performance counters (Azure storage metrics) which however track only the number of transactions for one storage account. This might be an efficient way to compute the overall transactions on a daily basis, but does not yield any tenant-specific information. Further, tracking the number of transactions must again be done by introducing specific tracing in order to get precise data.

4) *Outgoing data transfer*

The final cost factor is the outgoing data transfer (leaving the data center). These costs are presumably irrelevant unless queries on the table storage are directly performed by tenants, which is rather unusual.

C. *Blob and Queue Storage*

Principles and techniques for handling cost aspects for blobs and queues are quite similar to Azure table storage and the same mechanisms as explained in *B* can be applied. However, the queue storage consumption seems to be irrelevant since queues will usually not keep large amounts of data, unless there is some congestion in the system. The dominant cost factor will be the transactions.

D. *Compute Instances*

In Azure, computing power is organized by means of Web and Worker Roles, as described in Section III. The major cost factor is the number of hours a role runs. Any application can be distributed over several Web and Worker Roles. Furthermore, an application can scale out by setting up additional instances of an implemented role to handle sporadic load peaks [14] with a load balancer.

The Azure management API yields some information that can be used to monitor costs such as the size of a role (S, M, L etc.), the number of instances for each role, their status (running, suspended etc.), the starting time, etc. In principal, it is enough to poll the data when the current consumption is needed. However, we are not aware of removed instances and roles since they silently disappear from the report. In order to get notice of any decrease of instances, it is necessary to poll periodically. Some uncertainty remains as an instance can run only for one minute, being charged with one hour. This event will presumably get lost unless we check within that minute.

Generally, this data does not reveal any tenant-specific information; it just shows values of the overall consumption of a multi-tenant application. If there is a relationship between Web/Worker Roles and tenants, such a separation would be possible, however, thwarting principles of multi-tenancy. To obtain tenant-specific information, additional logging should monitor the number of requests. This kind of data is available in performance counters, but again only covers the whole application.

Please note, there is no obvious relationship between VM operation costs and how much a tenant contributes to

these by measuring CPU time etc. Hence, these are only rough indicators for a tenant's portion of usage.

E. *Further Notes*

It is important to note that measurements themselves could affect the costs. Consequently, there is a trade-off between collecting precise data and being cost-efficient. This basically concerns the frequency of periodical measurements, the efficiency of queries etc.

VI. DETERMINING A TENANT

One important issue for the previous discussion is how to extract a tenant, which uses the application, from the service URL. The following discussion summarizes relevant aspects. Thereby, we investigate four ways of defining SaaS URLs [18] and how to extract a tenant.

A. *Using a General URL*

A SaaS provider may offer a general URL in the manner of <http://www.SaaSprovider.com>. Each tenant has to register all of his users for the specific services with user and password; particularly, each user obtains a unique tenant identifier (*tenantID*). The assumption is that each user is exclusively associated with a single tenant. Using a service such as <http://www.SaaSprovider.com/Service1>, a tenant's user has to log in with his credentials. A central component is then able to determine the user's *tenantID*. While this implementation is rather simple, several fundamental problems are obvious in this approach.

At first, the service itself must be generic and unbranded until the user has logged in. Similarly, a tenant-specific customization can only take place after login. Before login, the service can only be general due to the unknown *tenantID*. As a direct consequence, it is difficult to have more than one identity provider (such as an own Active Directory). The identity provider cannot be known before the tenant is known. But in most cases, tenants want to specify the identity provider fitting to their infrastructure. Next, it is immediately visible that the user is accessing a multi-tenant application because the URL does not contain the tenant. Furthermore, there is no way to allow for anonymous users that have no account and consequently no relationship with a tenant. Hence, SaaS providers are restricted to supporting all solvent users. Finally, a user cannot have a relationship with more than one tenant unless they have different credentials.

To sum it up, although a tenant can easily be identified by picking up the login credentials of the users, this approach has some drawbacks and is unsatisfying. For that reason, we consider further possibilities as following.

B. *Tenant Parameter in the URL*

As an alternative to the first approach for URL design, the URL can per default contain the tenant's name as an identifier in two different ways, for instance:

- <http://www.SaaSprovider.com/tenant1>
- <http://www.SaaSprovider.com?t=tenant1>

Now, the application knows immediately who the accessing tenant is, and customization can take place for a

tenant just as various identity providers are possible for authentication. Furthermore, we can observe the advantage that anonymous users are possible as they do not depend on an identifiable relationship to a tenant. Additionally, users can have accounts with more than one tenant, because their access is scoped by tenant.

Unfortunately, there are still some problems. At first, it is still obvious that this approach is a multi-tenant application, because the URL specifies a host *SaaSProvider* that has no meaning to the user. That is why the user cannot deduce the service, which he actually wants to use, from the given URL. Next, both of the above provided URLs are difficult to guess, i.e., users will be unable to find the application by means of ‘URL surfing’. If the user just haphazardly tries out random URLs such as *www.tenant1.com/service1* or *http://service1.tenant1.com*, he will never score a hit, because the service is URL-invisible. It is to note that the URL is an important part of a company’s brand. Having URLs such as *http://www.SaaSProvider.com/tenant1* with someone else’s host name in a URL (here *SaaSProvider.com*) is only a “second class” branding and insufficient for big companies.

However, identifying a tenant with an ID in the URL is possible by extracting the tenant’s name by means of ASP.NET MVC URL Routing.

C. Tenant in a Sub-Domain

A better approach is to embed the tenant identifier (*tenantID*) in the URL as a sub-domain: *http://tenant1.SaaSProvider.com*. Moreover, it is possible to apply a DNS alias to redirect the URL to *www.SaaSProvider.com*. Advantages of this proposal are obvious: It is still possible to identify every single tenant, whilst the URL is branded since the tenant name, *tenant1*, appears directly within the URL, and it is now less obvious that *tenant1* is one of many tenants that are using the application. The URL can be found out with URL guessing and by trial and error.

Even the technical challenge of extracting tenants from the URLs can be solved, since the tenant is passed with the HTTP request in the Host Header, albeit it is more complicated.

D. Tenant in a Domain

Finally, a tenant may use its domain, e.g., *http://www.tenant1.com*. The URL can be mapped to *www.SaaSProvider.com* in the tenant’s DNS configuration. Here, the tenant can be identified by using the *Request.Url C#* class.

In summary, it can be stated that tenant identification is possible for all four approaches; this is the basis for our considerations to realize tenant-specific billing. However, the approaches are characterized by different quality and accordingly efforts and costs. This has to be considered when deciding how to conduct tenant identification.

VII. RELATED WORK

A lot of research is done in the field of multi-tenancy, where also traditional aspects of distributed computing remain important. Fehling et al. come up with prospects for the optimization of multi-tenants by the distribution of

the tenants regarding Quality of Service [18]. Additionally, security and privacy issues should also be regarded for multi-tenancy. To this end, Jensen et al. present an overview of technical security problems [20]. Besides, requirements for efficient multi-tenancy regarding performance or isolation are explored by Guo et al. They present a design and implementation framework to support multi-tenant services [2]. Since multi-tenancy is linked to large client amounts, economic concerns raise importance, too, as providers need to operate with high profit to remain competitive. To reduce overall resource consumptions in multi-tenant environments, [21] introduces a method for implementing cost-efficient multi-tenancy by optimized tenant placement. Also [22] puts values of utilization and performance models in genetic algorithms to reduce thereby costs, albeit, they do not concern tenant-specific billing.

Other researchers consider solutions to implement cost-efficient multi-tenancy, looking at the infrastructure, middleware and application tier, which all can be shared among tenants [23][24]. However, for fault-tolerance, one still needs an existence of the same application on different instances – regardless of the particular tier. So, if an application transparently moves to another instance, this must be traced and considered in the bill to fit a tenant-specific pricing. This problem is not considered there.

In general, providers bill their tenants in different models. The most common pricing models are either the tenants paying a fixed monthly fee, or in a pay-per-use model, where the tenant only pays for the resources he had used, or even the resources may be charged usage-based [25]. With multi-tenancy, SaaS providers’ profit may be increased, but on the other hand, one has to monitor each tenant resource usage and relate this to his monthly bill. Therefore, Cheng et al. set up a monitoring framework to trace tenants’ allocations at runtime and to observe the performance of each tenant based on the individual SLAs [26]. However, they do not provide a tenant billing model.

Bezemer and Zaidman, discuss, based on existing single-tenant applications, another aspect of costs associated with multi-tenant applications: maintenance efforts. The recurrence of maintenance tasks (e.g., patches or updates) raise operating costs and show the demand of exact planning of maintenance costs, which must be apportioned among the tenants [27][28].

Nevertheless, the profitable aspects for the SaaS providers are researched insufficiently in the field of multi-tenancy. Reflections about their balancing act between making revenue through tenants’ charges and paying for the tenants’ used capacity at the PaaS/IaaS provider are extremely understudied until now. Therefore, we came up with an overview of the remaining challenges for the SaaS providers, which want to offer their services to multiple tenants in an economical business model.

VIII. CONCLUSION

In order to save costs and run economical businesses, SaaS providers rely on multi-tenancy, albeit it is no recipe for more revenue. By building multi-tenant applications, a SaaS provider can support multiple tenants from different

organizations with shared instances, being simultaneously used. This and a better utilization by tenants through re-use may lead to higher revenue for SaaS providers.

Within the paper, we depict considerations that enable SaaS providers to succeed in balancing outgoing costs for the PaaS/IaaS resources and ingoing revenue from tenants to operate economical business. We motivate why it is necessary to monitor the detailed costs per each tenant in a more fine-granular manner. We focused on Microsoft Azure and came up with reasoning for multi-tenancy and discussed features of the Azure infrastructure. Until now, SaaS providers receive monthly bills from Azure about the past resource usage by its tenants. This is insufficient because no precise and in time tracking of tenant-specific costs is available. Although some tenant-specific costs can be determined with more or less effort, they might be expensive and lead to additional costs for the SaaS provider. Anyway, for multi-tenant SaaS providers some uncertainty about costs remains and their challenge is still to observe how much a tenant uses of a specific resource type in order to achieve high profitability.

As future work, we plan to also analyze other Cloud platforms such as Amazon IaaS/PaaS regarding its support to trace costs by each tenant. Further, we want to conduct experiments and analyze the corresponding data to give some concrete suggestions how to integrate tenant-specific billing in new and even already existing applications. We will also investigate and compare multi-tenant application built upon a PaaS Cloud and an IaaS platform in order to give an even more precise insight in cost factors. We think the PaaS version will produce more expensive bills, but will also decrease development costs than the IaaS version. Moreover, we work on adequate possibilities for application-specific logging. All this work should finally lead to a consumption-monitoring system.

REFERENCES

- [1] A. Dubey and D. Wagle, "Delivering software as a service," In: *The McKinsey Quarterly*, 2007, pp. 1-12.
- [2] C. Guo, et al., "A framework for native multitenancy application development and management," *Proc. on Enterprise Computing, E-Commerce and E-Services*, 2007, pp. 551-558.
- [3] B. Wilder, "Cloud Architecture Patterns: Using Microsoft Azure," Sebastopol: O'Reilly Media Inc., 2012, pp. 77-79.
- [4] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented clouds: Vision, hype, and reality for delivering IT services as computing utilities," *Proc. on HPC*, 2008, pp 5-13.
- [5] <http://appenda.com/library/software-on-demand/saas-billing-pricing-models> [retrieved: March 2013]
- [6] S. Walraven, E. Truyen, and W. Joosen, "A middleware layer for flexible and cost-efficient multi-tenant applications," *Proc. 12th Middleware*, 2011, pp. 370-389.
- [7] S. Walraven, E. Truyen, and W. Joosen, "Towards performance isolation in multi-tenant SaaS apps," *Proc. on Middleware for Next Generation Internet*, 2012, pp.1-6.
- [8] M. Lindner, F Galán, and Clovis Chapman, "The cloud supply chain: A framework for information, monitoring, accounting and billing," *Proc. on Cloud Computing*, 2010.
- [9] I. Ruiz-Agundez, Y.K. Peña, and P. Bringas, "A flexible accounting model for clouds," *SRII*, 2011, pp. 277 - 284.
- [10] U. Hohenstein, R. Krummenacher, L. Mittermeier, and S. Dippl, "Choosing the right cloud architecture - A cost perspective," *Proc. on Cloud Computing and Services Science (CLOSER)*, 2012, pp.334-344.
- [11] S. Seetharaman, "Energy conservation in multi-tenant networks through power virtualization," *Proc. on Power aware computing and systems, USENIX*, 2010, pp. 1-8.
- [12] Azure Pricing, 2013, www.windowsazure.com/en-us/pricing/details [retrieved: March 2013]
- [13] B. Calder, "Windows Azure Storage billing," <http://blogs.msdn.com/b/windowsazurestorage/archive/2010/07/09/understanding-windows-azure-storage-billing-bandwidth-transactions-and-capacity.aspx> [retrieved: March 2013]
- [14] A. Schwanengel, U. Hohenstein, and M. Jäger, "Automated load adaptation for cloud environments in regard of cost models," *Proc. on CLOSER*, 2012, pp.562-567.
- [15] J. Haridas, M. Atkinson, and B. Calder, "Azure Storage metrics," <http://blogs.msdn.com/b/windowsazurestorage/archive/2011/08/03/windows-azure-storage-metrics-using-metrics-to-track-storage-usage.aspx> [retrieved: Mar. 2013]
- [16] J. Schroeter, S. Cech, S. Götz, C. Wilke, and U. Abmann, "Towards modeling a variable architecture for multi-tenant SaaS applications," *Proc. on Variability Modelling of Software-Intensive Systems*, 2012, pp. 111-120.
- [17] U. Hohenstein and M. Jaeger, "Using Aspect Orientation in Industrial Projects: Appreciated or Damned?," *Proc. on Aspect-Oriented Software Development*, 2009, pp.213-222.
- [18] Designing Multitenant Applications on Windows Azure: <http://msdn.microsoft.com/en-us/library/windowsazure/hh689716.aspx> [retrieved: March 2013]
- [19] C. Fehling, F. Leymann, and R. Mietzner, "A Framework for Optimized Distribution of Tenants in Cloud Applications," *Proc. on Cloud Computing*, 2010, pp. 252-259.
- [20] M. Jensen, J. Schwenk, N. Gruschka, and L. Iacono, "On technical security issues in cloud computing," *Proc. on Cloud Computing*, 2009 pp. 109-116.
- [21] T. Kwok and A. Mohindra, "Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications," *Proc. on Service-Oriented Computing*, 2008, pp. 633-648.
- [22] D. Westermann and C. Momm, "Using software performance curves for dependable and cost-efficient service hosting," *Proc. on Quality of Service-Oriented Software Systems (QUASOSS)*, 2010, pp. 1-6.
- [23] C. Osipov, G. Goldszmidt, M. Taylor, and I. Poddar, "Develop and deploy multi-tenant web-delivered solutions using IBM middleware: Part 2: Approaches for enabling multi-tenancy," In: *IBM's technical Library*, 2009.
- [24] C. Momm and R. Krebs, "A qualitative discussion of different approaches for implementing multi-tenant SaaS offerings," *Proc. Software Engineering* 2011, pp. 139-150.
- [25] M. Armbrust, et al., "A view of cloud computing," *Communications of the ACM*, 53(4), April 2010, pp. 50-58.
- [26] X. Cheng, Y. Shi, and Q. Li, "A multi-tenant oriented performance monitoring, detecting and scheduling architecture based on SLA," *Proc. on Joint Conferences on Pervasive Computing (JCPC)* 2009, pp. 599-604.
- [27] C. Bezemer and A. Zaidman, "Multi-tenant SaaS apps: Maintenance dream or nightmare," In: *Technical Report of Delft Uni. of Technology, TUD-SERG-2010-031*, 2010.
- [28] C. Bezemer, A. Zaidman, B. Platzbeecke, T. Hurkmans, and A. Hart, "Enabling multitenancy: An industrial experience report," In: *Technical Report of Delft Uni. of Technology, TUD-SERG-2010-030*, 2010.

Community Clouds

A centralized approach

Claudio Giovanoli, Stella Gatzu Grivas
 Institute for Information Systems
 University of Applied Science Northwestern Switzerland
 Olten, Switzerland
 {claudio.giovanoli, stella.gatzuigrivas}@fhnw.ch

Abstract— Community cloud is one of the rising ideas in the area of cloud computing. Many companies do not move into the cloud, as they need tailored solutions to ensure industry specific security and regulatory requirements. A community cloud can perfectly fulfill this requirement and costs can be spread among several organizations. Providing a community cloud involves aspects like security, privacy, identification and access management that includes lot of organization. This prevents providers and users to build a community cloud despite its advantages. However, until now it is not as widely spread as other deployment models like public or private clouds. One reason is that providing a community cloud needs a lot of organizational effort and communication. Additionally no standard concept for doing this is elaborated so far. Some providers are offering community clouds or certain organizations build one. Nevertheless, each community cloud underlies a different approach. This paper discusses the federated and brokered approaches. Additionally, a centralized approach on how a community cloud can be built will be introduced.

Keywords-Cloud Computing; Community Clouds; Service Market; Brokering Service

I. INTRODUCTION

Cloud computing has become a significant technology trend and provides new possibilities and advantages. It has open new opportunities to businesses on how to improve the usage, efficiency and reduce spending of their IT systems. According to NIST (National Institute of Standards and Technology of the US) several service and deployment models are proposed [1]. As a deployment model beyond private and public clouds, the concept of a community cloud is proposed. Community clouds are a union of private clouds, which are tailored to a specific vertical industry, such as government, healthcare or finance, offering a range of services including infrastructure, platform or software. Often, organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations) need to fulfill specific security and regulatory requirements [1].

The use of community clouds is not widespread yet, but there is definitely interest. Gartner shows with its Hype Cycle for Cloud Computing that Community Clouds are in its advent. Nevertheless, Gartner sees a high potential for the topic within the upcoming two to five years [2].

Figure 1. Hype Cycle for Cloud Computing, 2012

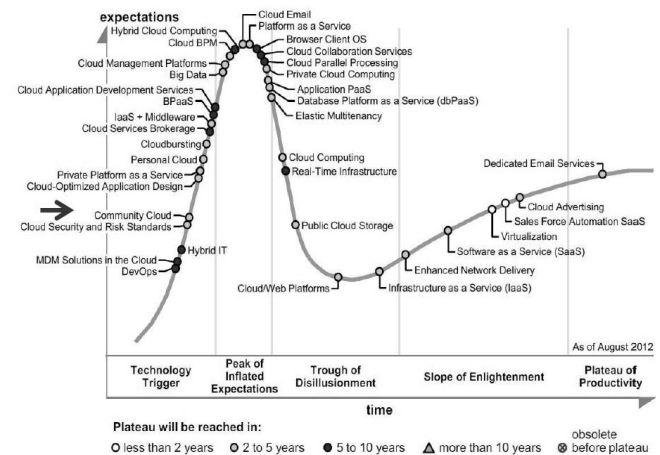


Figure 1. Gartner's Hype Cycle for Cloud Computing 2012 [2]

But what are the reasons community clouds are not widely used? Community cloud is a way of congregating users under an umbrella of services. Some businesses may hesitate to share common resources with competitors. A first obstacle on the way to a community cloud is to identify the appropriate community and to convince possible community mates to cooperate with. A second big drawback of this deployment model is to define the management, roles and responsibilities within all stakeholders. For interested customers, these additional efforts can be very discouraging to use such a shared cloud environment.

Despite the fact that community clouds are not yet established, there are some examples in the market. In particular, the industries of health, finance and government are early adopters of community clouds rollouts.

Most advantages of a community cloud are covered in the general benefits of cloud computing such as cost reduction and the shift from capital expenditures to operational expenditures. Nevertheless, a community cloud offers special advantages compared to other deployment models [3], [4]:

- Secure, private multi-tenant cloud computing satisfies demanding requirements of the organizations
- Flexible solutions to differing market needs
- Matching market fluctuations in demand

- Application or sensitive data can remain in the community network
- Less management than a private cloud
- Cost reduction by eliminating owned infrastructure and software licenses needs
- More efficient and potentially lower cost than existing systems and less cost than building an own private cloud or data center.

Several studies are showing that most of customers' concerns regarding clouds are compliance related issues. For example, a study published in early 2011 by KPMG [5] explains that companies are currently facing most often legal challenges. So, there are issues like security, uncertainty about the future control of their own data and to meet legal compliance, which hint potential users.

Thus, an important benefit of a community cloud is to address compliance requirements for specific groups like similar industries and to offer appropriate solutions to its concerns.

The aim of this paper is to discuss the different management models (federated, brokered and centralized) for such community clouds. After the introduction, challenges of community clouds will be discussed. The third section gives a short overview on current management models and introduces a centralized model. Thus, in the fourth section, a high level architecture of a centralized system will be suggested. The subsequent parts, sections V – IX, are explaining the different layers of the introduced models, from infoplace, to quality gate, over a brokering service, the cloud service management to finally the concept of a service market. The last section concludes the paper and gives a short overview on future work and next steps.

II. CHALLENGES OF COMMUNITY CLOUDS

Besides the introduced drivers, we see some key challenges within the idea of community clouds, which need to be considered before building up such kind of cloud.

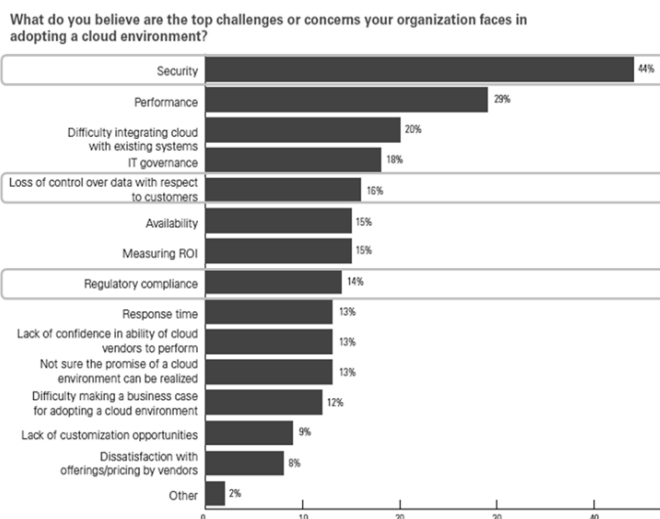


Figure 2. Customers' concerns entering a Cloud [5]

1) Organizational Structures and management models for building community clouds

By building up a community cloud, different stakeholders are involved. Thus, when considering a community cloud, at least as a special form of private cloud, two roles can be identified: the service providers and the service users.

Going one step further, a community deployment model can consist of several users and several providers, offering different services with some times similar functionality. This is mandatory to be able to prevent the well-known vendor lock-in effect [6]. However, this requires appropriate organizational structures and management models to avoid the loss of advantages. Several models will be discussed in the upcoming section.

2) Communications

Already before planning a community cloud, communications between the different stakeholders are crucial. Customers need to understand the advantages and risks of cloud services within such a closed environment. But, also, provider(s) have to understand the specific requirements each community has.

Even at an earlier stage, while thinking about the idea to build up a community cloud, an appropriate community has to be identified. Communications have to be initialized with first community members (clients and users). Thus, we suggest that the first step for building a community cloud is the establishment of an appropriate community. This community should create the business case, set the rules and organization form, and choose other members and providers.

But, communications between customers and providers, like announcements of common SLA adjustments, are playing also a key role during the cloud operations. To ensure a good cooperation between the community network, rules and responsibilities have clearly to be defined and announced to all stakeholders.

3) Ease of use

While establishing the cloud environment, not only security, efficiency and compliance issues have to be considered. As cloud promises fast and on-demand provision of IT services, customers have to decide easily, which services they need and want to use. Also ordering and service termination processes have to be allocated in such a way that users care able to access and execute services as easy as possible.

III. MANAGEMENT MODELS FOR COMMUNITY CLOUDS

Management models for community clouds follow either a federated or a brokered approach [7]. In a federated approach all institutions (members of the community cloud) share their own resources, whereas in a brokered approach sharing of the resources takes place through a third party, the

so-called broker. This means, the broker procures the resources (services) to the community cloud members.

Today the implementation of a federated management model most often faces challenges mainly due to two main reasons. First, it is difficult to tackle liability issues like the legal impact of a service outage or responsibilities. Secondly, it is hard to provide cost transparency. Questions about the responsibility of paying support, maintenance and operational costs are arising. However, such a federated model comes with its benefits. The vendor lock-in issue does not exist, risks are distributed and costs are reduced. Furthermore, it offers full control of the community members who can share best practice and their industry specific services.

Today, the brokered model is the usually deployed one, when implementing a community cloud. In the brokered model, institutions share provider resources through a so-called broker. The broker acts as an intermediary and should provide expert advice to the community. It takes care of trust establishment and contract settlement. The institutions only have one party to trust and one contract to sign. The brokers can also handle disputes in the cloud [8]. This model is transparent in terms of operation and accountability, awareness raising, guidance on expectations regarding the use of the community cloud, levels of security, and meeting legal obligation (compliance). Operations can be spread across multiple cloud providers whereby continuity is given. The broker is fully responsible for security issues, it forces specific security and regulatory requirements. Participating institutions do not need to test whether a cloud provider is effectively mitigating risks. It is the role of the broker to assure such aspects for the community. At last, a broker can provide value-adding services like federated identity management or resource federation [7].

Contrariwise, a third model, the centralized one, has only one IaaS provider and one broker. The model foresees an IaaS provider as the leading party, which is responsible for (i) establishing the infrastructure platform of the cloud including services such as

- Computational Power
- Networking
- Storage
- Virtualization
- etc.

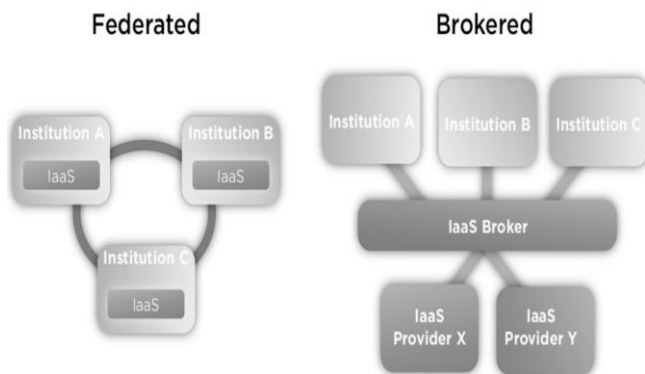


Figure 3. Two main models for Community Clouds [7]

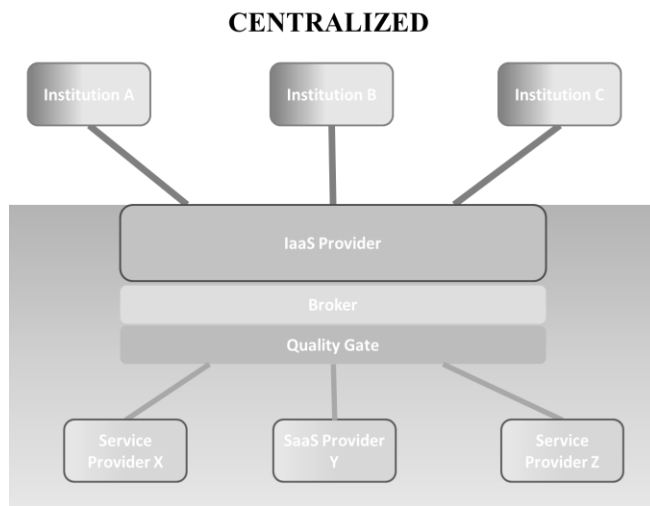


Figure 4. A centralized approach

and (ii) for procuring PaaS and SaaS providers. The broker is responsible for expert advice and acts as an intermediary. In this model the broker has different duties, as described in the brokered model. Operations are spread across multiple service providers.

With regards to performance aspects, it has to be considered that a centralized model is highly dependent of a sole IaaS provider. Thus, we are expecting that in a worst case scenario, the centralized approach can be less performing compared to the other models. While the federated and brokered models are offering the opportunity to change the provider in such a case, the centralized one does not.

IV. ARCHITECTURE OF A CENTRALIZED APPROACH

Following a centralized approach opens a wide range for the establishment of architecture of a community cloud. Besides choosing an IaaS provider for the leading provider role, we recommend five layers within the community cloud architecture. These layers support customers and service providers through different stages of the service lifecycle.

The cooperation between each of the five components establishes a trustworthy usage of the different cloud services within the community. It offers flexibility to the community, with regards to organization and communication aspects.

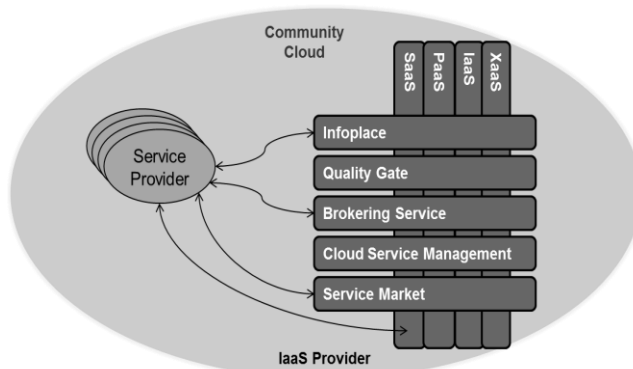


Figure 5. Proposed components of a centralized community model

Additionally, it ensures that users can easily get the needed services with a low commitment to time, money and management resources. Each of these five layers from infoplace, Quality Gate, to Brokering Service, Cloud Service Management and Service Market will be introduced within the following sections.

V. INFOPLACE

A so-called infoplace builds the entry point for the community members and can clearly support the establishment of the communication between the members of a community cloud.

Whereas potential cloud customers are facing several challenges and open questions like

- Which services are appropriate to obtain from the cloud environment?
- Do Cloud Services fit to the IT of my company?
- What are the advantages and benefits, given through Cloud services?
- Is my company prepared for the cloud?

the infoplace offers assistance to the customer, e.g. readiness assessments to evaluate potential technical or organizational gaps within the company.

An additional advantage of the infoplace is the use case repository. The use case repository enables to store the collected cloud use cases within the community. It follows a developed framework, which defines different areas of interest inside such a use case. Following this scheme also establishes that use cases can be compared on the different topics like the service model but also on technical and management issues.

The use cases should be (i) a viable source for the user to see how other have compete their cloud projects and (ii) to support the user by identifying different workloads / process areas, which are predestined to run in a cloud.

For realizing these infoplace requirements the University of Applied Science Northwestern Switzerland is building a platform for guiding users through the cloud life cycle. For this need, they introduce a project named CLiCk (Cloud Life Cycle).

The vision of the CLiCk-Infoplace is the provision of self-services and supportive information, which can be accessed on an appropriate platform through the accordant enterprises [9].

VI. QUALITY GATE

All actors within the community cloud need to fulfill certain criteria. Not everyone is allowed to use the services, not every application will be offered in the cloud, and not all service providers fulfill the compliance requirements of the users. Therefore a quality gate service has to be provided. The Quality Gate describes an independent service within the community cloud. Its main purpose is to assess the general and industry specific criteria, which have to be followed by all stakeholders in a community cloud (users and providers). General criteria could be for example the ability of auditing. Industry specific criteria can dictate e.g.

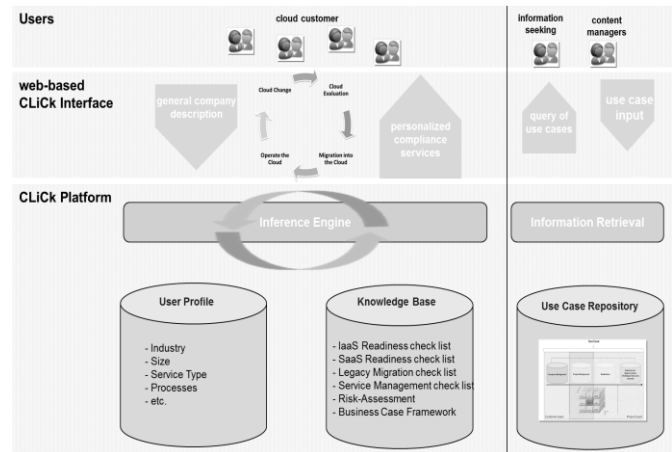


Figure 6. The CLiCk Infoplace [9]

form, location and minimal duration of storage for digital records [10]. The quality gate includes:

- *Quality of Service Providers:* The quality of service providers needs to be assured because the community has to follow certain legal restrictions. E.g. the service provider needs to prove, that their company obeys to according laws or that they handle sensitive data with needed concern. Another aspect is the sustainability of the service provider, it is important for the success that the company will exist further. To ensure the quality of service providers certain standards need to be fulfilled. These standards can be ISO standards or other certifications.
- *Quality of Services:* The offered services need to have a certain quality. For example a finance application for financial administration has the restriction that it needs to be certified by the government. The quality assurance service should elaborate a list of criteria, which an application needs to fulfill. This list of criteria differs depending on the type of application. For example an application for wage payment has other criteria than an application for drawing mind maps.
- *Quality of Customers:* The third category of the quality assurance process concerns the users. Goal is to assess candidates for community membership. While establishing the introduced community it is important to setup the conditions for entering the community and using the services out of the cloud.

To ensure a high level of trust, the role of assessing the introduced quality criteria within the gate should be executed by an independent actor.

Criteria should be defined and collected continuously through a consortium of community members and project independent advisories.

VII. BROKERING SERVICE

Cloud brokering is not yet finally defined. Several opinions about what, who and how cloud brokering services should be able to fulfill exist. One perspective of brokering

has been explained in Section III within the brokered community cloud model. A more generic view, applicable for most cloud deployment models is given by Buyya [11]. It can be understood as a part of a global marketplace, where service providers and consumers join to find suitable match for each other. It provides various services to its customers such as resource discovery, meta-scheduler, reservation service, queuing service, accounting and pricing services [11]. Gartner explains cloud brokering as a “cloud services brokerage (CSB) is a service provider that plays an intermediary role in cloud computing” [12]. They see three different types of brokerage scenarios: aggregation, intermediation and arbitration [13].

Within the introduced community cloud approach, brokering services are understood as a provision and convey of the available services. The cloud broker has knowledge about the used services by each customer and the available services in the market. If a customer needs a new service, e.g. additional software, or an altered quantity of a service, the brokering service executes the new requirement immediately and orders it from the service market. Like in the brokered community model, the broker can also take a leading role for contracting.

VIII. CLOUD SERVICE MANAGEMENT

The fourth layer deals with service management aspects. The cloud service management denotes the implementation and management of additional services that meet the needs of the community members and includes facilities like:

- a) *Installation and Configuration:* executes administrative tasks that occur primarily in the introduction, the transition or the early use of cloud computing. It includes, for example, adjustments of organizational processes and structures, and descriptions of specific cloud projects or complex issues and how these can be overcome.
- b) *Resource Management:* This topic deals with the distribution of the (hardware) resources (e.g. based on best practices) - also with regard to high scalability and flexibility. It also includes interoperability aspects, so far by defining standards for higher compatibility between different services is provided.
- c) *Service Monitoring and Reporting:* offers automated services to control if agreed parameters like availability, speed and quality of provided services are accordantly to Service Level Agreements. This includes also a customer service for reporting current figures about usage, costs and delivered performance of services.

A cloud service management can be provided through an appropriate mix of people, process and information technology.

IX. SERVICE MARKET

The final element of a community cloud is the provision of the individual cloud services independent from IaaS, PaaS or SaaS.

On a so called service market, customers are able to compare, select, buy and review applications. Users choose out of a set of qualified cloud services. Any offered service is tested and approved in the quality gate through an independent consortium. Usage of the service market has to be as simple and easy as other well know application stores like, e.g., Apples iTunes or Google's Play.

While the infoplace supports customers to find the appropriate service, the service market leads the client to the final purchase of a service. A service market model is potentially valuable for any sort of IT product or service that is sufficiently industrializes and packaged in order to be consumed by a non-expert end-user. Such kind of application store can become the marketplace to access cloud, commercial software products, skills, as well as to finally succeed reusing and exchanging software across different companies. The goal of a service market is to make IT offerings transparent, unambiguous and comparable. Furthermore, reduced procurement times, increased user satisfaction, and reduced costs should be the outcomes of such a marketplace.

The idea behind a marketplace is not explicitly bound to a community cloud. But according to Buyya et al. [14] it is predestinated to be applied for a specific industry respectively community similar to the logistics clouds.

X. CONCLUSION AND FUTURE WORKS

Cloud Computing is still in its advent, and the number of interested business and depending business models is increasing. But many institutions, whether potential customers or consultancies are hesitating to consume IT services in a cloud approach. As shown, many issues concern security and compliance areas. IT has not yet been successful in getting these issues out of the way to the cloud.

To decrease such security and compliance issues, a community cloud is one approach to face these challenges and to use the advantages of the cloud approach like reducing costs, faster time to market at the same time.

While a community cloud can improve the security and compliance issues, it also brings additional challenges. Compared to other deployment models, organizational and communication efforts within a community cloud are increasing as a whole. As other deployment models, like public and private, commonly are describing a business to business (customer to provider) dependency. A community approach opens relations to the entire community (customers and providers). To ensure the success and proper management of a community cloud the stated increased organizational and communication efforts is essential.

In a good working cooperation the additional effort can be spread over all community members and will not cause more effort for the single instance than using other deployment models.

For the process of establishing a community cloud we propose as a first step to identify and coin a proper community with specific similar concerns. This community shall define the requirements, goals, organizational and management approaches of the cloud. As concerns regarding compliance most often are related with the cloud data center location, the introduced centralized approach, including one leading IaaS provider, should enlighten the given regulatory requirements and ensure that Platform- and Software providers are in line with the community concerns too.

The introduced layered approach of the centralized solution supports the community to establish a vendor independent (excluding the infrastructure provider), flexible and high quality shared IT environment, where advantages of cloud computing can be gained. Thus, community members are able to focus on core businesses instead of handling with IT issues.

Whether a brokered, federated or centralized approach, community clouds in general are offering a considerable option for businesses with sensible IT issues. We see the community deployment model as a serious suggestion for future IT services in areas with special security and compliance needs.

As the introduced centralized approach is only a first high level architecture, the authors are currently identifying different domains to initiate a first pilot. Goal is to find few partners for establishing a pilot of a centralized community cloud. First talks are held with partners from energy, health and public industries. Based on their feedback funding and further partners in the industries for a pilot project will now be identified. As a first step of such a pilot program, the domain specific requirements will be assessed to setup the base for the different layers.

REFERENCES

- [1] Mell, P. and Grance, T., "The NIST Definition of Cloud Computing" Recommendations of the National Institute of Standards and Technology, NIST Special Publication 2011.
- [2] Gartner Inc. , "Hype Cycle Cloud Computing 2012", Gartner Incorporate, Stamford: 2012.
- [3] Nussbaum, C., "Dissecting the Cloud IV – Community Clouds?," 2012, Retrieved from <http://www.atomrain.com/it/technology/dissecting-cloud-iv-community-clouds> 07.04.2013.
- [4] Butler, B. , "Will community cloud services be the next big thing?" Digital Publication, Computerworld.uk, 2012, Retrived from <http://www.computerworlduk.com/in-depth/cloud-computing/3341808/will-community-cloud-services-be-the-next-big-thing/> 07.04.2013.
- [5] KPMG, "Clarity in the Cloud: A Global study of the Business adoption of Cloud", KPMG International, Switzerland: 2012
- [6] Armbrust et al., " A View of Cloud Computing: Clearing the clouds away from the true potential and obstacles posed by this computing capability" , ACM Communications, University of Berkeley: 2010
- [7] Teunissen, H. et al., "Community Clouds; Shared Infrastructre as a Service", University of Utrecht, Cloud Seminar, 2011, Retrieved from <http://de.slideshare.net/SNKN-CloudComputing/110616-community-cloudsseminar-cloud-computing> 03.04.2013.
- [8] Koelliker, C., "Community Clouds in Domains with Specific Needs: The Example of Swiss Government Cloud", Master Thesis, University of Applied Arts and Sciences Northwestern Switzerland, Januray 2013, unpublished
- [9] Giovanoli, C. and Gatzju Grivas, S." Building a Knowledge Base for Guiding Users through the Cloud Life Cycle", CLOSER Conference Paper, 2013, to be published.
- [10] Duranti, L., Rogers, C., "Trust in digital records: An increasingly cloudy legal area", Elsevier, University of British Columbia, Canada: 2012.
- [11] R. Buyya et al., "Cloudbus Toolkit for Market-Oriented Cloud Computing", Manjrasoft Pty Ltd, Melbourne, Australia, 2012.
- [12] Gartner Inc. , "Gartner Outlines Five Cloud Computing Trends That Will Affect Cloud Strategy Through 2015", Gartner Incorporate, Stamford, 2012
- [13] Gartner Inc. , "Three types of Cloud Brokerage will enhance Cloud Services", Gartner Incorporate, Stamford: 2009.
- [14] Buyya, R., Pandey, S. and Vecchiola, C., "Cloudbus toolkit for market-oriented cloud computing." Cloud Computing, pp. 24–44, 2009, Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-10665-1_4 03.04.2013.

Using MapReduce to Speed Up Storm Identification from Big Raw Rainfall Data

Kulsawasd Jitkajornwanich*, Upa Gupta*, Ramez Elmasri*, Leonidas Fegaras*, and John McEnergy†

*Computer Science and Engineering Department
University of Texas at Arlington
{kulsawasd.jitkajornwanich, upa.gupta}@mavs.uta.edu,
{elmasri, fegaras}@cse.uta.edu

†Department of Civil Engineering
University of Texas at Arlington
mcenery@uta.edu

Abstract— This paper describes an efficient MapReduce algorithm for converting raw rainfall data into meaningful storm information, which can then be easily analyzed and mined. Our previous work proposed a method to identify relevant storm characteristics from raw rainfall data. The original storm identification system takes too long to produce the summarized storm characteristics, because: (1) the raw rainfall data, which is considered as big data, is stored in a traditional relational database based on CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science, Inc.) ODM (Observations Data Model), which leads to substantial disk I/O; (2) the storm identification algorithm is based on recursion and regular depth-first-search (DFS), which leads to multiple retrievals for parts of the data. In this paper, we obtain a substantial improvement in performance by utilizing MapReduce. We also utilize the original raw rainfall data text files instead of using the data in the relational database. In our experiments, the performance of the new storm identification system is significantly improved compared to the previous one. With this new system, it will dramatically benefit hydrologists in helping them performing rainfall-related analysis (both location-specific and storm-specific) such as flood prediction using our identified storms.

Keywords-storm analysis; rainfall; big data; MapReduce; distributed computing; CUAHSI

I. INTRODUCTION

This paper describes an efficient MapReduce algorithm for converting raw rainfall data into meaningful storm information, which can then be easily analyzed and mined. Our previous work [1] proposed a method to identify relevant storm characteristics from raw rainfall data. The original storm identification system takes too long to produce the summarized storm characteristics, because: (1) the raw rainfall data, which is considered as big data [7][8], is stored in a traditional relational database based on CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science, Inc.) ODM (Observations Data Model) [17][18][9], which leads to substantial disk I/O; (2) the storm identification algorithm is based on recursion and regular depth-first-search (DFS), which leads to multiple retrievals for parts of the data. In this paper, we obtain a substantial improvement in performance by utilizing MapReduce. We also utilize the original raw rainfall data

text files instead of using the data in the relational database. In our experiments, the performance of the new storm identification system is significantly improved compared to the previous one. With this new system, it will dramatically benefit hydrologists in helping them performing rainfall-related analysis (both location-specific and storm-specific) such as flood prediction using our identified storms.

Our raw rainfall data, called MPE (Multi-sensor Precipitation Estimates) [19][20][21], is estimated by using combination of radars and physical rain gauges (multi-sensors) and is retrieved from National Weather Service (NWS) - West Gulf River Forecast Center (WGRFC) [19]. The raw data is supplied as hourly text files using the HRAP (Hydrologic Rainfall Analysis Project) standard grid coordinate system [20][17]. The raw rainfall data is converted into a relational database in order to follow the CUAHSI ODM standard, which was required for the HydroDesktop system [22] that allows hydrology users to search the rainfall data.

Our previous storm identification system used the relational data as input. Due to the relational database I/O overhead, and the tremendous amount of data, system performance was too slow. The data covers 17 years (1996 - 2012) of historical hourly precipitation, which is translated to 8.004123763 billion records in the database. We receive the rainfall data on an hourly basis covering 4 states (Texas, Colorado, New Mexico, and Louisiana) (mainly Texas) and part of Mexico (see Figure 1) covering 69,830 site locations. The number of records inserted per hour, day, month, and year is 69,830, 1,675,920, 50,277,600, and 603,331,200, respectively.

In this paper, we develop more efficient storm identification algorithms using the original text file formats, and the MapReduce framework to parallelize the processing.



Figure 1. Coverage of WGRFC observations [19][21]

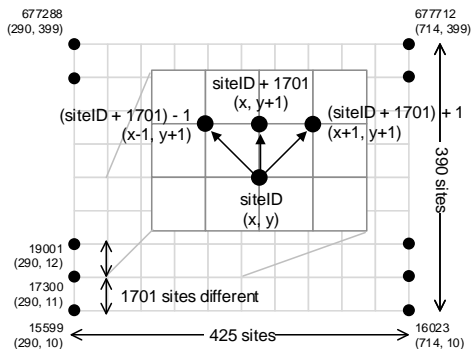


Figure 2. Relationships among neighboring sites

MapReduce is a programming paradigm developed by Google in 2004 [5] and now becoming a new standard for distributed computing. Our previous storm identification algorithms, based on recursion and depth-first-search, traverse the data exhaustively without taking advantage of the known regular grid structure of the raw rainfall data. We greatly improved the performance by using the original raw rainfall data and applying MapReduce to every component of the storm identification process. However, only local storm and hourly storm identifications (also known as event separator and sub storm identification in [1]) are discussed in this paper. The details of each component are described in Sections IV and V, which are MapReduce for local storm and hourly storm identifications, respectively. First, we review the storm identification concepts from [1] in Sections II and III. Section VI discusses experimental results. Related work is discussed in Section VII.

II. INPUT DATA STRUCTURE

The raw rainfall data is supplied as text files. The file name indicates a particular date and time (hourly, e.g., 2011041323_2011041400), and includes the precipitation data for all sites during that hour. Each row consists of row number, site id, and precipitation value (inches). The data is ordered by site id in a row major order from west to east and south to north. Sites are in an HRAP regular grid and four kilometers apart to north, south, east, and west. Each row in the grid has 425 sites and each column has 390 sites as shown in Figure 2. Because of the systematic grid structure, given any site, we can determine the neighboring sites by using the formulas in Figure 2. Moreover, given any site id, we can determine its HRAP local X and Y coordinates, and vice versa using the following equations: (1) and (2).

$$x = 290 + ((siteID - 15599) \text{ mod } 1701) \quad (1)$$

$$y = 10 + ((siteID - 15599) \text{ div } 1701) \quad (2)$$

III. STORM-RELATED CONCEPTS

In this section, we review some key components of the previous work [1] that are needed for this paper. Two main components are: (1) storm formalization and (2) storm identification process.

A. Storm Formalization

We formalize storms into three different categories (local storms, hourly storms, and overall storms), the goal of which is to develop a storm identification process and storm characteristics analysis. The following is some terminology needed for the storm formalization.

- *storm duration*: the time length over which precipitation occurs (hours) [23].
- *storm coverage*: the number of sites covered by a storm.
- *storm area*: the total area of a storm.

1) Local Storms

Generally speaking, *local storm* is a site-specific storm, which considers each site location independently when analyzing a storm. An example of local storms is the set of storms that occurred at site location 586987 last month. Local storm is one type of storm, which was researched by most hydrologists [11][12][13][14]. This may be due to the traditional way of storm analysis, which does the analysis primarily based on how raw rainfall data are collected and stored without applying distributed computing technology.

Formally, a local storm is a set of time points and associated rainfall data at a particular spatial site. Two distinct local storms are separated by at least h consecutive time points with zero precipitation, where h is called the *inter-event time* [11][12][16]. In this paper, inter-event time (h) is set to 6 hours as suggested in [11][12]. Several consecutive time points with zero precipitation within a local storm, however, are allowed as long as it is less than h time points. For any local storm, there will not be a subsequence of h or more consecutive zeroes in the series. Figure 3 shows some examples of local storms at site id, 586987. Some storm characteristics for this storm type include:

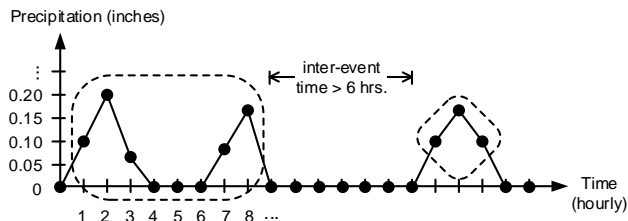


Figure 3. Examples of local storms at site id, 586987

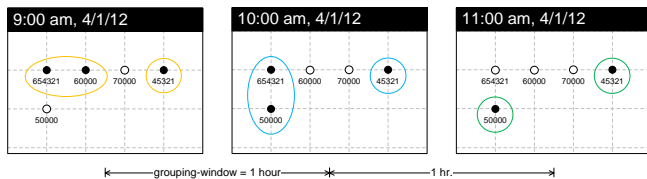


Figure 4. Examples of hourly storms at different hours on 4/1/12

- *storm depth*: the amount of precipitation occurring throughout the storm duration at a particular site [23].
- *storm intensity*: the storm depth divided by the storm duration (inches per hour) [23].

2) Hourly Storms

Informally, *hourly storm* is a time-specific storm, which has an orthogonal concept to local storm. It considers each hour independently when analyzing a storm. An example of hourly storms is the set of storms that occurred between 9:00 am and 10:00 am today. Hourly storm considers a specific time point (an hour) instead of considering a particular site location. In other words, local storm fixes one site and covers its data over many time points, whereas hourly storm fixes a time point and covers its data over many adjacent sites. Figure 4 shows some examples of hourly storms at different hours on April 1, 2012.

Formally, an hourly storm is a set of adjacent sites of local storms at a particular hour. However, a more relaxing definition can also be applied as discussed in our previous work [1] as *space-tolerance*. The concept of space-tolerance is to allow indirect neighboring sites to be considered as part of the same hourly storm. In this paper, we use the original definition of hourly storm, which takes into account only direct neighboring sites when identifying hourly storms. The following are storm characteristics that are applicable for this type of storm:

- *storm sites total*: the total amount of precipitation occurring at a particular hour for the sites of an hourly storm.
- *storm average*: the average precipitation (per site) for an hourly storm.

3) Overall Storms

Unlike local storm and hourly storm that consider either a site location or time (an hour) independently, it considers both location and time together when analyzing a storm. So, the result is the capture of storm as a whole, called *overall storm*, which can capture storm movement and other storm characteristics that could not be found in most hydrology papers [11][12][13][14]. An overall storm is built upon hourly storms. Some examples of overall storms are shown in Figure 5.

Formally, an overall storm is a set of hourly storms that meet two requirements: (1) *grouping-window* and (2) *spatial-window*. Grouping-window is the maximum time

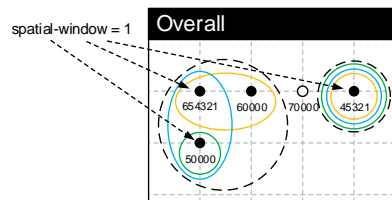


Figure 5. Examples of overall storms

interval within which hourly storms will be considered to be part of the same storm whereas *spatial-window* is the minimum number of common site(s) shared between two hourly storms. This formalization allows hourly storms that go to the same direction be considered together. However, in a rare situation, it is also possible that two different paths of hourly storms with different origins and/or destinations could end up being part of the same overall storm. In such case, the final path of the overall storm will be averaged based on those two paths. In this work, *grouping-window* and *spatial-window* are set to 1 hour and 1 site, respectively. Overall storm characteristics include:

- *storm overall depth*: the total amount of precipitation occurring throughout the storm duration across the hourly storms.
- *storm overall intensity*: the storm overall depth divided by the storm duration (inches per hour).
- *storm overall average*: the average precipitation (per site) for an overall storm.

B. Storm Identification Process

The main goal of our storm identification system is to analyze storms as a whole. Since a storm can start at one place and stop at another, we slice the whole storm into several pieces by hour. We then assemble each slice back together into the original overall storm. Each slice of storm is, in fact, an hourly storm. Figure 6 shows architecture of our previous storm identification system.

The storm identification process can be divided into three main components: (1) event separator (to identify local storms), (2) sub storm identification (to identify hourly storms), and (3) main storm identification (to identify overall storms). The architecture of our new storm identification system is shown in Figure 7.

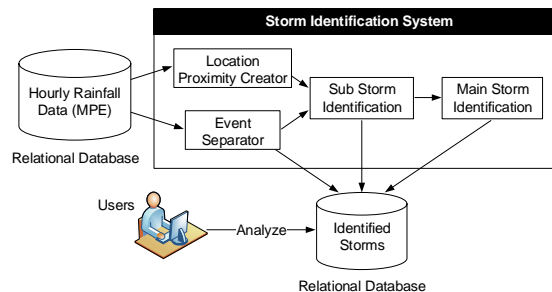


Figure 6. Architecture of previous storm identification system

IV. MAPREDUCE FOR LOCAL STORM IDENTIFICATION

The local storm identification identifies the storms at a particular site and specifies each storm duration (in hours) at that site. The previous implementation of local storm identification required the selection of data from the relational database and then sorting them. The computation is done based on the selected sorted data and the result is inserted back to the database. The selection, sorting, and insertion required substantial execution time, making it impractical to analyze the whole raw data.

Our new algorithms utilize MapReduce, and use the rainfall data text files as input. Each raw rainfall file contains the precipitation value of all the sites for a particular hour and hence, for the analysis of local storm, we need to group all the precipitation values by site and order them by time. Once all the values for a site are grouped together and ordered, then we can find all the local storms that occurred at that site. Thus, the local storm analysis contains two steps: (1) grouping precipitation values by site and ordering them by time and (2) finding the local storms for a site from the grouped values. In the MapReduce framework, there are three main phases: (1) map phase, (2) sorting and shuffling phase, and (3) reduce phase. The first two phases of MapReduce are used to perform the first step of our local storm identification and the reduce phase is used to find the local storms at the particular site.

Algorithm 1. Local Storm Identification

```

Input:
- Text file-format rainfall data
Output:
- Local storms data in text file format
1: class MAPPER
2: function MAP(key object, value line)
3:     key <-- (line.siteId, line.time)
4:     value <-- (line.precipValue, line.time)
5:     Emit(key, value)
6: class REDUCER
7: function REDUCE(key siteld, [val1, val2, ...])
8:     timeList, precipRec <-- null //timeList.size = inter-event + 2
9:     interEventTime <-- 0, lsId <-- 1
10:    timeList.Add(firstNonZeroPrecip.GetTime())
11:    precipRec.Add(firstNonZeroPrecip.GetPrecipValue())
12:    for all val ∈ values [val1, val2, ...] do
13:        precipRec.Add(val.GetPrecipValue())
14:        if (val.GetPrecipValue() = 0) then
15:            timeList.Add(val.GetTime())
16:            interEventTime++
17:        else
18:            tempTime <-- timeList[0], Clear(timeList)
19:            timeList.Add(tempTime; val.GetTime())
20:        end if
21:        if interEventTime ≥ 6 then
22:            initialTime, finalTime <-- timeList[0], timeList[1]
23:            value.Set(initialTime, finalTime, precipRec)
24:            Emit(siteld, lsId, value)
25:            Clear(timeList; precipRec), lsId++
26:        end if
27:    end for

```

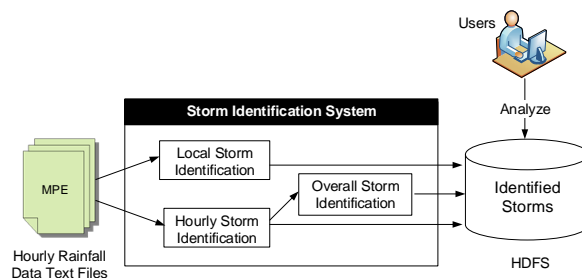


Figure 7. Architecture of current storm identification system

The pseudo code for the implementation for local storm analysis in the MapReduce framework is shown in Algorithm 1. Each of the map tasks takes one raw rainfall file and processes it line by line emitting the site and time together as the key and time and precipitation value together as the value. We take advantage of the key-comparator class and grouping-comparator class of MapReduce to group the data on the basis of site id and then sort them by time. The reducer gets a site id as a key and list of precipitation values sorted by time. This list is processed sequentially to identify all the local storms at that particular site.

V. MAPREDUCE FOR HOURLY STORM IDENTIFICATION

The second main component is hourly storm identification, the goal of which is to identify hourly storms for each particular hour.

In the previous approach [1], we assume that any non-zero precipitation site can be part of the hourly storm, meaning it can start at one site and stop at a very farther site as long as there are some connections among them. As a result, we implemented DFS to keep track of every possible site and perform site node revisiting when needed. This, however, led to a high time complexity problem. In addition, the algorithm interacts with the data in the relational database, which causes a large overhead.

In the new approach, the program is designed specifically to take full advantage of the original raw rainfall data text file structure. Since the grid (HRAP) is known and we know exactly which site is a neighbor of which, only those candidate neighboring sites need to be checked. Unlike previous approach which uses DFS to keep track of node, we use linked lists and append them together as we scan when necessary. Moreover, since the data in each text file is stored in row major order, we scan each grid row once. An overview of the hourly storm identification process is shown in Figure 8.

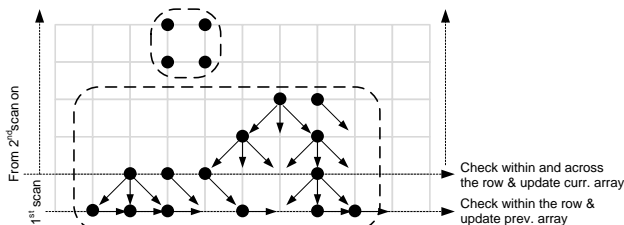


Figure 8. Overview of hourly storm identification process

Algorithm 2. Hourly Storm Identification

```

Input:
- Text file-format rainfall data
Output:
- Hourly storms data in text file format
1: class MAPPER
2: function SETUP()
3:   prev.InitializeArray(), curr.InitializeArray()
4:   hourlyStorms.InitializeArrayOfLinkedList()
5:   id <-- 0
6: function MAP[key object, value r)
7:   if r ∈ first bottom grid sites then
8:     if r.precip = 0 then
9:       prev[r.site].hsId <-- 0 //no hourly storm
10:    else
11:      if r.site = first site or r.leftNeighborPrecip = 0 then
12:        prev[r.site].hsId <-- id++
13:        temp <-- CreateLinkedList(r.site)
14:        hourlyStorms.AddLinkedList(temp)
15:      else
16:        prev[r.site].hsId <-- id
17:        hourlyStorms.GetLinkList(id).Add(r.site)
18:      end if
19:    end if
20:  else if r ∈ next above grid sites then
21:    if r.precip ≠ 0 then
22:      if r.site = first site or r.leftNeighborPrecip = 0 then
23:        CheckPrevious(r, id, prev, curr, 1)
24:      else
25:        CheckPrevious(r, id, prev, curr, 0)
26:      end if
27:    else
28:      curr[r.site].hsId <-- 0 //no hourly storm
29:    end if
30:  else
31:    prev <-- curr
32:  end if
33: function CLOSE()
34:   Emit(hourlyStorms)
35: function CHECKPREVIOUS(r, id, prev, curr, flag)
36:   if flag = 1 then
37:     if hslds of all 3 neighbors of r in prev. array = 0 then
38:       curr[r.site].hsId <-- id++
39:       temp <-- CreateLinkedList(r.site)
40:       hourlyStorms.AddLinkedList(temp)
41:     else
42:       minId <-- MinHsld(r.all3Neighbors in prev. array)
43:       curr[r.site].hsId <-- minId
44:       hourlyStorms.GetLinkList(minId).Add(r.site)
45:       UpdateHsld(r.neighbors, minId)
46:       minId <-- 0 //reset minId
47:     end if
48:   else
49:     curr[r.site].hsId <-- id
50:     hourlyStorms.GetLinkList(id).Add(r.site)
51:   if hsld of r's southeast neighbor in prev ≠ id then
52:     UpdateHsld(r.southeastNeighbor, id)
53:   end if
54: end if

```

The program starts from the very bottom grid row to the top by calling map function for each line in the text file. It begins to identifying hourly storms as soon as it reads in the data in order to minimize the number of checking. The data are then kept in two arrays called *previous* and *current arrays*, which are two-dimensional arrays and contains site ids and hourly storm ids. The current array always does the identification based on the previous array. There are two main parts of the program. The first part (line: 7-19) is executed only once for the very bottom row in a grid whereas another part (line: 20-32) is executed for the rest. The first part identifies hourly storms within the same row whereas the other part identifies hourly storms within and across the rows simultaneously. At the end of each row scan, the hourly storms so far are identified and are kept in an array of linked lists called *hourly storms list*, in which index of array indicates hourly storm id and linked list contains a set of adjacent non-zero precipitation sites of the hourly storm. When reached the last row, the final hourly storms are produced and already kept in the hourly storms list.

Since the raw rainfall data files are independent from each other, in which each file records hourly precipitation for an individual hour, MapReduce can easily be applied. Each hourly file is sent to different mapper nodes for the identification of hourly storms. At the closing of mapper, all hourly storms identified within the hour will be written back to a disk. Currently, no reducer is needed because there is no need to group the data or sort them in any order. The raw files, by themselves, are already grouped and sorted by site id in a row-major order as mentioned in Section II. An algorithm for hourly storm identification is shown in Algorithm 2.

VI. EXPERIMENTAL RESULTS

In the previous approach, the experiment was performed on the rainfall dataset, resided in a relational database, using a single server. The server runs on Microsoft® Windows Server® 2008 Enterprise operating system with 2.83 GHz Intel® Xeon® quad-core processors, 20 GB of RAM, 500 GB of local disk, and 10 TB of external disk. In the new approach, the experiment was performed on the same dataset that is in the original text file format rather than relational format using a Hadoop® cluster [6] of 1 frontend server and 18 worker nodes. Each worker node contains 3.2 GHz Intel® Xeon® quad-core processors, 4 GB of RAM and 1.5 TB of local disk allocated to HDFS. The server has the same specification but with 3 TB of local disk. The cluster is set up by using ROCKS Cluster 6.3 OS and then installing Hadoop® 1.0.3 on every node.

Both local storm and hourly storm identifications are analyzed over 16 months of data. The data has 11,488 hours and is 10 GB in size. The raw files are in text format. Each file is for all sites during a single hour and is zipped into one gzip file. These files are fed into to the MapReduce job for the storm analysis. There are separate map tasks for each

of the files because each file is gunzipped into separate .gz files.

The comparison between the time taken by the previous implementations and the new MapReduce implementations is shown in Table I. Please also note that the processing time does not include the time taken to load the data into HDFS/SQL. In addition, each experiment was performed 10 times and an average processing time is calculated.

The experiments of the new approach give the same results for both local storms and hourly storms as the previous approach but is executed significantly faster. The new approach allows programs to be executed distributedly on multiple machines and hence the efficiency of the storm analysis is increased. For local storm (LS) identification, the time improved to 2.79 hours, compared to 53.44 hours in the previous approach. For hourly storms (HS), the MapReduce (MR) took 0.45 hours, compared to 6.78 hours in the previous method (DFS).

VII. RELATED WORK

There are two main parts of related work: (1) storm characteristics analysis and (2) MapReduce framework for spatial data computing.

A. Storm Characteristics Analysis

In most hydrology papers, most rainfall data analysis is either site-specific or region-specific and only few do storm analysis by integrating them across sites [10][11][12][13][14][15][16]. Asquith et al. [11][16][15] studies storm statistical characteristics by looking at the means of storm inter-event time, depth, and duration. In [10][12][14], Overeem and Asquith study storm characteristics through their DDF (depth-duration-frequency) properties. Lanning-Rush [13] studies storm characteristics by focusing on their extreme precipitation (EP) values. Within these, only a small amount of data and limited number of gauges were used. The storm analysis was conducted mainly based on how raw rainfall data is collected and stored, which is by location stored in different folders. This might be a reason why there are not many programs developed to process rainfall data across sites. Consequently, the flexibility in analyzing overall storm characteristics was lacking.

Our work, on the other hand, allows rainfall data to be analyzed in both location-specific (site-specific and region-specific) and storm-specific. Additionally, a much larger amount of data across a large number of gauges on HRAP standard grid coordinates can be analyzed. Our efficient algorithms were custom designed to take advantage of the format of the original raw rainfall data, as well as adopt renowned distributed computing technology, called MapReduce, to analyze storms in a storm-specific manner.

TABLE I. EXPERIMENTAL RESULTS

Regions / Number of Raw Data	Sites	Processing Time (in hours)			
		Previous work		Current work	
		LS (sec/site)	HS (DFS)	LS (sec/site)	HS (MR)
1. East Texas (48,953,130)	4,643	8.67 6.72s/site	1.44	↑ ----- 2.79 hours for all 10 regions (0.27seconds/site) ----- ↓	↑ ----- 0.45 hours for all 10 regions ----- ↓
2. Edwards Plateau (73,415,532)	6,962	8.72 4.51s/site	1.23		
3. High Plains (31,711,927)	3,008	4.50 5.39s/site	0.32		
4. Low Rolling Plains (24,965,521)	2,368	3.35 5.10s/site	0.28		
5. North Central (59,082,957)	5,604	8.66 5.56s/site	1.17		
6. South Central (31,102,334)	2,949	4.28 5.22s/site	0.67		
7. South Texas (31,949,386)	2,933	3.97 4.87s/site	0.48		
8. Lower Valley (5,324,898)	601	0.55 3.32s/site	0.07		
9. Trans-Pecos (65,136,216)	6,177	6.86 4.00s/site	0.55		
10. Upper Coast (22,863,789)	2,168	3.88 6.45s/site	0.57		
TOTAL	37,413	53.44 5.14s/site	6.78		

This enables flexibility in analyzing overall storm characteristics.

B. MapReduce Framework for Spatial Data Computing

MapReduce has become the de-facto framework for the data-intensive applications. It is now being used for big data related to geography, sciences, humanities, statistics, etc. There has been previous work for spatial data analysis in MapReduce. Cary [2] shows the construction of R-Tree index from spatial data in MapReduce. It uses the mappers to partition the data and then every partition is sent to a different reducer which in turn build the R-Tree index on the input. Google used the MapReduce framework to study road alignments by combining satellite and vector data [3]. The work focused more on the complexity of the problem than the implementation in MapReduce. Hadoop[®] was also used to build octrees for later use in earthquake simulations at a large scale [4]. Octrees were built in the bottom up fashion in their approach. Mappers were used to first generate the leaf nodes and then reductions were performed to merge two homogeneous leaf nodes into a sub tree. This was done in iterations to build the final sub tree.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

In this work, we use the MapReduce framework to analyze large amounts of raw rainfall data. With this new system, the original input data structure was fully utilized in order to create more efficient algorithms for storm identification. It eliminates the major performance issue with the previous system, which mostly has to do with the retrieval of relational data overhead. The experimental results show significant improvement on both local storm and hourly storm identifications processes. This will allow hydrologists to perform: (1) storm analysis (both location-specific and storm-specific) such as storm frequency and characteristics analysis and flood prediction and (2) storm mining such as clustering on types of the storm and trajectory analysis, more efficiently.

B. Future Work

We will work on the computation of overall storm identification using the MapReduce framework. We will also be working on parallelizing the computation of storm area, storm center, and within storm variations [24] by the use of MapReduce framework.

REFERENCES

- [1] K. Jitkajornwanich, R. Elmasri, C. Li, and J. McEnery, "Extracting Storm-Centric Characteristics from Raw Rainfall Data for Storm Analysis and Mining," Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data (ACM SIGSPATIAL BIGSPATIAL'12), 2012, pp. 91-99.
- [2] A. Cary, Z. Sun, V. Hristidis, and N. Rishe, "Experiences on Processing Spatial Data with MapReduce," Proceedings of the 21st International Conference on Scientific and Statistical Database Management (SSDBM'09), 2009, pp. 302-319.
- [3] X. Wu, R. Carceroni, H. Fang, S. Zelinka, and A. Kirmse, "Automatic Alignment of Large-Scale Aerial Rasters to Road-maps, Geographic Information Systems," Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS'07), 2007.
- [4] S. W. Schlosser et al., "Materialized Community Ground Models for Large-Scale Earthquake Simulation," Proceedings of the 2008 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC'08), 2008.
- [5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04), 2004.
- [6] C. Lam, Hadoop in Action. Dreamtech Press, New Delhi, 2011.
- [7] B. Franks, Taming The Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics. John Wiley & Sons, Inc., Hoboken, New Jersey, 2012.
- [8] DevZone, Big Data Bibliography. O'Reilly Media, 2011.
- [9] R. Elmasri and S. Navathe, Fundamentals of Database Systems, 6th ed. Pearson Education, Massachusetts, 2010.
- [10] A. Overeem, T. A. Buishand, and I. Holleman, "Rainfall Depth-Duration-Frequency Curves and Their Uncertainties," Journal of Hydrology, vol. 348, 2008, pp. 124-134.
- [11] W. H. Asquith, M. C. Roussel, T. G. Cleveland, X. Fang, and D. B. Thompson, "Statistical Characteristics of Storm Interevent Time, Depth, and Duration for Eastern New Mexico, Oklahoma, and Texas," Professional Paper 1725. U.S. Geological Survey (USGS), 2006.
- [12] W. H. Asquith, "Depth-Duration Frequency of Precipitation for Texas," Water-Resources Investigations Report 98-4044. U.S. Geological Survey (USGS), 1998.
- [13] J. Lanning-Rush, W. H. Asquith, and R. M. Slade, "Extreme Precipitation Depth for Texas, Excluding the Trans-Pecos Region," Water-Resources Investigations Report 98-4099. U.S. Geological Survey (USGS), 1998.
- [14] W. H. Asquith and M. C. Roussel, "Atlas of Depth-Duration Frequency of Precipitation Annual Maxima for Texas," Scientific Investigations Report 2004-5041 (TxDOT Implementation Report 5-1301-01-1). U.S. Geological Survey (USGS), 2004.
- [15] W. H. Asquith, D. B. Thompson, T. G. Cleveland, and X. Fang, "Synthesis of Rainfall and Runoff Data used for Texas Department of Transportation Research Projects 0-4193 and 0-4194," Open-File Report 2004-1035. U.S. Geological Survey (USGS), 2004.
- [16] W. H. Asquith, "Summary of Dimensionless Texas Hyetographs and Distribution of Storm Depth Developed for Texas Department of Transportation Research Project 0-4194," Report 0-4194-4. U.S. Geological Survey (USGS), 2005.
- [17] J. S. Horsburgh, D. G. Tarboton, D. R. Maidment, and I. Zaslavsky, "A Relational Model for Environmental and Water Resources Data," Water Resources Research, 2008.
- [18] Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI), "ODM Databases," retrieved: October 26, 2011, from: http://his.cuahsi.org/odm_databases.html.
- [19] National Oceanic and Atmospheric Administration (NOAA), "National Weather Service River Forecast Center: West Gulf RFC (NWS-WGRFC)," retrieved: December 31, 2011, from: <http://www.srh.noaa.gov/wgrfc/>.
- [20] NOAA's National Weather Service, "The XMRG File Format and Sample Codes to Read XMRG Files," retrieved December 31, 2011, from: <http://www.nws.noaa.gov/oh/hrl/dmip/2/xmrgformat.html>.
- [21] J. McEnery, "CUAHSI HIS: NWS-WGRFC Hourly Multi-sensor Precipitation Estimates," retrieved: December 31, 2011, from: http://hiscentral.cuahsi.org/pub_network.aspx?n=187.
- [22] Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI), "HydroDesktop," retrieved: October 26, 2011, from: <http://his.cuahsi.org/hydrodesktop.html>.
- [23] Virginia Department of Conservation and Recreation, "Stormwater Management: Hydrologic Methods," retrieved: May 2, 2012, from: http://dcr.cache.vi.virginia.gov/stormwater_management/documents/Chapter_4.pdf.
- [24] A. Suyanto, P. E. O'Connell, and A. V. Metcalfe, "The Influence of Storm Characteristics and Catchment Conditions on Extreme Flood Response: A Case Study Based on the Brue River Basin," U.K. Surveys in Geophysics, vol. 16, 1995, pp. 201-225.

A RESTful Approach for a Cloud Gateway

Chang Ho Yun, Jong Won Park, Hae Sun Jung,
Yong Woo LEE

School of Electrical & Computer Engineering
The Ubiquitous-City (Smart City) Consortium, the
University of Seoul
Seoul, South Korea

{touch011, comics77, banyasun, ywlee}@uos.ac.kr

Haeng Jin Jang

Korea Institute of Science and Technology Information
Seoul, South Korea
hjjang@kisti.re.kr

Abstract— Polar research is active nowadays since it gives us many kinds of information about global climate change so that we can respond to it more properly. We found that the research can have much benefit by using a data farm approach, which gives high performance computing power without limit. Here, we are interested in providing more convenient and useful interface to use the high performance computing power in the polar research. This paper presents a cloud gateway, that is, a science research gateway which supports cloud and grid computing in a unique REST architecture. It provides facilities and interfaces which enable polar researchers to do computer supported remote collaborative work as well as to use data farms.

Keywords-Cloud Computing; Grid Computig; REST; Web Service; Science Gateway; Polar Research.

I. INTRODUCTION

Recently, the change of global climates has emerged as a global agenda [1]. It has attracted much attention so far and would do so more and more in future. Korea has been doing polar research to cope with the problem as well as many other useful issues which are global issues nowadays. Cloud computing and Grid computing can significantly help the polar research experts. We develop a scientific research gateway, called as Cloud Gateway, which enables the polar research experts to use the technologies with easiness and efficiency.

The Cloud Gateway also provides a collaboration environment for the various kind of polar research. Polar researchers can do computer supported cooperative work and share data among interesting research groups beyond geographical gaps and regardless of different working times. The management of polar metadata can be easy and efficient with it.

The Cloud Gateway consists of three tiers - Infrastructure Tier, Processing Tier and Presentation Tier - to support the distributed environment. It uses RESTful Web services [2] for the data transmission and service request between each tier. Also, it collects, processes and provides many kinds of information such as Portable Batch System (PBS) accounting information, information of file system, CPU information and slave node information to users.

The Cloud Gateway was designed to meet the following two requirements. Firstly, it should support geographically scattered multiple computing facilities such as clusters, web servers, databases, etc. through integrated service. Secondly, the service should be provided in a user-transparent way. That is, it should enable polar researchers to use the computing resources without pushing them to know any detailed knowledge of the underlying technologies of the Cloud Gateway.

The Cloud Gateway also has the following three distinctive factors. Firstly, it has three-tier architectures as explained before. Secondly, it uses REST technologies so that users can access geographically scattered multiple computing facilities through a single interface as explained before. Thirdly, the web portal is used as the user access point to the Cloud Gateway in order to meet the user transparency requirement.

The outline of the paper is organized as follows: Section 2 investigates related works. Section 3 outlines the design of the Cloud Gateway. Section 4 explains how it was implemented. Finally, Section 5 gives conclusions and our plan to the future works.

II. RELATED WORKS

Currently existing science gateways usually provide high end resources to a community of users, scientists, and engineers through web-based graphical interface [3]. A common approach in the previous generation was to adopt the JSR 168 portlet component model and WSDL/SOAP style web services.

The TeraGrid User Portal serves as a launch pad for new users and a control panel for current users by integrating TeraGrid Resource Provider, services, and information into a single web interface serving a national community of computational researchers [4][5].

The Linked Environments for Atmospheric Discovery (LEAD) Portal is a science application portal which was designed to enable effective use of Grid resources in exploring mesoscale meteorological phenomena [6].

WLCG provides graduate and accurate verification of performance of hardware resources such as CPU, storage, and network. It also provides the middleware services for

Grid projects and the LHC experiment-specific software applications [7].

PolarGrid portal added current social networking techniques to a typical science gateway model to enable a scientific collaboration [8]. It uses a RESTful Web-service and Web 2.0 technologies. However, it just uses them for user interface, not for managing computing resources.

World Wide Web was usually chosen as preferred infrastructure. Thus, most initiatives adopted Web technologies such as CORBA (Common Object Request Broker Architecture) [9], OLE/DCOM [10], SOAP (Simple Object Access Protocol), etc. Especially, SOAP is the de facto standard in current science gateways.

REST is widely used because of its simplicity and lightweight [11]. McFaddin et. al. [12] and Christensen [13] proposed RESTful service for mobile environments. Volkel [14] proposed RESTful wiki architecture. Twitter [15], Flickr [16], Amazon Simple Storage Service (Amazon S3) [17], Amazon Elastic Compute Cloud (Amazon EC2) [18], and others provide a REST application programming interface (API) to their users. However, the RESTful approach has been seldom applied to the management of science gateways. Contrastingly, our Cloud Gateway provides RESTful Web services to manage it.

III. ARCHITECTURE

Our Cloud Gateway uses the three tier Architecture and RESTful Web service technologies in order to support a distributed computing environment. There, technologies of Java platform was used in order to make the Cloud Gateway be independent of computing platform.

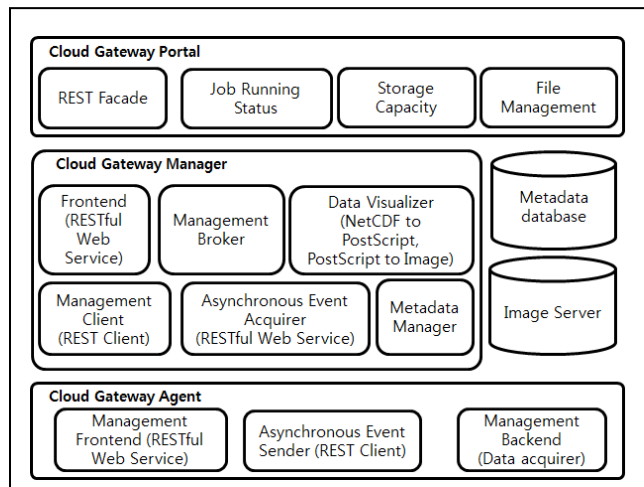


Figure 1. The architecture of the Cloud Gateway.

Figure 1 shows the architecture of the Cloud Gateway. The tier 1 of the Cloud Gateway is the Cloud Gateway Agent. The Cloud Gateway Agent is installed on PBS Clusters. It's role is to communicate with the Cloud Gateway Manager. It collects information of CPU, file system, accounting, etc. in the PBS system and passes an asynchronous message to the Cloud Gateway Manager when accounting information is

updated. The tier 2 of the Cloud Gateway is the Cloud Gateway Manager. It does logical processing. It determines whether the data exist in the metadata database or not. If the data does not exist, then it collects the data from the Cloud Gateway Agent when the Cloud Gateway Portal requests information of accounting or file system. The Cloud Gateway Portal shows information of accounting and system resources in the PBS clusters through graph or table. It provides interface to manage the file system of each node. Because the Cloud Gateway uses 3 tier architecture, users can easily manage the scatted resources in cloud computing environments and/or grid computing environments. Figure 2 shows the operational concept of our three-tier architecture that can manage multiple clusters.

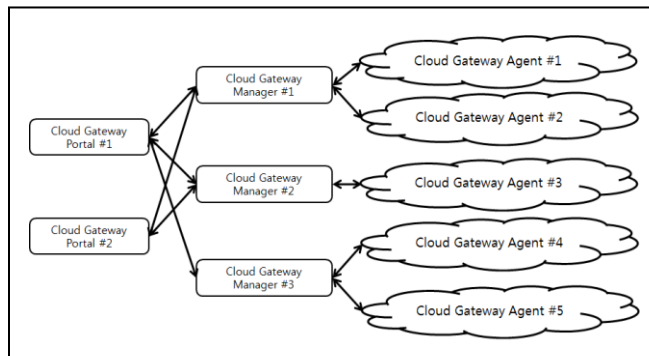


Figure 2. The multiple cluster management of the Cloud Gateway.

A. Cloud Gateway Agent

The Cloud Gateway Agent gives RESTful web service and is installed on the master node of the PBS cluster system. The components of the Cloud Gateway Agent are the Management Backend, the Management Agent and the Asynchronous Event Sender.

The Management Backend provides a management interface to the Management Agent and the Asynchronous Event Sender. It returns proper responses such as the CPU information, the file system information, the accounting information and the detail information of slave nodes of the PBS system to the Management Agent according to requests from the Management Agent. The Asynchronous Event Sender receives an event from the Management Backend and sends a notification of the event to the Asynchronous Event Acquirer of the Cloud Gateway Manager. And, it also sends the request message when the Cloud Gateway Agent is added to the Cloud Gateway Manager in a first time. The Management Frontend returns the response according to requests of the Management Client of the Cloud Gateway Manager through RESTful web service interface. The resource managed by the Cloud Gateway Agent can be accessed through HTTP methods such as GET, POST, PUT, and DELETE.

Figure 3 shows RESTful web service interface of the Cloud Gateway Agent. It consists of the ci (Common Information) and the pi (PBS Information).

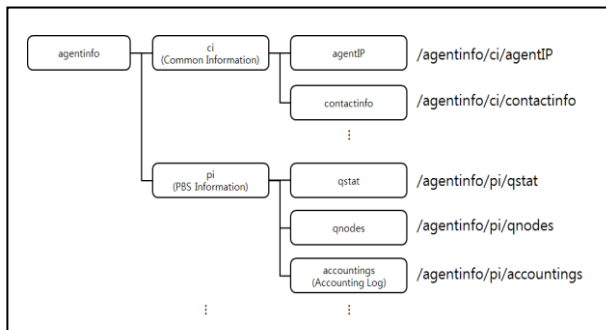


Figure 3. RESTful web service interface of the Cloud Gateway Agent.

B. Cloud Gateway Manager

The components of the Cloud Gateway Manager are the Frontend, the Management Broker, the Management Client, The Asynchronous Event Acquirer, the Metadata Manager, the Data Visualizer, the Metadata Database and the Image Server.

The Frontend provides a RESTful web service-based interface for the Cloud Gateway Portal. It receives requests from the Cloud Gateway Portal, passes them to the Management Broker, and returns responses. Figure 4 shows the RESTful web service interface of the Frontend. It can also be accessed through HTTP methods such as GET, POST, PUT, and DELETE.

The Management Broker is accessible by the Frontend and the Asynchronous Event Acquirer and provides them with the response according to the request from them. It manages the agents, obtains monitoring information from the Metadata Manager and the Management Client. The Management Broker cannot access other tiers and the Meta database and uses the Management Client to access other tiers. The Metadata Manager accesses the Meta database.

The Management Client and the Asynchronous Event Acquirer communicate with the Cloud Gateway Agent. They do not do any logical behaviors and pass the event to the Management Broker. The Management Client requests the necessary data such as the status of PBS slave nodes to the Cloud Gateway Agent which uses them to perform the service of the Management Broker.

The Asynchronous Event Acquirer receives the asynchronous event from the Asynchronous Event Sender of the Cloud Gateway Agent. The Metadata Manager can only access the Metadata Database. The Data Visualizer processes images for collaboration among polar researchers. These images are stored in the Image server. In request of the Cloud Gateway Portal, they are provided through the Frontend.

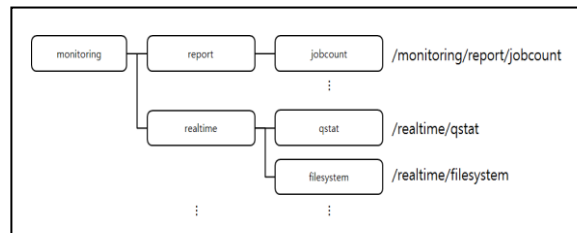


Figure 4. REST web service interface of the Cloud Gateway Manager.

C. Cloud Gateway Portal

The Cloud Gateway Portal is a user transparent web portal. Its components are the REST Facade, the Job Running Status, the Storage Capacity and the File Management. The Cloud Gateway Portal uses Springframework and Ajax/Javascript.

The REST Façade manages the requests and the responses from the Cloud Gateway Portal to the Cloud Gateway Manager. The Job Running Status requests accounting information of PBS cluster using the REST Façade, translates the response to contents such as the chart and the table and shows them. The Storage Capacity component requests the storage and the file system information of PBS cluster using the REST Façade, translates the response to contents such as the chart and the table and shows them. The File Management requests the information of the file systems in PBS cluster using the REST Façade, help download and upload and modify the files.

IV. IMPLEMENTATION

We implemented the Cloud Gateway on a Linux system using Java language. However, the Cloud Gateway Manager and the Cloud Gateway Portal can be installed on any operating system with Java and Tomcat. MySQL was used as the Metadata database. We used the Restlet package [19] to build the RESTful Web services. Jnotify package [20] was used to monitor PBS accounting as a tool [21] for the Cloud Gateway Agent.

The operations of the Cloud Gateway can be one of the following three types. First, the response result for request from the Cloud Gateway Portal exists in the Metadata Database of the Cloud Gateway Manager. Second, it does not exist in the Cloud Gateway Manager, so the Cloud Gateway Manager queries the request to the Cloud Gateway Agent. Third, the Cloud Gateway Agent sends the notification of changing the status of the PBS cluster to the Cloud Gateway Manager.

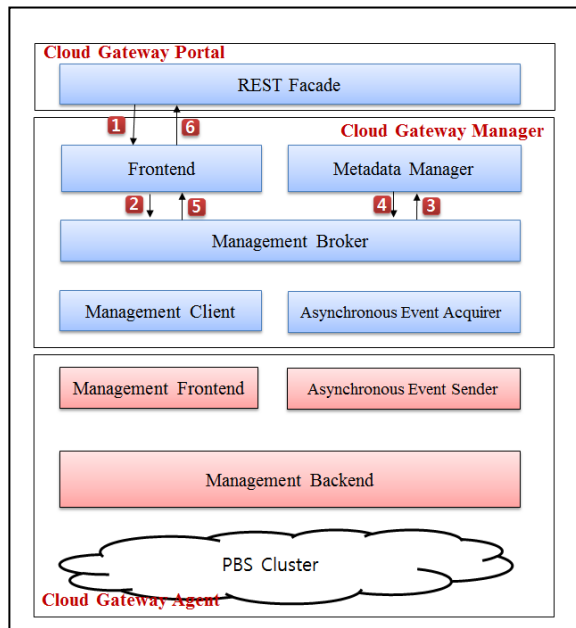


Figure 5. The processing of Job accounting.

An example of the first type is shown in Figure 5. When the user requests job accounting information, the REST Façade of Cloud Gateway Portal requests URL of “/monitoring/report/jobcount” to the RESTful web service interface of the Frontend of the Cloud Gateway Manager. Then, the Frontend sends the message to the Management Broker and the Management Broker analyses the request. Because the Management Broker can find response result in the Metadata Database, the Management Broker collects accounting information by using the Metadata Manager and returns the result to the REST Façade through Frontend.

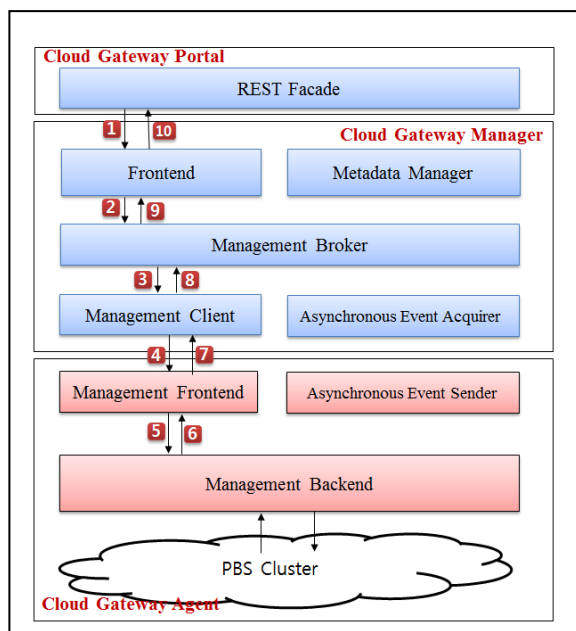


Figure 6. The processing of qnodes request.

An example of the second type is shown in Figure 6. When the user requests the information of slave nodes in PBS cluster, the REST Façade requests the URL of “/realtime/qnodes” to the RESTful web service of the Frontend of the Cloud Gateway Manager. The Frontend sends the request to the Management Broker and the Management Broker analyses it. Because the data are not in the metadata database, the Management Broker requests the URL of “/agnetinfo/pi/qnodes” to the Management Frontend of the Cloud Gateway Agent through the Management Client. The Management Frontend sends the received request to the Management Backend and the Management Backend queries the request to PBS cluster. The result of the query is returned to the Management Client through the Management Frontend. The Management Client sends it to the Management Broker. The Management Broker returns the result to the REST Façade of the Cloud Gateway Portal.

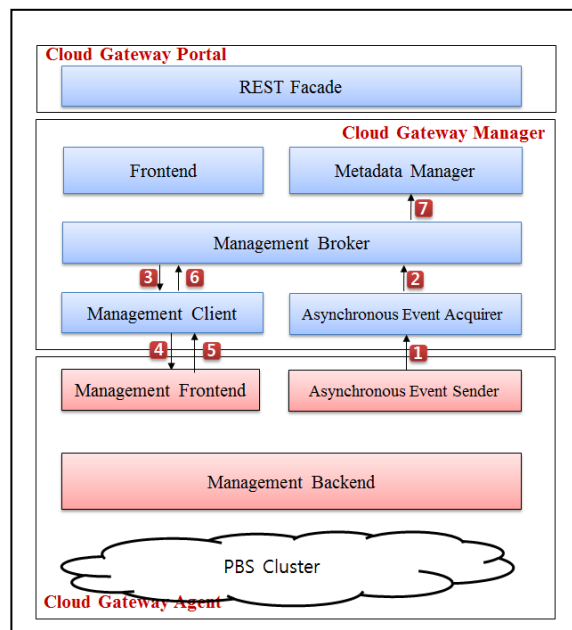


Figure 7. The processing of asynchronous event.

An example of the third type is the asynchronous event in the Management Backend. The Management Backend of the Cloud Gateway Agent monitors the accounting logs of PBS cluster. If the logs are found to be changed, then the Management Backend sends the asynchronous event to the Asynchronous Event Acquirer through the Asynchronous Event Sender. The Asynchronous Event Acquirer sends the event to the Management Broker. The Management Broker analyses it and checks the need to update the Metadata database. If the Metadata Database is needed to be updated, then the Management Broker requests the data to the Management Frontend of the Cloud Gateway Agent through the Management Client. The Management Frontend returns the result that is acquired from the Management Backend. Then the Management Client sends the result to the Management Broker. Now the Management Broker updates

the Metadata Database using them through the Metadata Management.

Figure 8, 9, and 10 show the snapshot of job running status, the snapshot of job status, the snapshot of file management respectively.

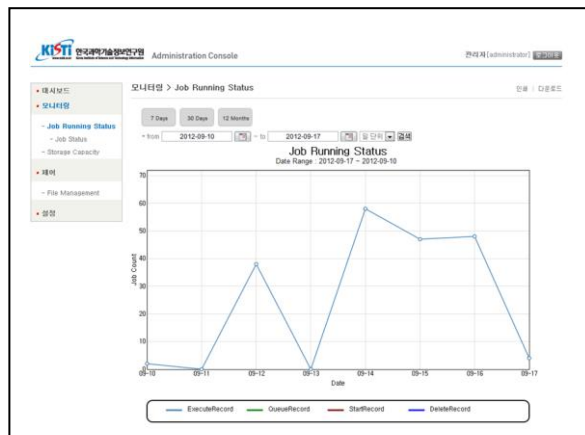


Figure 8. The snapshot of job running status.

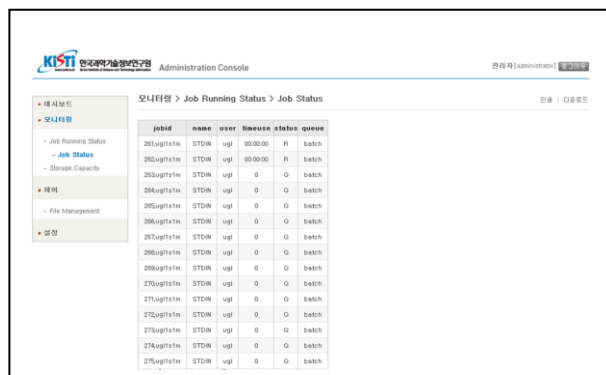


Figure 9. The snapshot of job status.

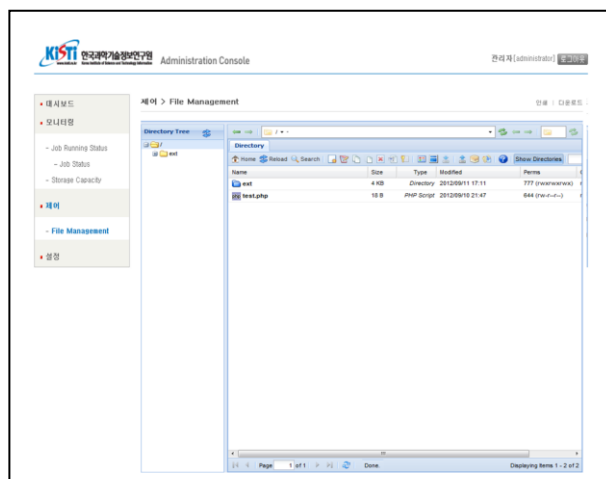


Figure 10. The snapshot of file management.

V. CONCLUSION

This paper proposed our cloud gateway model and its RESTful approach in order to support cloud computing, Grid computing, computer supported collaboration, etc. for the polar research. The Cloud Gateway uses the three tier architecture to provide the RESTful web service. Therefore, users can access geographically scattered multiple computing facilities such as clusters, web servers and databases through a single interface easily, efficiently and user-transparently. The future works are planned to add analysis tools for geospatial query components and visualization components.

ACKNOWLEDGMENT

This study was supported by the Ministry of Education, Science and Technology in Korea and KISTI (Korea Institute of Science and Technology Information). This study was also supported by the Seoul Research and Business Development Program, Smart (Ubiquitous) City Consortium (10561) and Seoul Grid Center. This work was also supported by the 2011 research fund of the University of Seoul (Yong Woo LEE : the corresponding author).

REFERENCES

- [1] M. Kok, W. Vermeulen, A. Faaij, and D. Jager, Global Warming and Social Innovation: The Challenge of a Climate Neutral Society, Earthscan Publications Ltd, 2002.
- [2] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures, doctoral dissertation, 2000.
- [3] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, and S. Pamidighantam, "TeraGrid Science Gateways and Their Impact on Science," IEEE Computer, vol. 41, Nov. 2008, pp. 32-41, doi: 10.1109/MC.2008.470.
- [4] M. Dahan, E. Roberts, and J. Boisseau, "TeraGrid User Portal v1.0: Architecture, Design, and Technologies," Proc. International Workshop on Grid Computing Environments, Nov. 2006.
- [5] J. Basney, V. Welch, and N. Wilkins-Diehr, "TeraGrid Science Gateway AAAA Model: implementation and lessons learned," Proc. The 2010 TeraGrid Conference, Aug. 2010, pp. 1-6, doi: 10.1145/1838574.1838576.
- [6] M. Christie and S. Marru, "The LEAD Portal: a TeraGrid gateway and application service architecture," Concurrency and Computation: Practice and Experience, vol. 19, Apr. 2007, pp. 767-781, doi: 10.1002/cpe.1084.
- [7] D. Bonacorsi and T. Ferrari, "WLCG Service Challenges and Tiered architecture in the LHC era," IFAE 2006, pp. 365-368, doi:10.1007/978-88-470-0530-3_68.
- [8] Z. (G.) Guo, R. Singh, and M. Pierce, "Building the PolarGrid Portal Using Web 2.0 and OpenSocial," Proc. The fifth Grid Computing Environments Workshop, 2009, article no.5, doi: 10.1145/1658260.1658267.
- [9] Object Management Group, Inc, Cobra [retrieved: Mar. 2013], http://www.corba.org/
- [10] Microsoft, DCOM [retrieved: Mar. 2013], http://www.microsoft.com/COM/
- [11] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big" web services: making the right architectural decision," Proc. The 17th international conference on World Wide Web (WWW 08), 2008, pp. 805-814, doi: 10.1145/1367497.1367606.

- [12] S. McFaddin, D. Coffman, J. H. Han, H. K. Jang, J. H. Kim, J. K. Lee, M. C. Lee, Y. S. Moon, C. Narayanaswami, Y. S. Paik, J. W. Park, and D. Soroker, "Modeling and Managing Mobile Commerce Spaces Using RESTful Data Services," Proc. The Ninth International Conference on Mobile Data Management (MDM 08), Apr. 2008, pp. 81-89, doi: 10.1109/MDM.2008.38.
- [13] J. H. Christensen, "Using RESTful web-services and cloud computing to create next generation mobile applications," Proc. The 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA 09), 2009, pp. 627-634, doi:10.1145/1639950.1639958.
- [14] M. Volkel, "Semwiki: a restful distributed wiki architecture," Proc. The 2006 international symposium on Wikis (WikiSym 06), 2006, pp. 141-142, doi:10.1145/1149453.1149486.
- [15] Twitter, Twitter REST API [retrieved: Mar. 2013], <https://dev.twitter.com/docs/api>
- [16] Yahoo, flickr REST API [retrieved: Mar. 2013], <http://www.flickr.com/services/api/response.rest.html>
- [17] Amazon, Amazon Simple Storage Service REST API [retrieved: Mar. 2013], <http://docs.amazonwebservices.com/AmazonS3/latest/API/APIRhest.html>
- [18] Amazon, Amazon Elastic Compute Cloud REST API [retrieved: Mar. 2013], <http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/Welcome.html>
- [19] Noelios Technologies, Restlet [retrieved: Mar. 2013], <http://www.restlet.org>.
- [20] O. Yadan, jnotify [retrieved: Mar. 2013], <http://jnotify.sourceforge.net/>
- [21] R. Mach, PBS XML Accounting Toolkit [retrieved: Mar. 2013], <http://pbsaccounting.sourceforge.net/>

Adaptive Multimedia Learning Delivered in Mobile Cloud Computing Environment

Aleksandar Karadimce, Danco Davcev

Faculty of Computer Science and Engineering, University Ss Cyril and Methodius

Skopje, R. Macedonia

akaradimce@ieee.org, danco.davcev@finki.ukim.mk

Abstract— The process of integrating multimedia files, such as different types of learning objects (video, images, audio, animation, etc.) to m-learning systems requires more computational resources than mobile devices can provide. Considering mobile device limitations, such as storage, computing power and bandwidth, we propose a mobile cloud computing in order to deliver adaptive multimedia learning courses to students. In this paper, we propose a PaaS cloud-based framework, which offloads the process of dynamically adapting the multimedia content to the context-aware mobile learning environment. Hence, the student is provided with multimedia that is tailored to his or her cognitive style and the content is adapted according to context – aware network conditions.

Keywords- mobile computing; cloud computing; adaptive multimedia learning; user profile; context-aware.

I. INTRODUCTION

Incorporation of mobile devices during m-learning is done in order to improve and increase the scalability, collaboration and availability of learners. Multimedia learning systems typically include different kinds of multimedia resources such as audio, video, images, text etc., since they provide an efficient learning environment. There are always newer multimedia functionalities available on mobile devices that need to be exploited during the process of m-learning.

Mobile language learning with multimedia using the image and audio-based training has opened the opportunity to develop mobile augmented and virtual reality spheres [1]. However, on the other hand, we can see the opportunities that mobile cloud computing architecture has provided for mlearning, with different cloud service models in university education, using a Software-as-a-Service (SaaS) cloud model, as in [2]. Similar related work has been done in [3], with the promotion of potential of m-learning using cloud computing for talent training in universities. In [4], X. Bai presented an application for interactive learning through mobile devices combined with the new technology of cloud computing where live lectures from the instructor's webcam are streamed to the cloud. Hence, the students interact with the lecturer and this increases the collaboration between them.

In [12], D. Kovachev et al. introduce the future prospects of web and mobile multimedia development for creating the next generation of mobile web applications and the new standards and protocols like HTML5 and XAML [16].

Similar to the research done in [8], we have also used the web browser that is an integral part of mobile devices to be used for accessing the adaptive multimedia learning system. The main contribution of this paper is exploiting the processing power of mobile cloud computing to adapt the multimedia content to the cognitive style and context – aware network conditions of the mobile user.

This paper is organized as follows: Section II presents our proposed architecture of the adaptive multimedia learning framework. Section III describes the data workflow for provisioning multimedia learning requests. Section IV presents the results from our experimental mobile multimedia learning system. Finally, Section V concludes the paper.

II. ARCHITECTURE OF ADAPTIVE MULTIMEDIA LEARNING SYSTEM

Existing m-learning systems typically include different kinds of multimedia resources because they help learners to be more interactive and interested for collaboration. Using the existing services on mobile devices, students are able to send their requests to be processed within the mobile cloud computing environment, in order to receive diverse multimedia learning resources. One of the first generic frameworks for mobile learning through cloud computing designed for education practitioners was presented by X. Bai [4]. They have adapted the course material for a mobile device in the form of learning content and developed a prototype of mobile-based assessment as a proof-of-concept for mobile learning. A similar framework could be found in [5]. That research has introduced an interactive mobile live video learning system in a cloud environment. Using a camera, the instructor's video presentation was captured and then was uploaded on a private cloud. Later, students using GPRS/WiFi connectivity on mobile devices are able to progressively download or replay the video [5].

The proposed adaptive multimedia learning framework (see Figure 1) in this research adopts the multimedia content to the cognitive style and context – aware network conditions of the mobile user. All of the requests from student's mobile device are sent to the mobile cloud and the response is appropriate multimedia content. The mobile cloud computing role is to offload and to reduce the workload of mobile devices by exploiting the remote multimedia processing resources in the cloud. That way, all the SQL queries that are sent from students and professors and the heavy-duty processing tasks will be executed in the mobile cloud.



Figure 1. Architecture of adaptive multimedia learning framework

Processing of the requests in the mobile cloud starts with gathering the context-aware information from the mobile user. Firstly, the mobile request is processed by the Request broker and scheduled for execution in the cloud. Next, Content adaptation component is analyzing the request and checks that the multimedia content is adapted to the context-aware conditions and user cognitive style. The proposed m-learning development environment is based on the Platform as Service (PaaS) cloud model that comes with integrated developer tools, a database management system and a web server. The web browser platform can work on different operating systems for mobile devices, and, in that manner, students and professors simultaneously can access the multi-tenant cloud-based platform from any location, at any time. The students can work on their application independently and efficiently without additional problems with software installation and compatibility issues.

Depending on the context-aware network conditions and the cognitive style estimation, the multimedia files are adapted and the request is sent to the Content delivery component to broadcast the appropriate information. The main advantage of the proposed architecture is that it offers a direct and flexible connection between the student and the mobile cloud environment. Different kinds of mobile devices (iPhone, HTC, Nokia etc.) using the diversity of access networks can connect to the Internet or telecommunication networks (using WiFi, WiMAX, UMTS, GPRS, HSDPA, 4G or LTE) and provide access to the needed service. The intention is to provide a set of mobile services that will allow mobile devices to communicate with the mobile cloud.

III. DATA WORKFLOW FOR PROVISIONING MULTIMEDIA LEARNING REQUESTS

The main focus of the proposed adaptive multimedia learning framework are the multimedia files. Initially, their delivery depends on context-aware network conditions and the type of mobile device that the students are using. Therefore, multimedia files need to be adapted according to the available bandwidth of the network connection and they should be encoded with the corresponding format and coding of the user’s mobile device. Because of existing mobile device limitations, such as storage, computing power and bandwidth, we propose to use mobile cloud computing in order to deliver adaptive multimedia learning content. The results from our research have provided the data workflow diagram for provisioning multimedia learning requests in the adaptive multimedia learning framework.

The advantage of the proposed architecture is that all of the encoding and adaptation of the multimedia learning content is done in the mobile cloud and the ready multimedia data (MM data) is streamed back to the user. The top layer of the data workflow is dedicated to the mobile device. From there, the request is sent to the cloud. The collection station gathers all necessary context-aware information: type of mobile device, OS of mobile device and bandwidth settings of the network connection (see Figure 3).

Before users start using an application for the first time, cognitive style estimation will be conducted. We have used the Verbal-Visual Learning Style Rating (VVLSR) questionnaire, intended to tap the cognitive perception style [15]. The VVLSR questionnaire is an original one-item rating task used to estimate visualizer-verbalizer style dimension using a single question [15]. The question is: “In a learning situation sometimes information is presented verbally and sometimes information is presented visually. Please check mark indicating your learning preference” [15]. The answers rate the preference for visual versus verbal learning on a 7-point scale, as shown in Figure 2. The process of collecting of all VVLSR answers from the application users will fill the user profiles database that is stored in the mobile cloud. Verbalizers are users that have provided answers counted -3,-2,-1; visualizers are users with answers counted +3,+2,+1; and bimodal users provide count 0, which are saved in the user profiles database.

Hereafter, the user of mobile device can easily send his or her requests to be processed within the mobile cloud computing environment. After processing the received requests in the Analysis/Estimation engine we will have adapted multimedia content ready to be delivered to the user.

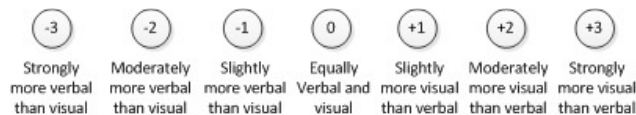


Figure 2. Verbal-Visual Learning Style Rating 7-point scale [15]

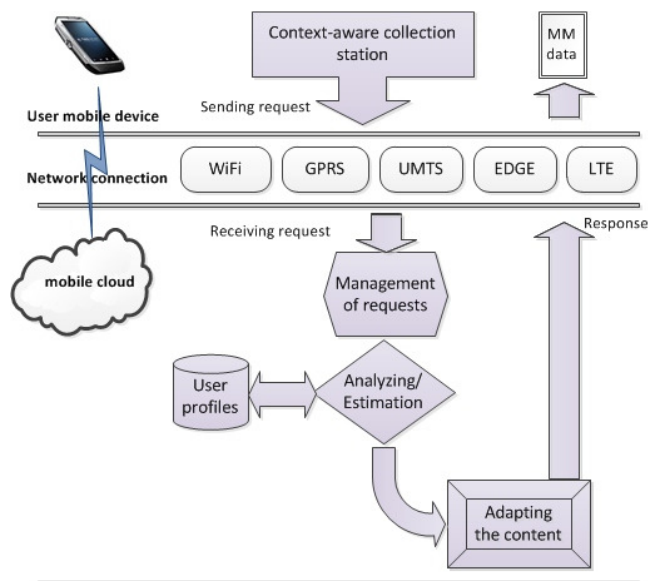


Figure 3. Data workflow for provisioning multimedia learning requests

Using the proposed cloud-based framework the CSE (computer science engineering) students have taken the role of main developers. They have created database objects, have written SQL request queries and have developed the web-based application. The proposed multi-tenant mobile cloud computing environment provides delivery of the distance learning environment where all of the participants are grouped by different roles. The professor has a supervisory role that allows him to be able to access the development environment. He can provide scaffolding and propose more efficient solutions or can interactively support error debugging in the application development progress. In the proposed data workflow, students and professors have independent access to the same development environment, while the end users have access only to the application level.

The main benefit of this proposed architecture is that no software download is required for CSE students to start development on any mobile device with its own web browser that is integral part of the device. The Web service reference is based on a Web Services Description Language (WSDL) document that describes the target Web service. When you create a Web service reference using a wizard, first it analyses the WSDL and collects all the necessary information to create a valid SOAP (the Simple Object Access Protocol) message. Using web-based mobile services, users of mobile devices easily send their requests to be processed within the mobile cloud computing environment. The received HTTP requests can then be managed and scheduled for processing in the mobile cloud.

Using the context – aware network conditions, presented by QoS and the user profile database, which takes into consideration the user cognitive perception style measured by QoE, we have the mapping settings given in Table 1.

TABLE I. MAPPING BETWEEN QoS AND QoE METRICS

QoS Bandwidth	QoE- cognitive perception style	
	Visual perception	Verbal perception
High	HQ images, audio and video	3D graphics, text and audio
Low	Icons and images	Text and audio

The mobile cloud is the development platform that contains the user profile database and the bandwidth network QoS estimator. The mobile devices provide information for current bandwidth. If that bandwidth is above the QoS threshold (1000 KB/s) than the user is in high bandwidth region; otherwise, below the QoS threshold, the user is considered to be in low QoS bandwidth region.

Depending on the available QoS factors, the framework can adapt to low or high bandwidth scenarios. The proposed mapping, given in Table 1, considers at the same time the user cognitive perception style. For better network conditions, in high bandwidth, the system can deliver more dynamic and high quality information. However, in low bandwidth conditions we have estimated that more static media is delivered (icons, images, text and audio). Therefore, according to the context –aware preferences, the system is providing adaptive change of the multimedia type [6]. In the last step, the requested multimedia learning content is streamed back to the mobile device. The mobile user receives dynamically adapted multimedia content to the given context compatible with its mobile device.

IV. EXPERIMENTAL MOBILE MULTIMEDIA LEARNING SYSTEM

Cloud computing is a real benefit for young and developing universities that do not have diverse computer laboratory facilities. With the integration of cloud computing technology for the courses, such as Distributed Database systems, requirements for dedicated development platform and intensive computational resources are inevitable. There already have been different related studies for cloud service models in the university, using a Software-as-a-Service (SaaS) cloud model, as in [2]. Similar related work has been done in [3], with the promotion of the potential for m-learning using the cloud computing, and the stress is put on virtualization, using the Infrastructure-as-a-Service (IaaS) cloud model.

We have proposed an experimental mobile multimedia learning system that is based on the Platform as Service (PaaS) cloud model, and it will extend the potential for developing web applications that require database background. The Oracle APEX [13] platform is a comprehensive web-based SQL and application development environment that delivers platform for fast, reliable development and running on web applications [7].

The Oracle APEX web-based platform in a mobile cloud environment allows us to write our custom SQL query and generate reports according our needs directly from the mobile device, see Figure 4.

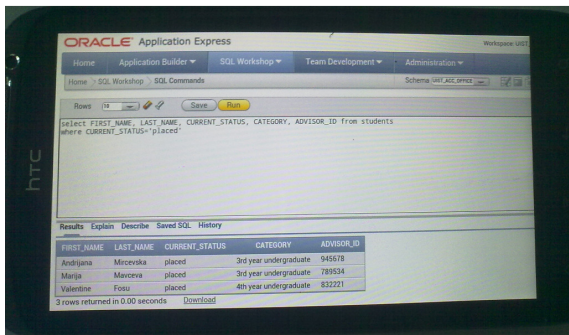


Figure 4. Running SQL query on Oracle APEX using HTC.

After we have compiled the configuration of the application, we have run experiments with three most used mobile devices: iPhone, HTC, Sony Ericsson and Nokia, which can access the Oracle APEX framework. Using the APEX platform, university students have used a simple use case web-based application for online shopping. The database system consisted of 3 tables: Customers, Products and Orders.

Figure 5 provides a comparison overview of the Products table which contains multimedia content, presented on iPhone, HTC and Nokia mobile devices. Using this overview, we cannot distinguish any significant difference when presenting the multimedia content between the three types of mobile devices.

We have done similar comparisons for the Customers and Orders database tables. There was also no difference in the display of information between the three mobile devices. Significant dissimilarity between the mobile devices was noticed in the Reports in Figure 6, where a web-based OLAP report is executed. Here, Nokia and iPhone mobile devices did not provide the expected multimedia graphs for the Report of Sales by Category/Month. The Sony Ericsson mobile device that uses Android mobile OS, on the other hand created a colorful histogram for the Report of Sales by Category/Month, as seen in Figure 6.

We noticed from our research that there was also a different adaptation of the multimedia content for different interfaces, depending on the web browser used by the mobile device because they displayed the same code page differently. Adapted multimedia content is usually compressed using compression algorithms or codecs in order to achieve smaller file size for faster transmission or more efficient storage [12]. The results from the experiments have shown that using different mobile devices to access a single cloud computing platform, in this case Oracle APEX, produces different user experiences.

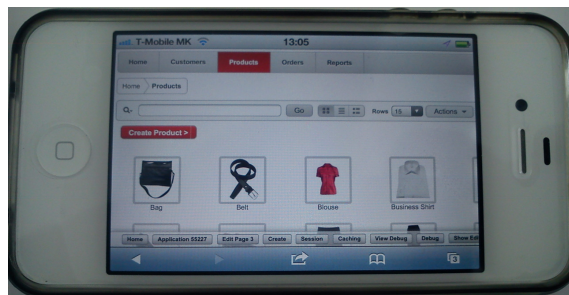
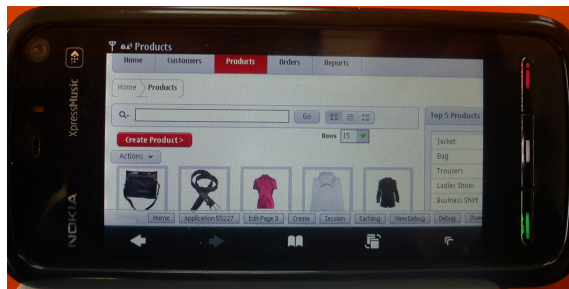
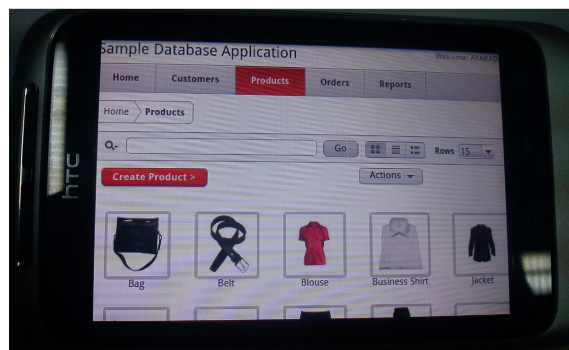


Figure 5. Comparison of database tables for Oracle APEX with HTC (top), Nokia (middle) and iPhone (bottom).

We have used OPNET for our simulations. It provides a comprehensive development environment with a full set of tools including model design, simulation, data collection, data analysis and support on the modeling of communication networks [14]. This simulator provides a way to model the network behaviors by calculating the interactions between modeling devices. We have used the Discrete Event Simulation (DES) because it enables modeling in a more accurate and realistic approach. It creates an extremely detailed, packet-by-packet model for predicting the activities of the network. The simulation models of individual mobile device were developed using the OPNET network simulation software that provides a virtual network communication environment.

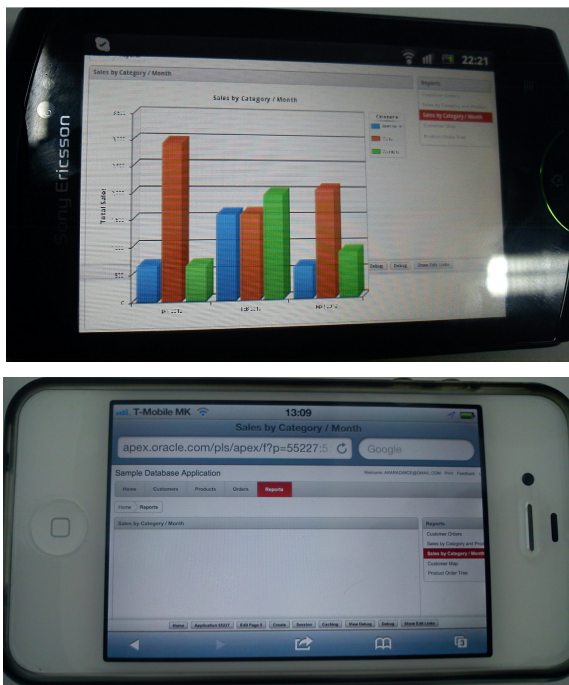


Figure 6. Comparison of reports for Oracle APEX with Sony Ericsson (top) and iPhone (bottom).

The network simulator was configured to run one hour of multimedia learning content, and the comparison of the results is presented in Figure 7. The dark blue line measures the performance from laptop using a classical web-based application without any mobile cloud environment, which takes most of the performance load. The red line measures the HTC mobile device, the green line measures the Nokia mobile device and light blue line measures the iPhone mobile device. For all mobile devices we have used the mobile cloud environment to show the decreased performance load. The simulation results from the multimedia performance load analysis clearly show that mobile devices using applications in a mobile cloud environment have a decreased performance load compared to the classical web-based application.

A. Manjunatha et al.[16] are exploring the data intensive calculations for mobile and cloud computing landscape. Similarly, S. Wang et al.[17] are addressing the adaptive mobile cloud computing techniques for graphic rendering. The integration of mobile and cloud is used for adaptive display virtualization [18]. Similarly, R. S. Khune et al.[19] proposed a cloud-based intrusion detection system for Android mobile devices that provides continuous in-depth forensic analysis to detect any misbehavior in network. In our case study mobile devices with Android mobile OS have presented the complete report with multimedia information. On the other hand, mobile devices that have Symbian and Apple mobile OS have not displayed completely the needed multimedia content. The multimedia content is usually compressed using compression algorithms or codecs, in order to achieve smaller file sizes for faster transmission or more efficient storage [12].

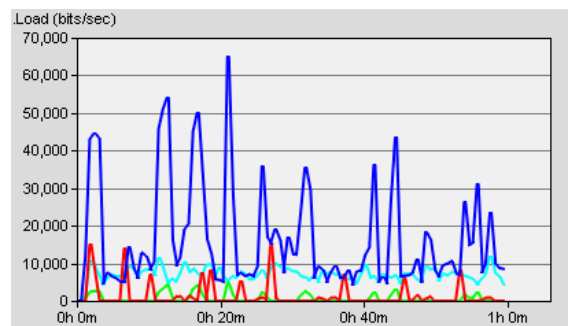


Figure 7. Comparison of performance load in OPNET simulator.

TABLE II. COMPARISON OF DIFFERENT MOBILE DEVICE MEDIA PLATFORMS

Type of media format and codecs support	Android 3.0 [9]	Symbian S60 [10]	Apple mobile OS [11]
Audio	AAC, MP3, MIDI, OGG (vorbis), WAV	MP3, OGG (vorbis), WMA	AAC, HE-AAC, MP3, VBR, AIFF, WAV
Image	JPEG, GIF, PNG, BMP	JPEG, PNG, MBM	JPG, TIFF, GIF
Video	H.263, H.264 AVC, MPEG-4, VP8	WMV, MP4, 3GP	H.264, MPEG-4, Motion JPEG

In Table 2, a comparison of the different mobile device media platforms is based on different formats and coding protocols for Apple, Android and Symbian mobile OS [9-11]. This represents another major challenge that m-learning faces, to adapt multimedia contents in order to be compatible with different mobile devices. Future mobile cloud computing applications should be able to provide conversion of media types to a compatible media format and codecs support for present mobile devices.

V. CONCLUSION

Mobile multimedia learning systems provide an intuitive and collaborative environment where users can experience the advantages of flexibility, portability and scalability. Mobile cloud computing can reduce the workload of mobile devices by exploiting the remote multimedia processing resources in the cloud. Therefore, all the services need to be designed in order to put less workload on the mobile device and allow heavy-duty processing tasks to be done in the cloud.

We have developed a framework for mobile adaptive multimedia learning systems that is delivered using a mobile cloud computing environment. This kind of environment provides users with the appropriate mapping settings between the type of context – aware network conditions presented by QoS and user profile estimation, which takes into consideration the user’s cognitive perception style stated

by QoE. The experiments done using three different types of mobile devices have heightened the importance of choosing the appropriate device to be used in mobile multimedia learning systems.

REFERENCES

- [1] S. Joseph, and M. Uther, "Mobile language learning with multimedia and multi-modal interfaces," Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education. (ICHIT'06), 16-17 Nov. 2006, pp. 124-128, DOI:10.1109/WMTE.2006.30.
- [2] S. Xiaojuan, R. Guang-wen, and W. Zhe, "The application of cloud computing SaaS delivery model in university talents training," Proceedings of the IEEE Digital Manufacturing and Automation (ICDMA), 5-7 August 2011, pp. 1203-1205. DOI:10.1109/ICDMA.2011.296.
- [3] Q. Shuai, and Z. Ming-quan, "Cloud computing promotes the progress of M-learning," Proceedings of the IEEE conference of Uncertainty Reasoning and Knowledge Engineering (URKE), 4-7 August 2011, pp. 162-164, DOI: 10.1109/URKE.2011.6007934.
- [4] X. Bai, "Affordance of Ubiquitous Learning through Cloud Computing," Proceedings of 2010 Fifth International Conference on Frontier of Computer Science and Technology, 978-0-7695-4139-6/10, 2010 IEEE, pp. 78-82, DOI: 10.1109/FCST.2010.109.
- [5] S.Mohana Saranya, and M.Vijayalakshmi, "Interactive mobile live video learning system in cloud environment," Proceedings of the IEEE International Conference on Recent Trends in Information Technology, ICRTIT 2011. 3-5 Jun. 2011. Chennai, pp. 673-677, DOI: 10.1109/ICRTIT.2011.5972458.
- [6] A. Karadimce, and D. Davcev, "Personalized Multimedia Content Generation Using the QoE Metrics in Distance Learning Systems," Proceedings of Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications. ADAPTIVE 2012. 22-27 Jul. 2012. Nice, France, pp. 1-6, ISBN: 978-1-61208-219-6.
- [7] J. Wang, and J. L. Kourik, "Delivering database knowledge with web-based labs," Proceedings of American Society of Business and Behavioral Sciences. ASBBS in Las Vegas. Volume 19 Number 1. February 2012, pp. 923-931.
- [8] L.J. Zhang, "EIC Editorial: Introduction to the Body of Knowledge Areas of Services Computing," Proceedings of IEEE Transactions on services computing, Vol. 1, No. 2. April-June 2008. pp. 62-74. <http://tab.computer.org/tsc/tsc2008020062.pdf> [retrived: February, 2013]
- [9] Android Supported Media Formats. <http://developer.android.com/guide/appendix/media-formats.html> [retrived: February, 2013]
- [10] Symbian OS - News, Tutorial and Development Updates. <http://symbian-os-development.blogspot.com/> [retrived: February, 2013]
- [11] Apple iPhone specification. <http://www.apple.com/iphone/specs.html> [retrived: February, 2013]
- [12] D. Kovachev, Y. Cao, and R. Klamma, "Mobile Multimedia Cloud Computing and the Web," Proceedings of the 2011 Workshop on Multimedia on the Web. MMWEB '11. Graz, Austria, September 8, 2011, pp. 21-26, DOI:10.1109/MMWeb.2011.16.
- [13] D. Baker, and T. Jennings, Oracle Database 2 Day + Application Express Developer's Guide, Release 4.0. E15516-04.Oracle, 2010. http://docs.oracle.com/cd/E17556_01/doc/appdev.40/e15516.pdf [retrived: February, 2013]
- [14] Z. Lu, and H. Yang. Unlocking the Power of OPNET Modeler. Cambridge University Press, January 2012.
- [15] R. E. Mayer, and L. J. Massa, "Three facets of visual and verbal learners: Cognitive ability, cognitive style and learning preference," Published in Journal of Educational Psychology. 833-846. DOI: 10.1037/0022-0663.95.4.833.
- [16] A. Manjunatha, A. Ranabahu, A. Sheth, and K. Thirunarayan, "Power of Clouds In Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development," Published in 2nd IEEE International Conference on Cloud Computing Technology and Science. 496-503.DOI 10.1109/CloudCom.2010.78.
- [17] S. Wang, and S. Dey, "Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications," Published in IEEE Transactions on Multimedia. January 2013. 1-14. ISSN:1520-9210.
- [18] S. Sridhar, G. Satish, G. Raja, and Sumalatha Ramachandran, "Adaptive display virtualization and dataflow model selection (ADVADAMS) for reducing interaction latency in thin clients," in 2012 International Conference on Recent Trends In Information Technology (ICRTIT). April 2012. 233-238. ISBN: 978-1-4673-1599-9.
- [19] R. S. Khune, and J. Thangakumar, "A Cloud-Based Intrusion Detection System for Android Smartphones," in 2012 International Conference on Radar, Communication and Computing (ICRCC). December 2012. 180-184. ISBN:978-1-4673-2756-5.

Digital Signature as a Cloud-based Service

Wojciech Kinastowski

Institute of Control and Information Engineering
Poznan University of Technology
Poznan, Poland
wojtek@kinastowski.pl

Abstract—Cloud-based digital signature can be seen as a model for reliable, convenient, on-demand network access to security infrastructure that performs cryptographic operations of digital signature. This study proposes a protocol for data exchange between signer and signing-enabled cloud environment in the cloud-based digital signature model. It also covers performance results and implementation notes of Signer entity.

Keywords-Digital Signature; Cloud Computing; Cryptography.

I. INTRODUCTION

Recently, cloud has become a new paradigm for delivering computing as a utility. Although the theory behind cloud computing is based on decades of the existing technologies and research, enthusiastic response from developers and widespread acceptance among users confirms that cloud computing is here to stay and likely to play an even more important role as a concept in many fields of information technology, including encryption. Defining cloud computing as a “model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1], and digital signature as “the result of a cryptographic transformation of data which, when properly implemented, provides the services of: origin authentication, data integrity and signer non-repudiation” [2], cloud-based digital signature can be seen as a model for reliable, convenient, on-demand network access to security infrastructure that performs cryptographic operations of digital signatures.

The main difference between a standard digital signature system and a cloud-based one is that, while the first operates in the “close” environment of a personal computer and plugged-in dedicated devices (microchip card and card reader), the cloud-based system involves network data exchange between signer and signing-enabled cloud environment. This paper proposes a protocol for this data exchange and, as a result, outlines Software as a Service (SaaS) cloud that performs digital signature.

The paper is organized as follows. Section 2 describes some basic requirements for cloud-based digital signature system. Next, in Section 3, the protocol's entities and data flow are analyzed. Section 4 details each step in the protocol. Section 5 is based on the implementation of Signer entity

and covers performance results and implementation notes. Finally, the related work and motivation for future work are discussed at the end of the paper.

II. REQUIREMENTS

Requirements for cloud-based digital signature protocol are associated with the demands for newly designed public-key cryptosystems reported in the literature [4,5,6].

A. Security

Security of cloud-based digital signature system simply refers to the protection of user's private key from being retrieved and/or used without authorization. Each time the private key is restored in the cloud it can be extracted and used outside the system (attack on key). Other threats are related to unauthorized use of the private key inside the system, which may be affected by a modification of data sent for signing (attack on data) or being impersonated online (impersonation attack).

Considering the source of risk to the system's security, we can identify two main groups of threats. The system can be compromised by vulnerabilities in supporting software (including operating system, web browser, web server, database server etc.). This kind of threats can be called indirect because they are not related to the process of cloud signature itself. The affected system may disclose confidential data or allow unauthorized modification to data flow. The ability to protect the system against indirect threats is obviously limited. Therefore, when designing a secure cloud signature system, it is necessary to analyze the effects of a successful attack using vulnerability in supporting software. In such a case, security of user's private keys must be preserved.

The other group of risks is directly related to vulnerabilities in the system's protocols and procedures (direct threats). They may occur in each component of the system and at each stage of the process. In contrast to the indirect risks, a successful attack using the features and characteristics of the protocols and procedures of designed system results in disruption of the signature process and often allows an attacker to compromise private keys restored in the cloud. Therefore, a secure cloud signature system must prove its resistance to direct threats.

When analyzing security of centralized cloud signature system, all the involved protocols and procedures need to be examined to understand the scope of potential attacks. When only a single private key can be compromised, we are talking

about local-scope threats. The attacks which threaten all private keys and any signing process are considered global-scope.

B. Usability

ISO [23] defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" (ISO 9241-11:1998). The emphasis placed on this requirement stems from the belief that current systems do not correspond with modern standards of usability (well known from electronic payment systems and e-banking) and that high usability is always at odds with the requirement of high security level [3,7].

A radical method of achieving high usability is to eliminate dedicated devices for digital signature (microchip cards, card readers) and propose data e-signing as in-cloud service. By transferring processing logic to infrastructure provider (cloud) and providing a simple access interface, the process of digital signature can be reduced to standard authentication and secure data transfer.

C. Cross-platform and integration capabilities

In order for any kind of digital system to be considered cross-platform, it must be able to operate in any hardware and software configuration. Dedicated hardware in conventional digital signing solutions impose mandatory system requirements. It makes porting the system to new platform (e.g., mobile devices) very complicated. It also makes it difficult to integrate digital signing services with other electronic services.

Providing an interface for digital signature services through standard network protocols has multi-platform capabilities at both the hardware and software level. Transfer of processing logic to cloud also offers great opportunities for integration with other electronic services residing in the cloud.

III. PROTOCOL BASICS

We can identify four basic protocol entities:

A. Signer

Signer (User) is the client for signature service, whose private key is restored in the cloud in digital signing process. Considering the complexity of the digital signature process, the system requirements for signer are minimal. They encompass a mobile device with an active SIM card (e.g., phone) and a device with Internet access (e.g., Internet enabled PC with modern web browser). These very basic requirements allow processing regardless of hardware and software platforms. For the mobile device, it means flexibility in terms of architecture and operating system as well as services offered by the mobile operator. For the Internet enabled device, there are no operating system and web browser restrictions. Nevertheless, there are computing power and web browser supported technologies issues related to client-side cryptographic operations. This is discussed in Section 5.

The concept of moving processing to the cloud eliminates the need for dedicated hardware and software. Signer does not have to deal with a microchip card, a card reader and pre-installed software.

B. Issuer

Issuer is an entity that owns or creates data signed by Signer in digital signing process. In this paper, the most basic model is presented, which assumes that Issuer and Signer are the same user. However, it should be noted that more complex models with separation of these roles can be presented. Regardless of role separation, issuing data is also characterized by "cloud-based processing logic". Thus, the system requirement remains the same for both Signer and Issuer.

C. Proxy

Proxy provides the interface for the digital signature service in cloud. The device consists of a single server or a group of servers with software that supports HTTP communications protocol (web server), database management system and dedicated applications. The role of proxy server is reduced to managing and monitoring user access to a hardware security module (HSM) where cryptographic operations of cloud-based digital signature are implemented. Process management includes user's authentication as well as collecting and formatting data sent to the HSM. Proxy also performs monitoring and logging system events.

D. Hardware Security Module (HSM)

HSM is a device with built-in secure cryptoprocessor dedicated to managing cryptographic keys and carrying out cryptographic operations of cloud-based digital signature. The HSM certified by NIST [24] is considered tamper-resistant, which is why the environment of this protocol entity is assumed secure in both the logical and physical layer.

As mentioned earlier, the basic model of cloud-based signature service assumes that Signer signs data he owns. The model describes the interaction of three entities (Signer/Issuer, Proxy and HSM). Signer/Issuer and Proxy communicate with HTTP protocol. In order to provide a higher level of security, this communication should be made over a secure TLS channel. HSM can be connected to Proxy as a built-in device (e.g., PCI device) or reside as a standalone cryptoserver. The detailed configuration of cloud environment (Proxy and HSM) is beyond the scope of this paper.

IV. PROTOCOL DETAILS

User, in addition to unique identifier (name) and password (pass), has a mobile phone with active SIM card and corresponding phone number. This device is used to receive text messages, sent from the signing system, containing the value of one-time password (OTP).

Each user is assigned an asymmetric public-private key pair ($k_{pub}^{user}, k_{priv}^{user}$) representing electronic signature keys. Key

k_{prv}^{user} is used to digitally sign data, which is why its protection is critical from a security point of view.

Hardware security module maintains its own asymmetric key pair $(k_{pub}^{hsm}, k_{prv}^{hsm})$, symmetric key K , the value of OTP_{secret} for the one-time password generation algorithms and implements the following:

- Gen - password-based key derivation function [8],
- Sym^{enc}, Sym^{dec} - encryption and decryption algorithm of symmetric cipher working in Authenticated Encryption (AE) mode [9,10],
- Asym - asymmetric cipher,
- $Sign_{Asym}$ - digital signature algorithm,
- Gen_{OTP} - one-time passwords generator [11,12].

Proxy stores \hat{k}^{user} necessary to restore the user's private key:

$$\hat{k}^{user} = Sym_K^{enc}(Sym_{Gen(pass)}^{enc}(k_{prv}^{user})) \quad (1)$$

In order to sign a document (doc), the following steps are performed:

1. User connects to the Proxy and pre-authenticates. In order to keep the protocol as simple as possible, Signer uses only one password in the system. Although the pre-authentication process is used mainly for phone number identification, it uses the same password that secures user's private key. That's why security requirements for this process should be relaxed, for example, by using collision-rich functions [7]. Another idea is to allow clients to pre-authenticate to servers using zero-knowledge proofs.
2. The server identifies the phone number of the authenticated user and initiates the process of providing one-time code OTP.
3. The user downloads the software, necessary for protocol communication, as a dynamic website. Using the supplied implementation of algorithms User generates:

$$\widehat{doc} = Sym_{Gen(pass||OTP)}^{enc}(doc) \quad (2)$$

$$\widehat{pass} = Asym_{k_{pub}^{hsm}}(pass) \quad (3)$$

and sends (login, $\widehat{pass}, \widehat{doc}$) to Proxy.

4. Proxy forwards ($\widehat{pass}, \widehat{doc}$) dataset received from user together with \hat{k}^{user} suitable for an authenticated user to the security module (HSM).
5. HSM restores:

$$pass = Asym_{k_{prv}^{hsm}}(\widehat{pass}) \quad (4)$$

$$OTP = Gen_{OTP}(k^{user} || OTP_{secret}) \quad (5)$$

$$doc = Sym_{Gen(pass||OTP)}^{dec}(\widehat{doc}) \quad (6)$$

$$k_{prv}^{user} = Sym_{Gen(pass)}^{dec}(Sym_K^{dec}(\hat{k}^{user})) \quad (7)$$

- As the algorithm Sym^{dec} operates in AE mode, operation (6) confirms the integrity and authenticity of the document and verifies the one-time password. Similarly, operation (7) also authenticates User by verifying (pass).
6. Security module (HSM) signs a document using the user's private key k_{prv}^{user}

$$doc_{sign} = Sign_{k_{prv}^{user}}(doc) \quad (8)$$

Fig. 1 depicts a detailed view of the protocol flow by describing the sequence of actions in a process. The key features can be summarized as follows:

- Independent proofs. Security of the user's private key relies on two independent proofs of identity: something the user has (registered SIM card and the phone receiving one-time passwords) and something the user knows (password).
- 'Sole control'. The private key remains under the user's 'sole control'. Key data is encrypted with password known only by Signer. It is impossible to restore even by the service provider. The only person who can do that is Signer. The concept of 'sole control' is discussed in detail in [14].
- Security functions in HSM. All main security functions are moved to a secure environment of Hardware Security Module. Outside the HSM private keys and data to be signed are always encrypted. Verification of independent proofs (password and one-time password) is also implemented in HSM by using a symmetric cipher in AE mode.
- High usability level. From Signer's point of view digital signature process has been reduced to standard authentication and secure data transfer (see Fig. 1). Signer does not need any dedicated devices for digital signature.
- Event logging. Proxy can be used as an event logger in the system, which meets the requirement to include generating digital signature into the security process of public key infrastructure (PKI) pointed out in [6].

V. SIGNER ENTITY IMPLEMENTATION NOTES

As mentioned earlier, there are some implementation issues related to client-side cryptographic operations that must be analyzed in order to estimate the additional computational overhead of the proposed protocol when comparing to basic server-side digital signature protocol, with no client-side encryption (e.g., one proposed in [16]).

First of all, the client-side cryptographic operations, performed in step 3 of the protocol, are executed transparently in browser environment and will probably be implemented in JavaScript. Most web programmers agree that the biggest challenge in web design lies in dealing with the variety of browsers. While the majority of active page elements are reliably rendered in most browsers, each browser has its own quirks when it comes to the implementation of JavaScript engine. This might cause

different overhead for the same machine when performing cryptographic computation in different browsers. Secondly, client-side data encryption requires loading local files. Such feature is not supported by older browsers. A standard way to interact with local files was introduced in HTML5 specification, so an up-to-date, HTML5-enabled browser is required to interact in the protocol. Although this entails additional restrictions, the need to use an up-to-date browser also meets the security requirements mentioned in Section 3.

Further notes are based on Signer entity implementation, prepared as dynamic HTML page with SJCL library for cryptography in JavaScript [20]. For asymmetric encryption 256-bit ElGamal ECC was used. Symmetric encryption is performed with 128-bit AES in CCM mode. Table I shows the average execution time for step 3 (see Section 4) for different sizes in different browsers.

It has been observed that performing symmetric encryption on larger files causes browser to freeze. This behavior is unacceptable in terms of usability. To avoid this, larger files should be split into smaller parts and encrypted separately. When choosing the size of file splitter the following factors must be taken into consideration. Still, encryption of large file parts might cause the browser to freeze on older machines. Small file parts increase the number of iterations in encryption loop, which influences overall performance.

Table II shows the average execution time of encrypting 10 MB file with different splitter size. The test was performed on two different computers with high and low computing power, respectively.

In addition to computation overhead, there is also the additional download size of required scripts. Using well-known optimization techniques this size can be reduced to approximately 50kB, which is negligible from the user's point of view.

VI. RELATED WORK

A secure digital signature creation environment, based on mobile devices and smart cards, is defined and analyzed by A. Mana et al. [15]. Storing private key on signer's SIM card is proposed by H. Rossnagel [16]. A more server-side approach with encrypted private keys is presented by M. Centner et al. [17]. The same authors in [18] designed a digital signature service based on smartcard-reader middleware as a Java applet. A proof-of-concept prototype of this approach has been implemented as a web-based signing service. A signing scheme for thin clients, with server based processing is presented by Y. Lei et al. [13]. J. Anderson et al. [7] proposes a protocol, which allows users to store secrets, such as private keys, in the cloud, using the services of several key recovery agents.

On-going work on novel signing service schemes is also related to European Commission's mandate M/460. The UE standardization platform is prepared by two European standardization organizations, CEN [25] and ETSI [26]. In [19], the Commission indicates new perspectives and challenges for the platform. Many of them (e.g., cross-border compliance) can be implemented with cloud-based processing logic.

TABLE I. AVERAGE EXECUTION TIME FOR DIFFERENT DOC SIZES IN DIFFERENT BROWSERS

File size	Execution time(ms)		
	Chrome	Firefox	IE
100kB	688	344	186
200kB	814	392	245
500kB	1186	559	422
1MB	1521	820	688
10MB	11183	5825	5188
20MB	23634	11564	9932

TABLE II. AVERAGE EXECUTION TIME FOR DIFFERENT DOC SIZES IN DIFFERENT BROWSERS

Splitter size	Execution time(ms)	
	Computer 1	Computer 2
100kB	6246	15319
500kB	5955	14452
1MB	5884	13747
5MB	5673	freeze

Things to consider when moving digital signature model, or, more general Public Key Infrastructure into cloud are addressed by H. Kharche et al. [4]. Brown and Robinson [5] show how existing security protocols (like TLS) can derive from cloud computing. Important cloud-specific security issues are also pointed out by R. Chow et al. [22].

VII. CONCLUSION AND FUTURE WORK

The proposed cloud-based digital signature protocol meets the usability and cross-platform requirements laid down in Section 2. Although the protocol was designed taking into account the security requirements, future studies are required in order to prove its security.

As the proposed protocol is mainly focused on signer-cloud communication, further studies are required to show how such digital signature model can exploit cloud benefits. Moreover, the protocol can be extended to handle more complex models (e.g., with Signer and Issuer role separation). Advanced digital signature services can be also developed based on the proposed protocol (e.g., Forward-Time Public Key proposed in [21]).

The cloud-based digital signature can also be analyzed for compliance with law and regulations of the qualified electronic signature. When it comes to EU regulations, similar studies are presented by M. Centner et al. [17].

REFERENCES

- [1] P. Mella and T. Grance, "The NIST Definition of Cloud Computing". Special Publication 800-145, NIST, Sep. 2011.
- [2] Security requirements for cryptographic modules, FIPS PUB 140-2, NIST, Dec. 2002.
- [3] D. Davis, "Compliance Defects in Public-Key Cryptography", Proc. 6th Usenix Security Symp., Jul. 1996, pp.171-178.

[4] H. Kharche and D. S. Chouhan, "Building Trust In Cloud Using Public Key Infrastructure -A step towards cloud trust", International Journal of Advanced Computer Science and Applications, vol. 3, no. 3, Mar. 2012, pp. 26-31.

[5] J. Brown and P. Robinson, "PKI Reborn in the Cloud", conference slides, RSA Conference Europe, Oct. 2011, <http://365.rsaconference.com/docs/DOC-3037> [retrieved: March 2013].

[6] C. Ellison and B. Schneier, "Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure", Computer Security Journal, vol. 16, no. 1, 2000, pp. 1-7.

[7] J. Anderson, F. Stajano, "On Storing Private Keys 'In the Cloud' Extended Abstract", unpublished, <http://www.cl.cam.ac.uk/~jra40/publications/2010-SPW-key-storage.pdf> [retrieved: March 2013].

[8] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, IETF, Sep. 2000.

[9] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, IETF, Sep. 2003.

[10] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption", ACM Transactions on Information and System Security (TISSEC), vol. 6, no. 3, Feb. 2003, pp. 365-403.

[11] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, IETF, May 2011.

[12] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, IETF, Dec. 2005.

[13] Y. Lei, D. Chen, and Z. Jiang, "Generating Digital Signatures on Mobile Devices", Proc. 18th International Conference on Advanced Information Networking and Applications, Mar. 2004, pp. 532-536.

[14] Public Statement on Server Based Signature Services (Forum of European Supervisory Authorities for Electronic Signatures), Forum of European Supervisory Authorities for Electronic Signatures (FESA), October 2005, <http://www.fesa.eu/public-documents/PublicStatement-ServerBasedSignatureServices-20051027.pdf> [retrieved: March 2013].

[15] A. Mana and S. Matamoros, "Practical Mobile Digital Signatures", Proc. EC-WEB '02 Proceedings of the Third International Conference on E-Commerce and Web Technologies, Sep. 2002, pp.224-233.

[16] H. Rossnagel, "Mobile Qualified Electronic Signatures and Certification on Demand", Proc. 1st European PKI Workshop Research and Applications, Jun. 2004, pp.274-286.

[17] M. Centner, C. Orthacker, and C. Kittl, "Qualified Mobile Server Signature", Proc. 25th International Information Security Conference, Sep. 2010, pp. 103-111.

[18] M. Centner, C. Orthacker and W. Bauer, "Minimal-footprint Middleware for the Creation of Qualified Signatures", Proc. WEBIST 2010 International Conference on Web Information Systems and Technologies, Apr. 2010, pp. 64-69.

[19] Proposal for a regulation of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market, Commission staff working paper, Jun. 2012.

[20] E. Stark, M. Hamburg, and D. Boneh, "Symmetric cryptography in javascript", Proc. ACSAC '09 Annual Computer Security Applications Conference, Dec. 2009, pp.373-381.

[21] J. Riordan and B. Schneier, "Environmental Key Generation towards Clueless Agents. Mobile Agents and Security", G. Vigna, ed., Springer-Verlag, 1998, pp. 15-24.

[22] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing, computation without outsourcing control", Proc. 2009 ACM workshop on Cloud computing security, Nov. 2009, pp.85-90

[23] International Organization for Standardization, <http://www.iso.org> [retrieved: March 2013]

[24] National Institute of Standards and Technology, <http://www.nist.gov> [retrieved: March 2013]

[25] European Committee for Standardization, <http://www.cen.eu> [retrieved: March 2013].

[26] European Telecommunications Standards Institute, <http://www.etsi.org> [retrieved: March 2013].

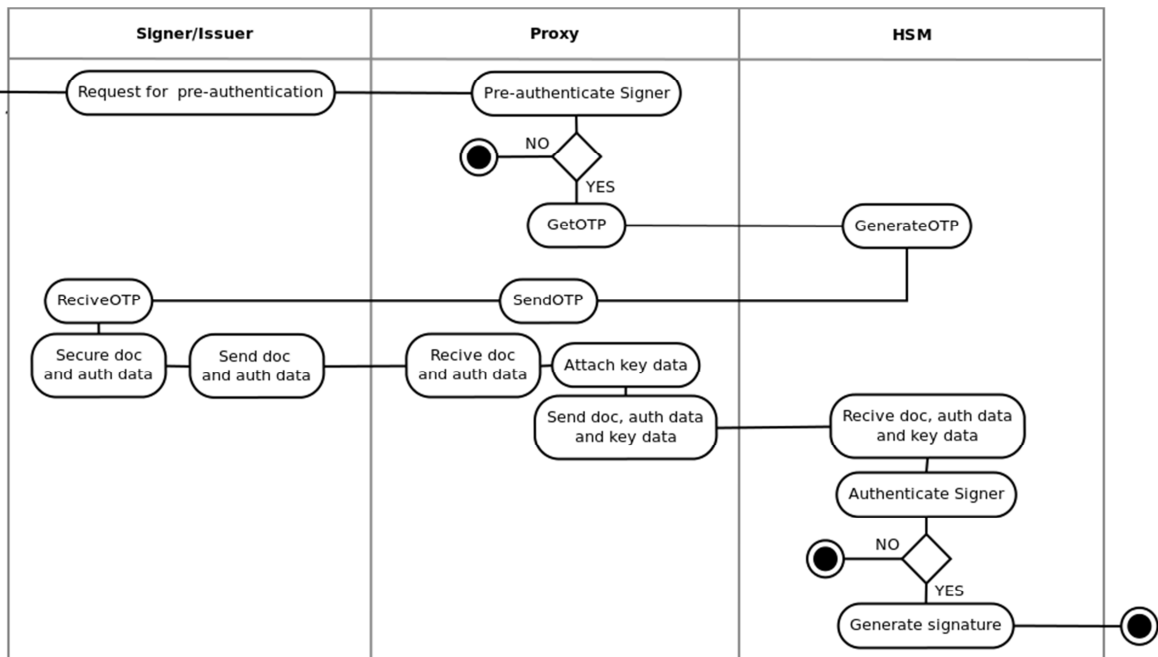


Figure 1. UML activity diagram for cloud-based digital signature protocol.

Cloud-Enabled Scaling of Event Processing Applications

Irina Astrova
Institute of Cybernetics
Tallinn University of Technology

Tallinn, Estonia
 irina@cs.ioc.ee

Arne Koschel
Faculty IV, Department for Computer Science
University of Applied Sciences and Arts
Hannover

Hannover, Germany
 akoschel@acm.org

Ahto Kalja
Institute of Cybernetics
Tallinn University of Technology

Tallinn, Estonia
 ahto@cs.ioc.ee

Abstract—Event processing is an important established concept for event-driven system development – with database triggers and event processing engines being typical examples of event processing technology. With nowadays movement into cloud computing, highly flexible scalability in cloud environments becomes an important challenge for event processing applications as they have many event sources and events to be processed there. As the core contribution of our work, we propose a novel approach to providing event processing applications with cloud-enabled scalability transparently to users (viz., the application developers) as part of an event-driven system itself.

Keywords—*Infrastructure-as-a-Service (IaaS) clouds; IaaS scalability; event processing applications; agents; event-driven systems.*

I. INTRODUCTION

Traditional applications execute in a sequential way. But the real world is driven by events, which can come from several event sources. So how can these events be caught by traditional applications? One can create threads, which run in loops to catch the events and dispatch them to event consumers that can perform actions in response to the events. The biggest problem with this approach is that the applications can waste a lot of resources with otherwise not needed loops. Another big problem is an increased time between the raise of the events and their catch. Event processing applications provide a solution to these problems.

Event processing applications can be defined as sense-and-respond applications, i.e., the applications that can react to and process events. An event processing application can play the role of an event source, an event sink, or both. Event sources can handle off events to event sinks. It should be noted that an event source does not necessarily generate an event, nor an event sink is necessarily an event consumer. Furthermore, event sources and event sinks can be completely decoupled from each other: one can add and remove event sources and event sinks as needed without impacting other event sources and event sinks.

Event processing applications use the following concepts:

- ▲ **Event:** In an event processing application, every event is represented as an event object. This object holds all information about the event such as the timestamp when the event was caught, the event type, the event source, etc. After the catch of an

event and transforming it to an application object, it is handed to an event stream.

- ▲ **Event stream:** An event stream is like a FIFO (First In, First Out) queue. Application objects in the stream are handled sequentially in the order of their arrival. The speciality of this type of queue is that an agent can subscribe to the stream and select which events it wants to receive.
- ▲ **Agent:** The drivers of an event processing application are one or more agents. They get the events from an event stream and react to or operate on those events. Examples of operations on events: selection, aggregation and composition. To structure agents and create a high cohesion with loose connections, an event processing network is used.
- ▲ **Event Processing Network (EPN):** An EPN models an event processing application as a set of interconnected application components whose execution is driven by events. Therefore, it is typically represented as a directed graph, where events are flowing through edges into nodes, which in their turn represent application components.
- ▲ **Event channel:** This is typically a messaging backbone, which transports the (formatted) events between event sources and event sinks. Because of the variety of event sources, not all events will be created in the format required for processing them by agents. In those cases, the events need to be formatted prior to being deposited them in an event channel.

Next we are presenting an example of event processing applications. This example is a door access log into a company, which uses a radio frequency identification (RFID) transponder to control the work time of its employees.

1. Employee A comes to work and activates the RFID transponder at the door with his chip, thus generating an access event.
2. The information on the chip is scanned and given to the adapter of an event processing application.
3. The application creates an event object and injects the data into it.
4. With a bundle of the subscriptions, the application knows which agents are interested on this event type (say Agents A and B) and put the event into the agent's event streams.
5. Agent A only reacts to the access event and logs the timestamp of the event and the information on the employee's chip to a database.

6. Agent B waits for another access event by the same employee in a time window of 10 hours.
7. Employee A activates the RFID transponder at the door with his chip again, when leaving work.
8. The application creates an another event object with the information on the employee's chip and passes it to the agents.
9. Agent A logs this event to the database.
10. Between the first and the second access events, Agent B produces a new event with the time which has passed between them.
11. Due to some other subscription, Agent B knows another agent, say Agent C, which is interested in the new event because it needs to gain the employee's work time out of it.

Step 6 shows how the agent uses a selection operation to get the information it needs. In this case, the agent also uses a technique, which is called windowing. It is possible to define a window by time (as it is in the example) or by the number of events in an event stream. Step 10 is an example of the composition of events. Here two events are merged into a new one. Once the new event has been composed, any agent in the application can use that event.

The remainder of this paper is organized as follows. The next section gives the motivation for our approach. This is followed by a description of our approach and a brief overview of the work related to the combination of event processing and cloud computing. The final section concludes the paper.

II. MOTIVATION

Event processing applications are important because the real world is event-driven [12]. With great demand on high-speed and cost-efficient processing of events, event processing applications are calling for IaaS (Infrastructure-as-a-Service) scalability. IaaS scalability lets the applications make optimum utilization of resources such as CPU and RAM at different workload levels in order to avoid over-provisioning (i.e., having too many resources), under-utilization (i.e., not using resources adequately) and under-provisioning (i.e., having too few resources) [1]. In traditional environments, over-provisioning and under-utilization can hardly be avoided [2]. There is an observation that in many companies the average utilization of servers ranges from 5 to 20 percent, meaning that many resources are idle at no-peak times [3]. On the other hand, if the companies shrink their infrastructures to reduce over-provisioning and under-utilization, the risk of under-provisioning will increase. While the costs of over-provisioning and under-utilization can easily be calculated, the costs of under-provisioning are more difficult to calculate because under-provisioning can lead to a loss of users and zero revenues [3].

Since event processing applications experience variability in utilization of resources, they are calling for an infrastructure that can dynamically scale according to the application demand. IaaS scalability is one of the major advantages offered by IaaS clouds. This gives rise to the idea

to deploy event processing applications into IaaS clouds. However, IaaS scalability is not just about having a scalable (virtual) infrastructure, but also about writing scalable applications. Valuable rules of thumb have been provided by Amazon.

Amazon provides a best practices guide [4] on how to write applications for the best fit for IaaS clouds. The most important guidelines are: an application should be divided into loosely coupled components that can be distributed across several servers and executed in parallel. Furthermore, the application should be as stateless as possible. If an application component fails or is temporarily not available, the application should continue to run. This can be achieved by developing the component as self-rebooting and using a message queue [5]. If the component is temporarily not available, messages will be stored in the queue and delivered later when the component comes alive again. These rules clearly indicate that IaaS scalability depends on the application design as well as the communication mechanism used to implement the application components. Therefore, IaaS scalability cannot be achieved by simply deploying applications into IaaS clouds. Rather, an IaaS cloud can guarantee an infrastructure equal to the application demand only when applications are designed properly or their design is amenable to appropriate scaling (horizontal or vertical).

However, event processing applications typically rely on a centralized event coordinator and could easily become a scalability bottleneck as a result of that [11]. Event processing applications are inherently stateful, which implies that services cannot be migrated or located anywhere, without affecting the application performance. Therefore, the deployment of event processing applications to IaaS clouds typically requires redesigning the applications for leveraging on-demand resource utilization. Therefore, the biggest problem is how to minimize the changes need to be done to the application design.

Another big problem is how to scale EPNs in IaaS clouds. An event-driven application can specify an EPN, which assembles the other components (e.g., event sources, event sinks and event streams) together. Virtual machines in IaaS clouds can scale horizontally by cloning a virtual machine or vertically by adding more resources to a virtual machine. Besides the scaling of virtual machines, the virtualization technologies inherent to IaaS clouds allow for the scaling of EPNs. Unfortunately, this very desirable feature is not supported by IaaS clouds yet, thus further complicating the deployment of event processing applications into IaaS clouds.

As an attempt to solve the problems above, in our previous work [6][10] we proposed to make event processing applications scalable through the integration of an event processing engine into a cloud architecture. In this paper, however, we propose a different approach.

III. OUR APPROACH

IaaS scalability is important for event processing applications because these applications experience variability in resource utilization. Therefore, our approach is aimed at

providing event processing applications with IaaS scalability. IaaS scalability is service-oriented, meaning that scaling decisions are made on the basis of infrastructural metrics such as CPU and RAM utilization [1].

The basic idea behind of our approach was to bring IaaS scalability into an event-driven system itself. An event-driven system can generally be comprised of several event sources, event processing applications and event sinks. Event sinks have the responsibility of applying a reaction as soon as an event occurs. The reaction might or might not be completely provided by the sink itself. For example, the sink might just have the responsibility to filter, transform and forward the event to another component or it might provide a self-contained reaction on such an event.

Event sources, event processing applications and event sinks can be decoupled of each other; one can add or remove any of these components without causing changes to the others. However, an event-driven system could get quiet complex due to a large number of agents and event sinks to synthesize events out of aggregated data. Moreover, the agents are independent of each other – they can be distributed across several servers and executed in parallel. The problem is that it is very difficult for a scaling mechanism to decide which agents should use which rules to produce which output. Also how could this decision be made when the cloud should scale itself? Therefore, it was not an easy task to bring IaaS scalability into an event-driven system.

Figure 1 gives an overview of our approach, which includes the following components:

- ⤴ **Load Balancing Agent (LBA):** Each EPN has its own LBA monitoring and interpreting (internal) technical events occurring in an Event-Processing-as-a-Service cloud and their data. LBAs ensure the performance and the availability of each EPN (or its agents), as they are the ones, which perceive the need to provision or decommission resources. Scaling decisions are made by LBAs on the basis of the current resource utilization and calculated by LBAs themselves. The resource utilization is aggregated out of technical events. For example, if the minimum or maximum threshold is crossed, scaling rules will be fired and a scaling mechanism will kick in.
- ⤴ **Scaling Agent (ScA):** In addition to the LBA, each EPN has its own ScA, which can clone the EPN for horizontal scaling or restart it on a bigger virtual machine for vertical scaling.
- ⤴ **Central Scaling Agent (CScA):** The CScA evaluates technical events against scaling rules. Scaling actions may include, e.g., the invocation of a service or the triggering of a scaling process. In addition, the CScA maintains the EPN topology.

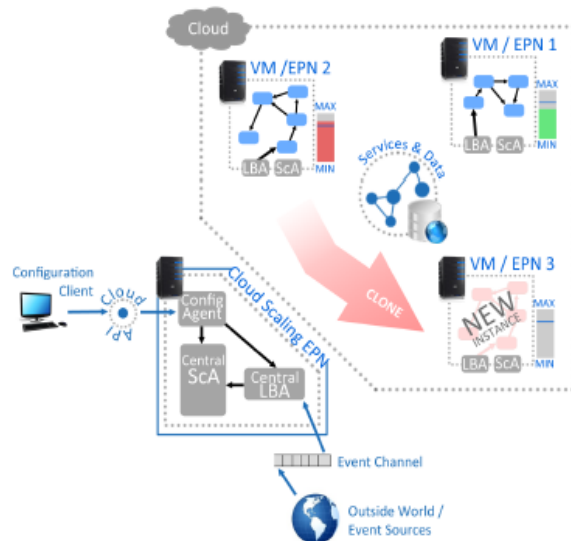


Figure 1. Cloud-enabled scaling of event processing applications

- ⤴ **Central Load Balancing Agent (CLBA):** If the CScA defines how to scale, the CLBA defines what to scale. The CLBA takes the load of each EPN into account. Each LBA has to periodically send the information on the current resource utilization of its EPN to the CLBA. The CLBA then instructs the CScA to provision or decommission resources. This allows the CScA to foresee critical situations and to make scaling decisions beforehand. The CLBA is also responsible for all external events. An exposed interface (e.g., web services) make the interaction between the outside world and the cloud possible.
- ⤴ **Configuration Agent (CA):** The CA allows for the configuration of the whole scaling mechanism (e.g., scaling rules and thresholds) and the EPN topology through the cloud API. The CA could be implemented as an agent fitting into the idea of a Dynamic Control Plane [7], which gives users (viz., the application developers) the possibility to configure the cloud through an easy-to-use administrative interface.
- ⤴ **Cloud-Scaling EPN:** The CLBA, the CScA and the CA are “networked” together to form an EPN for the scaling of an Event-Processing-as-a-Service cloud. This cloud hosts services to be leveraged by event processing applications as needed. As a result, the cloud can scale up and down according to the application demand.

Our architecture can be used by the following event processing applications:

- ⤴ Disaster management, where the input data need to be gathered from various heterogeneous distributed sources (e.g., scientific sensors) and processed using the event processing technology to react on disasters.

- ▲ Online business development, where the clicks of website visitors need to be processed as events to identify the interest to the website.

IV. RELATED WORK

Technical events occurring in an IaaS cloud are related to resource utilization. Event processing engines can help in monitoring and high-speed processing of these events. Therefore, recently it was proposed to integrate an event processing engine into an elastic controller in order to enhance IaaS scalability [8][9].

An IaaS cloud requires that applications are designed especially for the cloud. The scaling of traditional applications is typically easy. The question is how to scale event processing applications. These applications follow their own design rules and thus, they have to be tailored to the cloud. Therefore, in our previous work [6][10] we proposed to integrate an event processing engine into a cloud architecture itself, providing scaling decisions out of scaling rules through the cloud API.

However, in this paper we decided to move from a different direction – we tried to adapt IaaS scalability to an event-driven system.

V. CONCLUSION AND FUTURE WORK

Event processing applications need to handle a lot of information. Thus, the ability to process this information quickly is important for those applications. But processing the information quickly implies processing it efficiently, which in turn implies spending less money on an infrastructure. And this is the point where event processing applications could benefit from the deployment into IaaS clouds whose scalability enables efficient and cost-saving event processing. However, a cloud architecture that allows event processing applications to benefit from IaaS scalability is currently missing [6][10]. Therefore, with our approach and its components described below, we aim to fill this gap.

Each EPN will have a Load Balancing Agent (LBA), which periodically sends the load of its EPN to the Central Load Balancing Agent (CLBA). If the minimum or maximum thresholds specified by users through the Configuration Agent (CA) are crossed, the CLBA will instruct the Central Scaling Agent (CScA) to provision or decommission resources. In addition to the LBA, each EPN will have a Scaling Agent (ScA) acting on behalf of the CScA. The CScA will translate the CLBA's instructions into an appropriate scaling action taken by the ScA to adjust the load of its EPN. It should be noted that users will be kept totally unaware of these scaling actions and delivered with the illusion of a scalable infrastructure, the infrastructure that can scale horizontally (by cloning an EPN) or vertically (by restarting an EPN on a bigger virtual machine).

Our approach is geared to make event processing applications scalable, while minimizing changes to be done

to the application design and allowing for the scaling of EPNs as if they were virtual machines.

In the future, we are going to implement our approach and evaluate its performance.

ACKNOWLEDGMENT

Irina Astrova's and Ahto Kalja's work was supported by the Estonian Centre of Excellence in Computer Science (EXCS) funded mainly by the European Regional Development Fund (ERDF). Irina Astrova's and Ahto Kalja's work was also supported by the Estonian Ministry of Education and Research target-financed research theme no. 0140007s12.

REFERENCES

- [1] J. Cáceres, L. Vaquero, L. Rodero-Merino, Á. Polo, and J. Hierro. Service scalability over the cloud, Handbook of Cloud Computing, eds. B. Furht and A. Escalante, Springer Verlag, Berlin, Heidelberg, 2010
- [2] C. Braun, M. Kunze, J. Nimis, and S. Tai. Cloud Computing, Web-based dynamic IT-Services. Springer Verlag, Berlin, Heidelberg, 2010
- [3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. Communications of the ACM, 53(4), 2010, pp. 50–58
- [4] J. Varia. Architecting for the cloud: best practices. last accessed: January 2013, http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf
- [5] P. Marshall, K. Keahey, and T. Freeman. Elastic site: Using clouds to elastically extend site resources, Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, IEEE, 2010, pp. 43–52
- [6] I. Astrova, A. Koschel, and M. Schaaf. Automatic scaling of complex event processing applications in Eucalyptus. Proceedings of the 15th IEEE International Conference on Computational Science and Engineering (CSE), IEEE, 2012, pp. 22–29
- [7] L. MacVittie, A. Murphy, P. Silva, and K. Salchow. Herscheruber die wolke: Anforderungen an cloud-computing. Technical report, 2010
- [8] H. Lim, S. Babu, and J. Chase. Automated control for elastic storage, Duke University, 2010
- [9] L. Vaquero, L. Rodero-Merino, and R. Buyya. Dynamically scaling applications in the cloud. ACM SIGCOMM Computer Communication Review, 41, 2011, pp. 45–52
- [10] A. Koschel, I. Astrova, M. Schaaf, S. Gatzju Grivas, S. Priebe, J. Raczek, J. Reehuis, and K. Scherer. Integrating complex event processing into Eucalyptus, Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2011
- [11] N. Shalom. Interview with Michael Di Stefano from Integrasoft on their complex event processing cloud services using Esper GigaSpaces. last accessed: January 2013, <http://blog.gigaspace.com/interview-with-michael-di-stefano-from-integrasoft-on-their-cep-cloud-services-using-esper-gigaspace/>
- [12] M. Schaaf, A. Koschel, S. Gatzju Grivas, and I. Astrova. An active DBMS style activity service for cloud environments. Proceedings of the 1st International Conference on Cloud Computing, GRIDs, and Virtualization, IARIA, 2010, 80–85

Cloud Computing Services Potential Analysis

An integrated model for evaluating Software as a Service

Giuseppe Ercolani

Universidad de Murcia - Facultad de Comunicación y Documentación
Campus Universitario de Espinardo - 30100 - Murcia, Spain
e-mail: giuseppe.ercolani@um.es

Abstract— This paper address, in a practical and integrated model, a possible solution of issues concerning the Software as a Service (SaaS) introduction and evaluation. A selective top-down analysis is proposed to guide the overall assessment. The construction of the Potential Adoption Index (PAI), in the last stage of the process, aims to facilitate the evaluation and comparison process of the acceptance of this technology by evaluating: the functional requirements, the total cost of ownership (TCO) and the related concerns and benefits from technical and business perspective.

Keywords-Cloud Computing; SaaS; adoption; evaluation

I. INTRODUCTION

In recent years, the term cloud computing has been used to identify an evolution paradigm in the computer industry.

This refers to a set of advanced technologies that affect the focus of the organizations and businesses in plan, management and use of its technology infrastructure in the near future.

As noted by [1] the base of the cloud computing is the evolution of three phenomena: virtualization, grid computing and web services

The increasing bandwidth availability of Internet connection and the accessibility from mobile and portable devices has encouraged the spread of applications created for this environment and the access to available resources exclusively through internet (both often offered free of charge for a basic or private usage).

In this way everyone can connect to a website with a browser, fill out a form to access the service, select the available options, the most convenient form of payment and start working with the program or service contracted, without requiring a server, Information Technology (IT) staff, software licenses, installing applications or arrange a backups strategy.

Still, despite the availability of SaaS solution, the main question is that if it is convenient to adopt a solution based on cloud computing.

This article examines the pros and cons described in scientific literature and the potentials that this form of computing may have inside an enterprise.

The topic of this research may be found in the line of "Technology adoption and implementation research" in the business-technology framework defined by [2].

An integrated model for calculating the Potential Adoption Index (PAI) will be presented in order to quantify the benefits and disadvantages of cloud computing adoption.

The PAI, which includes the evaluation of features, benefits and concerns from the business perspective and the technical fit from cloud experts' viewpoint, indicates the overall adoption utility level.

The structure of the paper is as follows. Firstly, the term cloud computing will be referenced. Then an integrated model analysis is presented in three stages. The computation of the PAI synthesizing, in a numerical result, benefits and disadvantages of adopting a cloud computing SaaS solution. A numerical example is presented in order to explain the construction of the PAI and the interrelation between the different elements of the model.

II. TOP-DOWN SELECTIVE ANALYSIS

There are a multitude of definitions of cloud computing as in [2], [3], and [4] the one taken as a reference for this paper is the one proposed by [5].

The proposed evaluation model for the development of the PAI, consists of three consecutive steps, related to each other, in order to consider:

- A. *a functional analysis, that explores the features of the SaaS solution the company would like to implement/ deploy or integrate;*
- B. *an economic analysis, that quantifies the costs of implementation and maintenance;*
- C. *an attribute analysis, that evaluates characteristics, benefits and concern also needed for the PAI calculation.*

The top down process, guide the reader from a rough overall assessment to a more defined quantification of the analytical aspects of the SaaS analyzed.

A. *Functional analysis: identifying cloud candidate*

Primarily the company must decide which business functions want to move into the cloud and the type of data that will be stored and shared in order to be able to monitor, develop, analyze and use these actions for their growth, implementing services that meet business needs.

If there is at least one alternative to consider in order to facilitate the comparison between different products "white

papers”, reviews or comparisons papers, that examine the functional features of the programs offered by the providers could be downloaded.

In addition, most of the SaaS providers, normally, offer full product evaluation for a limited period of time which facilitates the analysis and comparison without any installation or additional cost. Other providers allow some interaction with the support staff, via email, chat or phone for free. This helps the evaluation of the products offered and to clarify the software functionalities.

For an adequacy SaaS selection, the software functionalities are first inspected because some of the features of Cloud Computing do not lend themselves to an easy customization [3], [6], and multitenant application customization should be made through configuration [7].

In this stage, the suggested methodology (see Table I) include to:

- Identify one or more SaaS solutions available on the market for the specific aspect that the company would like to implement (Collaboration, CRM, ERP, SFA, etc.);
- Identify the functionalities that are required for the company, tagging each of them with a label “Required”, “Nice to have” and “Not required”;
- After having investigated the specific functionality in any specific SaaS or using professional experts for the specific software, mark each of the functionality attribute with a tag indicating if it is available:
 - “Yes”;
 - “Yes but need to be configured”;
 - “Yes can be customized”;
 - “Not Available”

Candidates for the functional evaluation are key users, managers-owners and product experts.

TABLE I. FUNCTIONAL SUITABILITY TABLE IN SUPPORT OF FUNCTIONAL ANALYSIS WITH QUALITATIVE ORDINAL SCALE

Software Functionality	Is it Available?	Is the SaaS Suitable? (Qualitative ordinal scale)
REQUIRED	YES	SUITABLE
	CONFIGURABLE	SUITABLE but +COST / +TIME for the implementation
	CUSTOMIZABLE	SUITABLE but +++COST / +++TIME for the implementation
	NO	NOT SUITABLE: REJECT
NICE TO HAVE	YES	SUITABLE: no cost for an nice to have functionality
	CONFIGURABLE	SUITABLE but +COST / +TIME if implemented
	CUSTOMIZABLE	SUITABLE but +++COST / +++TIME if implemented
	NO	SUITABLE but not implementable if will became a requirement
NOT REQUIRED	YES	SUITABLE: no cost if this functionality will became a requirement
	CONFIGURABLE	SUITABLE but +COST / +TIME if will became a requirement and implemented
	CUSTOMIZABLE	SUITABLE but +++COST / +++TIME if will became a requirement and implemented
	NO	SUITABLE but not implementable if will became a requirement

Each level of suitability will have an immediate repercussion at this stage (e.g., if a required functionality is not available imply the rejection of the SaaS solution) or in will be penalized/rewarded in the further assessment (e.g., in the economic analysis with integration and customization costs evaluation).

The main purpose of this process is to verify the overall fit of the analyzed SaaS package to meet the functional requirements needed by the company.

This relatively simple process helps formal selection of a SaaS without any deep or wide expertise in Cloud Computing. This aims at reducing the number of candidates by selecting some of them very quickly based on a brief review of key functionality and company needs. This also keeps additional costs to a minimum level and stays in line with specific characteristics of cloud computing.

At the end of the first stage will be one or more programs offered as SaaS that should be at least functionally compatible with the essential requirements of the company, with an explicit level of integration and customization required for its adoption. The essential requirement identified, which will need configuration or customization, will be economically estimated in the next phase.

B. Economic analysis: identifying the costs

An assessment could be obtained using the Total Cost of Ownership (TCO) formulation proposed by [8] based on the combination of three costs types, in order to determine the financial impact of SaaS adoption.

Alternatively the proposed TCO method in [9], where a mathematical modeling of cost types is introduced along with a case study, could be used for the same purpose.

The identified economic values will be evaluated within the financial dimension in the next phase (attribute analysis).

C. Attribute analysis and the Potential Adoption Index (PAI) calculation

The evaluation of attributes as features, benefits and risks associated with a SaaS solution will be analyzed in order to calculate the "Potential Adoption Index" (PAI).

In order to determine the SaaS benefits and concern the taxonomies proposed in [10] has been used to generate an evaluation matrix:

- the three main dimensions of cloud related benefits (deployment advantages, financial savings, and functional aspects) have been integrated with the main cloud characteristics (on demand self-services, broad network access, resource polling, rapid elasticity, measured services) in Table II;
- the three main dimensions of cloud related concerns (alignment with existing operating model in organization, management and control of organizational data and services, and legal aspects) have been incorporated in Table III.

For each dimension, underlying category and attributes has been specified to improve the level of detail to offer a more analytic evaluation.

In Table II are exposed the essential characteristics of cloud computing, with cloud deployment, financial and functional benefits.

TABLE II. ASSESSING KEY FEATURE AND MAJOR BENEFITS ASSOCIATED WITH THE CLOUD

Cloud characteristics and benefits		WEIGHT	RATING	WEIGHT * RATING
Essential Characteristics	On-demand self-service	0,013	3	0,039
	Broad network access	0,013	3	0,039
	Resource pooling	0,025	3	0,075
	Rapid elasticity	0,025	4	0,100
	Measured service	0,005	4	0,020
deployment	similarity with other technology already used in the company (ie. Outsourcing)	0,006	1	0,006
	ease to setup	0,006	3	0,018
	ease to maintain	0,006	2	0,012
	speed - implementation time	0,006	4	0,024
	structuring of payment			
	contract payment terms (monthly...)	0,018	2	0,036
	change of subscription fee (end of contract, anytime)	0,019	4	0,076
	penalty on early termination	0,025	4	0,100
	data return on subscription cancel	0,076	4	0,304
	cost scalability (per user, group)	0,013	2	0,026
Benefit	pay-for-use			
	Total cost per year	0,025	3	0,075
	small capital expense	0,025	2	0,050
	convert capex to opex	0,025	2	0,050
	(saving)			
	personnel	0,000		0,000
	hardware	0,000		0,000
	infrastructure	0,000		0,000
	maintenance (update/upgrade)	0,000		0,000
	energy	0,000		0,000
	management	0,000		0,000
	other services			
	provide user training	0,025	3	0,075
	training charges fee	0,025	2	0,050
	self support /documentation	0,025	2	0,050
customer support by phone	0,025	3	0,075	
customer support by email	0,025	3	0,075	
customer support web-ticket	0,025	3	0,075	
Client manager (primary contact)	0,025	2	0,050	
business consulting	0,013	2	0,026	
functional	up to date			
	planned frequency	0,005	1	0,005
	policy to notify update/upgrade	0,013	3	0,039
	expansion (new modules deployment)	0,013	2	0,026
	evolution	0,013	2	0,026
TOTAL BENEFIT		0,563		1,622

The financial dimension in the category saving will be used if the SaaS solution replaces a non-cloud application (dismissal) with identifiable economies or if a different deployment method (other than public cloud) is adopted in order to evaluate the economies from moving some of the company resources (IT staff, hardware, infrastructure, maintenance, energy, management) in the cloud provider domain.

Table III includes the list of concerns attributes evidenced when implementing a cloud computing solution.

TABLE III. ASSESSING KEY FEATURE AND MAJOR BENEFITS ASSOCIATED WITH THE CLOUD

Cloud related concerns		WEIGHT	RATING	WEIGHT * RATING
Concerns	integration			
	existing formats, interface, structured data	0,005	3	0,015
	operating system compatibility	0,013	3	0,039
	mobile compatibility	0,020	2	0,040
	browser compatibility	0,013	4	0,052
	customization			
	customization	0,003	2	0,006
	configurability	0,013	2	0,026
	availability			
	Network provider	0,025	2	0,050
	Intranet- LAN	0,025	3	0,075
	SaaS Provider	0,025	3	0,075
	performance			
	network bandwidth usage/available	0,013	2	0,026
	response time-reactivity (latency)	0,013	2	0,026
	Quality of service	0,003	3	0,009
	off-line functionality (if any)	0,000	0	0,000
	redundancy in data	0,000	0	0,000
	redundancy in services	0,000	0	0,000
	uptime/downtime requirement (99,9%)	0,000	0	0,000
	transfer (data lock-in)			
	manageable transferability of data	0,025	3	0,075
	security			
	Authentication (ie. User+psw)	0,025	4	0,100
	secure protocol security certification (ES. ISO 27001/27002, ISACA COBIT, PCI, NIST)	0,003	1	0,003
	encryption option	0,003	3	0,009
	disaster management	0,012	2	0,024
	security record	0,003	1	0,003
	management			
	data encryption	0,003	2	0,006
	updates/upgrades	0,013	4	0,052
	deletes	0,003	2	0,006
	backups/recovery	0,013	2	0,026
	logs and access investigation	0,003	1	0,003
	relocation (data lock-in)			
fast data portability	0,005	2	0,010	
secure data portability	0,005	2	0,010	
simple data portability	0,005	2	0,010	
control loss				
granularity of access control	0,012	2	0,024	
layer of security	0,005	2	0,010	
data loss				
reliability	0,008	2	0,016	
recover on client request	0,008	2	0,016	
disaster plan	0,003	2	0,006	
legal	liability (of cloud provider)			
	legal protection	0,003	2	0,006
	disclosure with government agency and courts	0,008	3	0,024
	legislation	0,013	1	0,013
	data confidentiality - privacy	0,013	2	0,026
	data ownership	0,013	2	0,026
	data auditability	0,013	2	0,026
	location of the information- country	0,020	4	0,080
	SLAs negotiation or customization	0,013	1	0,013
	TOTAL CONCERN	0,437		1,088
TOTAL BENEFIT	0,563		1,622	
GRAND TOTAL	1,000	PAI	2,710	

The evaluation of cloud computing characteristics included in Table II, looks to verify the need for this new technology for the company while the supplier's ability to provide a product according to the commonly accepted basic general requirements. In fact, quite often, for marketing reasons and without considering the essential features, web-based solutions are advertised as cloud computing solution, also known as "cloud washing" [11].

Two columns, "Weight" and "Rating", must be evaluated to measure the relevant factors in cloud computing environment during the SaaS product evaluation analysis:

- "Weight": considers the importance, relevance or interest of the company to the examined characteristic in a Cloud Computing context to meet the business needs (with a decimal valuation between 0 and 1, with 0="not important" and max.value < 1). The total sum of the values given in this column for the two tables must be equal to 1. Candidates for evaluation are stockholders having reviewed the functional and economic analysis results of the SaaS under evaluation.
- "Rating": estimates the SaaS solution predisposition of addressing the specific attribute in conformity with the specific company context (with values between 1 and 4. With the following evaluations 1="compliance is poor", 2="the compliance is less than average ", 3="above average ", 4="top"). Candidates for evaluation are SaaS experts having analyzed the functional and economic analysis results of the SaaS under evaluation.

A third column "Weight * Rating" or calculated weighted score contains the multiplication result between the "Weight" and "Rating" of each row.

The "Potential Adoption Index" (PAI) is the result of the sum of the weighted score calculated (column "Weight * rating") in Table II and Table III.

Regardless the number of aspects analyzed (or rows) and included in Table II and Table III, the PAI may range from a minimum of 1 and a maximum of 4. The total average score is 2.5.

If the PAI value exceeds 2.5 this means there is a positive balance between economic components, characteristics, risk factors and benefits of cloud computing in the adoption of the analyzed SaaS solution for the particular company. The results, in the proposed example presented in Table III, PAI = 2.71 indicates this event.

In case of multiple comparison, the SaaS solution with the highest PAI indicates the product that has greater potential for the company.

III. CONCLUSION AND FUTURE WORK

This paper presents an integrated top-down selective analysis for calculating the PAI index representing the adoption potential of a SaaS solution for a company.

Functional analysis, economic analysis (TCO) and a detailed attribute analysis are evaluated and linked together in the integrated model.

These attributes (characteristic, benefit and concern) are estimated from stakeholders for their specific relevance in

regard to the company and the cloud environment; and from SaaS experts for their willingness to generate benefits that could be made in the specific business context.

The joint result determines the PAI value, which could be conveniently used to select or assess the SaaS adoption.

The presented framework has not been tested or applied in any real case study. After these preliminary findings, in order to confirm the validity of the proposed solution a more in depth study should be conducted. A case study or other research strategy must also be completed and the results need to be verified and validated.

REFERENCES

- [1] N. G. Carr, "The End of Corporate Computing," MIT Sloan Management Review, vol. 3, 2005, pp. 67–73.
- [2] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing — The business perspective," Decision Support Systems, vol. 1, 2011, pp. 176–189, doi:10.1016/j.dss.2010.12.006.
- [3] W. Sun, X. Zhang, C. J. Guo, P. Sun, and H. Su, "Software as a Service: Configuration and Customization Perspectives," Proc. IEEE Congress on Services Part II, 2008, pp. 18–25, doi:10.1109/SERVICES-2.2008.29.
- [4] S. Leimeister, M. Böhm, C. Riedl, and H. Krcmar, "The Business Perspective of Cloud Computing: Actors, Roles and Value Networks," ECIS 2010 Proceedings, 2010, Available at: <http://aisel.aisnet.org/ecis2010/56> [retrieved: 03, 2013].
- [5] P. Mell and T. Grance "The NIST Definition of Cloud Computing," 2011.
- [6] C. P. Bezemer, and A. Zaidman, "Multi-tenant SaaS applications: maintenance dream or nightmare?" Proc. Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), ACM, 2010, pp. 88–92, doi:10.1145/1862372.1862393.
- [7] M. Nitu, "Configurability in SaaS (software as a service) applications," ACM Press, 2009, pp. 19–26, doi: 10.1145/1506216.1506221.
- [8] S. Bibi, D. Katsaros, and P. Bozaris, "Business Application Acquisition: On-Premise or SaaS-Based Solutions?" IEEE Software, 29(3), 2012, pp. 86–93, doi:10.1109/MS.2011.119.
- [9] B. Martens, M. Walterbusch, and F. Teuteberg, "Costing of Cloud Computing Services: A Total Cost of Ownership approach," Proc. of the Annual Hawaii International Conference on System Sciences, 2011, pp. 1563–1572.
- [10] P. Géczy, N. Izumi, and K. Hasida, "Cloudsourcing: managing cloud adoption," Global Journal of Business Research (GJBR), vol. 6, 2012, pp. 57–70.
- [11] A. Adamov and M. Erguvan, "The truth about cloud computing as new paradigm in IT," Proc. International Conference on Application of Information and Communication Technologies (AICT 2009), 2009, pp. 1–3, doi:10.1109/ICAICT.2009.5372585.

Context-Aware Data-Flow in the Cloud

Mandy Weißbach and Wolf Zimmermann
Institute of Computer Science

University of Halle
Halle (Saale), Germany

Email: {weissbach, zimmermann}@informatik.uni-halle.de

Welf Löwe

Software Technology Group

Linnaeus University
Växjö, Sweden

Email: welf.lowe@lnu.se

Abstract—In the last few months, clients of services running in a cloud are getting more and more aware of storing and processing their data in the cloud. In this paper, we present a context-aware data-flow analysis approach to allow clients to negotiate services that store or process (directly or indirectly) their data in undesired locations. The approach is context-aware to satisfy the stateless character of services in a multi-tenant cloud. We show that the use of a dynamic context-aware data-flow analysis ensures that the clients' data does not reach undesired locations in the cloud.

Keywords-context-aware; data-flow; service-oriented; data security;

I. INTRODUCTION

Undoubtedly, Cloud Computing is one of the most growing internet technologies worldwide. The preparatory study undertaken for the European Commission estimates that the public cloud would generate EUR 250 billion in GDP (Gross domestic product) in 2020 [1].

Reasons for the popularity of Cloud Computing are obvious: IT-departments can be outsourced, investments in resources, e.g., hardware, software or space, become no longer necessary, energy costs can be reduced and cloud services are available from everywhere.

Cloud Computing also plays an important role in the private sector. About 56 % of the internet users store private data, e.g., pictures, music or documents, in the cloud.

Because of the private and commercial use of Cloud Computing sensitive data may be stored and processed by cloud services. Unfortunately, encryption of data is not an option to keep sensitive data secure. When data needs to be processed by the used cloud services, it needs to be available in decrypted form [4](research in the field of processing encrypted data is just at the beginning). The abstracted infrastructure of a cloud makes it impossible for the user to know the exact location their applications or data are running on [2], [3]. So, one major obstacle in using cloud services is that clients have no control where their data are being stored and processed [2].

However, if cloud servers are located at different locations, they obey national laws on the server's location. These might be rather different than the location of the cloud user. Therefore there might be unauthorized access to clients' data

that might be legal in the country of the server of the cloud service, e.g., through [5], but illegal in the client country [6]. Despite this fact, we focus on data-security in the cloud.

In our previous work [6], we described an approach that enables a client to control the data-flow in the cloud. Data-flow to undesired locations could be negotiated by the client. Cloud services were allowed to use other services in desired locations and so on. Even callbacks between cloud services installed at desired locations are allowed [6].

Our previous work assumes that there is one client, which has a list of undesired locations. This client uses the cloud services by its own. So, there exists only one view on the cloud services. In this work, we generalize to cloud services used by several clients where each client may have its own wish of undesired locations.

Suppose client X has country $wLoc$ as its undesired location and client Y has country $vLoc$ as its undesired location, cf. Figure I. Client X calls service Z which is installed on a server located in country $xLoc$. Service Z can use service W or service V . Service W is installed on a server located in country $wLoc$. Service V is installed on a server located in country $vLoc$. Since the Service Z is installed on a server in country $zLoc$, both clients are going to use service Z . While the static data-flow analysis for client X is done, service Z can use service V , because V is located in country $vLoc$. Service W would be negotiated by client X , because W is installed on a server located in country $wLoc$. The same data-flow analysis is done for client Y . Now, service Z can not use service V because service V is installed on a server in country $vLoc$, an undesired location of client Y , cf. 1(a). Service Z can use service W , because service W is a desired location of client Y . However, client X wants to use service Z , service Z can not derive whether client X or client Y is calling and therefore service Z does not know which service (service V or service W) to use (One-View-Problem described in Figure 1(b)). So, our approach is extended to support different views from different clients to support multi-tenant services.

We realized this approach with a context-aware data-flow analysis in the cloud. The used static data-flow analysis is an conservative approach [6], which can guarantee in the case of an positive answer that no sensitive data flow

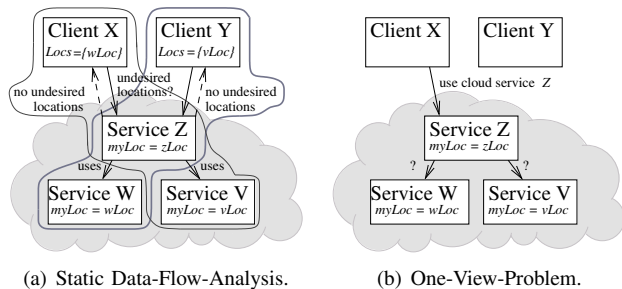


Figure 1. Controlling Data-Flow in the Cloud.

```

/*@return : true -> data-flow to undesired location(s)
           false -> data-flow only to desired locations*/
/* UnDesX,S,L : data-flow from service B over provided
               functions S to service X in countries L*/
BOOL undesired(SET(ProvidedB) S, SET(Locations) Locs) {
  if myloc ∈ Locs return true;
  foreach service X used by B do
    if UnDesX,S,L return true;
  return false;
}
    
```

 Figure 2. Implementation of *undesired* [6].

direct or indirect to services installed on servers in undesired locations. In the case of a negative answer, there could be a direct or indirect data-flow to services installed on servers in undesired locations. Instantiation of one service on several servers in different countries are not considered. This work follows the service-level-agreement principle (SLA). So, based on the result of the context-aware data-flow analysis, the client can negotiate a service that is installed on a server at an undesired location. In order to increase trust in the given answer, we assume the use of the proposed cryptographic approach in [6].

The paper is organized as following: In Section II, we introduce a service model example. The context-aware data-flow analysis with respect to the presented example is given in Section III. Section IV discusses related work and Section V concludes this work.

II. SERVICE MODEL EXAMPLE

This section gives a short overview of our service model and states the problems that could occur if we are not aware of the context.

We assume that each service A provides a set of functions, denoted by $Provided_A$. This might be given as a WSDL-Description (Web Service Description Language). Furthermore, each service A must use another service. We assume that this is not hard-coded in the implementation of A , but there is a pair of variables $I x$ where I contains the set of functions that is called on x , and x can be bound (dynamically) to a service X that provides at least I , i.e., $I \subseteq Provided_X$. Functions in I are called *required functions* of A w.r.t. x . The set of candidate services must be published and we assume that a registry Reg maintains all published services. We also assume an acyclic use structure of the

services. Section III shows how this assumption can be dropped.

Example 1: Multi-Tenant Clients

Consider services A , B and C in Figure 3. $A.b$ can be bound to service B and also $C.b$ can be bound to service B . The provided interface of B is $Provided_B = \{x, undesired\}$. The required functions of A w.r.t. b are $\{x, undesired\}$. The required functions of C w.r.t. b are also $\{x, undesired\}$. The required functions of B w.r.t. d are $\{f, undesired\}$. So service B can simultaneously be used by service A and by service C . ■

A client would like to negotiate an agreement that a selected service guarantees to avoid data-flow from the clients' data to a set $Locs$ of undesired locations. For the purpose of negotiation, service B may offer a function $undesired \in Provided_B$ that returns *true* iff data flows via some operations o from the provided interface of B to services at undesired locations, cf. with Figure 2.

Remark: It is sufficient to take into account only the set $S \subseteq Provided_B$ of operations used by the client. □

If service A uses service B , it needs to ask B (via B 's function *undesired*) whether it can guarantee that its data do not flow to a location in $l \in Locs$ (undesired locations). Obviously, this needs only to be guaranteed for those operations of B where B passes (possibly processed) data of A . For simplicity, we assume that each service X knows its location and this location is stored in a constant $X.myLoc$.

Example 2: Negotiation of Undesired Locations

Consider services A , B , C , D , E and F in Figure 3. Service A would like to use service B . Service B is located in $BLoc$. B can also be used by service C . B itself uses service D located in $DLoc$. Service D uses service E or F . E is located in $ELoc$. F is located in $FLoc$. We assume that all services (except possibly client A and client C) are published.

Suppose that client (service) A wants to avoid storing its data (neither in original nor in processed form) at servers in $FLoc$. Before client A actually uses service B it would like to know whether data passed to B are never stored at a server in $FLoc$. Let $Locs$ be the set of servers in $FLoc$. The procedure *negotiate* searches for a published service B offering at least the operations specified in I_B (I_B is the set of functions of the required service that are called from client A). For the purpose of negotiation, client A calls $undesired(I_B, Locs)$ because client A calls $b.x(mydata)$, if b is bound to service B . Service B calls function $f \in Provided_D$, if d is bound to service D . So, a call of $b.x(mydata)$, if b is bound to service B , implies that data of client A flows to service D by the call $d.f(data)$ of service B , if d is bound to service D . Thus, the call $undesired(I_B, Locs)$ must return *false* only if $B.myLoc \notin Locs$ and $undesired(I_D, Locs)$ returns *false*. The functions $f \in Provided_D$ calls g and $g \in Provided_E$ or $g \in Provided_F$, it depends on whether

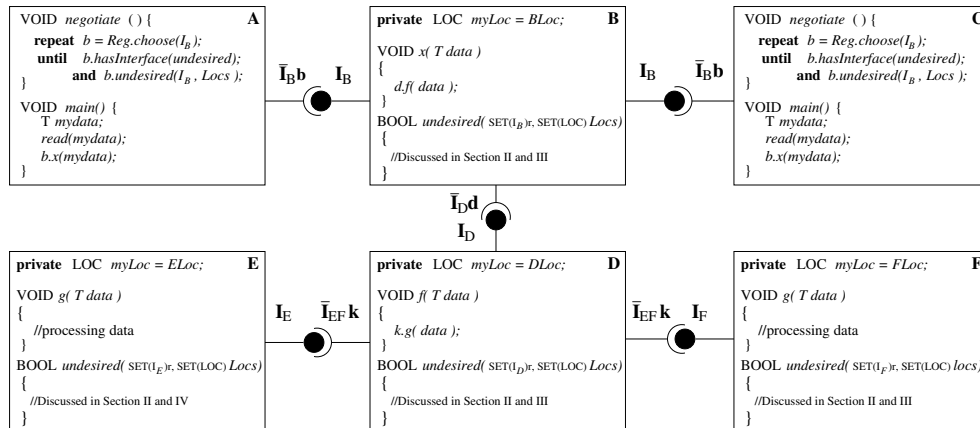


Figure 3. Storing Data at undesired locations: Two-View-Example

k is bound to service E or F . The argument of the call $d.f(data)$ flows to the call $k.g(data)$ of service E if k is bound to service E or $k.g(data)$ of service F if k is bound to service F , respectively. So, if k is bound to service E , there is a data-flow from client A over services B and D to service E . Service E is located in $ELoc$, which is not a undesired country. Therefore $B.undesired(I_B, Locs)$ returns *false*, because $D.undesired(\{f\}, Locs)$ returns *false* i.e., client A can use service B . But if there is a data-flow from client A over service B and D to service F , located in $FLoc$, $B.undesired(I_B, Locs)$ returns *true*. Because client A does not want to store or process data in $FLoc$, $D.undesired(\{f\}, Locs)$ returns *true* and therefore $B.undesired(I_B, Locs)$ returns *true*, i.e., client A cannot use service B . ■

Because the service model architecture is multi-tenant, service B can simultaneously be used by client A and by client C . The set $Locs_A$ of undesired locations of client A might be different from the set $Locs_C$ of undesired locations of client C . If B uses service D , it needs to ask D (via D 's function *undesired*) whether it can guarantee that A 's data do not flow to a location in $Locs_A$. Obviously, the data-flow needs to be guaranteed in context of the clients. If the context is not considered, service D can not distinguish between the calling services A and C . So if A calls B and B calls D , it is possible that a later call of D by B which was called by C is not detected as a call from C . So the undesired countries $Locs_A$ may be applied for client C .

Example 3: Context-Aware Data-Flow

Consider the services A, B, C, D, E and F in Figure 3. Service A would like to use service B . Service B is located in $BLoc$. B itself uses service C located in $CLoc$ while C uses service D or E . D is located in $DLoc$. E is located in $ELoc$. F is located in $FLoc$. Client A wants to avoid storing its data (neither in original nor in processed form) at servers in $FLoc$. C wants to avoid storing its data (neither in original nor in processed form) at servers in $ELoc$.

Suppose the negotiation process starts. Service B, D and E will be accepted by A because $undesired(I_B, Locs)$ returns *false*. Before client A starts to use service B , client C tells service B it also wants to use service B . A starts the negotiation process and for service B, D and F , the negotiation process will succeed. However, service A starts to use service B . Service B calls function f of service D . But service D can not distinguish between clients A and C . So it is possible, that service D binds to service F . But the undesired countries $Locs_A$ include the location $FLoc$ of service F .

To distinguish between client A and client C , we need to introduce a context-aware data-flow analysis mechanism to know or compute the chain of used services by a client in the service model.

III. CONTEXT-AWARE MECHANISM

We introduce the principle of context-aware attributes. Lists of context-dependent attributes are created. If the function *undesired* of a service B , called by client A , returns *false*, the *call id* of the caller and the called service is added to the attribute list, e.g., *cl* of service B . However, service B can distinguish with the help of the *call id*, whether client A or client C was the original caller.

Example 4: Context-Aware Multi-Tenant Clients

Consider the services A, B, C, D, E and F in Figure 3. Client A would like to use service B and specifies a set $Loc_A = \{FLoc, XLoc\}$ of undesired locations. Also, client C would like to use service B and specifies a set $Loc_B = \{ELoc, XLoc\}$ of undesired locations. Service B is located in $BLoc$. B calls service D located in $DLoc$ while D uses service E or F . Both, service E and service F offer the same functionality. The only difference is, that service E is located in $ELoc$ and F is located in $FLoc$.

Suppose, the following scenario: client A wants to bind to service B . The negotiation process starts. Service B uses service D and service D finds through a Registry *Reg*

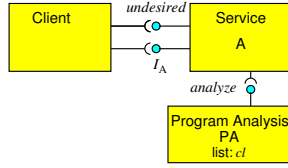


Figure 4. Architecture of the Context-Aware Mechanism.

service E and F . Since service F is located in $FLoc$, an undesired location, the call $undesired(I_F, Locs)$ will return *true*. Client A will negotiate the use of B and client A will start to find via the *Reg* new services. A is going to bind to service B again. This time B is going to ask D and D is calling service E . The call $undesired(I_E, Locs)$ by service D will return *false* because $ELoc$ is not in $Locs_A$. However, service D wants to bind to service E in context of service A (service A called B , B called D). So, the *callid*, computed with the service use chain, can be stored in a context attribute list *cl*. Before service D calls function $g \in Provided_E$, service D checks if the *callid* of E is registered for A . If there is a *callid* of E registered, service D knows that service E can be used. Now, if service C wants to use service B , a new negotiation process starts. Now, the call of $undesired(I_B, Locs_C)$ and the call of $undesired(I_D, Locs_C)$ will return *false*. The call of $undesired(I_E, Locs_C)$ will return *true* and no *callid* is set and the list with the used service chain will be discarded. Client C will negotiate service B and the negotiation process starts as described before. This time, service D calls F . The function $undesired(I_F, Locs_C)$ will return *false*. So, the *callid* of B , D and F will be added to the context attribute list *cl* of client C . However, if client C calls service B and service B calls service D , service D can choose with respect to the context attribute service F .

To implement the context-aware data-flow analysis, we need a trusted third party which

- can compute the resource information (location) of the used cloud service,
- ensures that the used cloud services act according to promised behavior of *undesired* and
- maintains the information of the chain of used services by a certain client.

Example 5: Context-Aware Mechanism

The first requirement is satisfied by every service itself, cf. section II. Every service stores its location information. The second and third requirement can be given by an independent certified program analysis service *PA*. *PA* performs the program analysis, computes the result of *undesired* and will be extended to maintain the information of the chain of used services by a certain client. For more details of the work of the unextended *PA*, we refer to [6].

We propose a context-aware mechanism described by the

following algorithms in pseudo code and a sequence diagram in Figure 5. To start the negotiation the client calls a registry to ask for a service with the required Interfaces by providing the set of undesired locations *Locs*, the *callID* of the client A and the required Interface I_{req_A} :

Algorithm 1: negotiate

INPUT: $callID, Locs, I_{req_A}$
OUTPUT: *true, service can be used*
 false, service can not be used

```

repeat Service  $b = Registry.choose(I_{req_A})$ 
until  $\neg b.undesired(I_{req_A}, Locs, callID)$ 
end
return true
    
```

However, the used service B selects a Program Analyzer *pa* and starts the data-flow analysis, by calling *analyze*, cf. 5.

Algorithm 2: undesired

INPUT: $I_{req_A}, Locs, callID$
OUTPUT: *true, data – flow to undesired locations*
 false, –data – flow to undesired locations

```

 $pa \leftarrow choose()$ ;
return  $pa.analyze(Locs, callID, I_{req_A}, sourceText_B)$ 
    
```

Besides the data-flow analysis the Program Analyzer *pa* also stores the context-aware attribute *callID* of the client to keep track of the used services by client A .

Algorithm 3: analyze

INPUT: $Locs, callID, I_{req_A}, sourceText_B$
OUTPUT: *true, data – flow to undesired locations*
 false, –data – flow to undesired locations

```

 $callID \leftarrow computeCallID(callID)$ 
for each location in  $Locs$  do
if (location ==  $sourceText_B.myLoc$ ) then return true;
end
end
 $cl \leftarrow cl.add(callID)$ 
 $I_{req_B} \leftarrow doDataFlowAnalysis(I_{req_A}, sourceText_B)$ 
return  $negotiate(callID, Locs, I_{req_B})$ 
    
```

Remark:

As the *PA* is able to keep track of the analysis requests of client A , it can check for cycles before processing the analysis request. In particular, it checks whether a query $undesired(callID, Locs, I_{req_B})$ for client A is currently being analyzed, i.e., whether there is an open analysis request $undesired(S', Locs)$ with $S' \subseteq S$. If yes, it can return immediately *false*. This is valid because if there is a data-flow from S' to an undesired location *loc*, then there must be another call of a provided function to service B with a data-flow to an undesired location. \square

Now, with the help of the computed list containing the chain of used services of client A , this information can be used to guarantee, that the data of client A flows only to undesired locations. Before every service connects to another service it can be checked asking the used *PA* if in the context of the client this connection is allowed.

Remark: We assume a IAAS in a trusted cloud environment [7]. This approach depends on trust in a trusted cloud federation and we are using the encryption mechanism

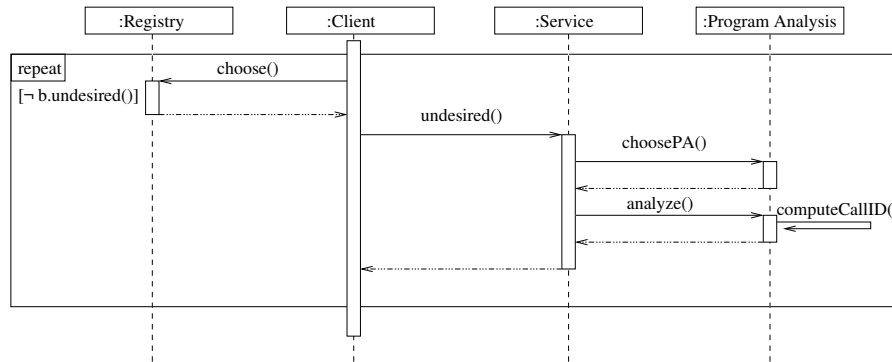


Figure 5. Sequence Diagram of the Context-Aware Mechanism.

described in [6].

IV. RELATED WORK

[6] considers data security in the cloud. In contrast to our work, this approach is not context-aware. [8] monitors data-flow between services in order to detect malicious services. They do not do a static data-flow analysis but they assume a multi-tenant cloud infrastructure. Also, context-awareness with respect to the client is not assumed. [9] investigates data-flow analysis in the context of service computing. Compared to our work, they focus on static process adaptation to investigate if a service gets all the data it needs. [10], [11] focus on data security within smart-phone applications. While we allow sensitive data to leave the client, they forbid sensitive data to leave the smart-phone.

There are also many works on context-aware service-oriented systems. Truong and Dustdar [12] present a couple of projects, e.g., CA-SOA [13], CoWSAMI [14], WASP [15] and inContext [16], [17], to make service-oriented systems context-aware. CoWSAMI [14] is an interface-aware context-gathering-environment. CA-SOA [13] formalizes an ontology-based context model. Different views of different clients using a chain of web services were not considered. [18] proposes a multi-tenant service-oriented architecture middleware for Cloud Computing. They focus on multiple users sharing an instance and native multi-tenancy. In contrast to our work, using certain services in context of the location is not considered.

Baldauf et al. also states some requirements that need to be supported by a context-aware system. In contrast to our work, [13], [14], [15], [16], [17] assume, that the context information of the user has to be collected by some mechanism, e.g., polling [16], [13]. In our work, the client itself supports the system with context information, the list of undesired locations, mechanisms like polling are not needed. [19] also presents techniques to compute, with the help of context information, the right service to get coupled to. In our work, the client itself can decide whether a context

is given or not. A computation of contextual information [20] to find the best fitting service does not need to be done.

Focusing data security in the cloud is done by [3]. Brandic et al. guarantee data security by data fragmentation. A data analyst or the domain expert decide where data can be stored and which data need to be fragmented and stored in different geographical regions. The client itself can not decide where its data is stored or processed.

To the best of our knowledge, there exist no paper that is using the context information of a client to control the data-flow in the cloud and enables the client to negotiate services.

V. CONCLUSION

In this work, [6] was extended to allow different views on a service-oriented system in the cloud. The extended work allows multiple clients to decide where their data is stored, processed and transferred within the cloud. Our approach supports different views to fit into multi-tenant service-oriented architectures.

We have two context information: the client information of used services and a list of undesired countries specified by the client. With the extended static data-flow analysis and the contextual information, the coupling of services in context of the user can be computed at runtime. We obtain a multi view or multi-tenant environment with loosely coupled services, which will be coupled on demand in context of the client.

Techniques to collect contextual information, e.g., polling, are not an issue. Every service is supported with the list of undesired countries by the client itself (direct or indirect). Information of the used services are stored by a program analysis service.

To evaluate the proposed approach, the implementation of a tool is in process and subject for future work.

In this work, we considered data-flow analysis on the SaaS (Software as a Service) level. Subject of further work will be the generalization of the data-flow analysis to IaaS (Infrastructure as a Service) and PaaS (Platform as a Service).

Due to the complexity of the IaaS and PaaS, we expect on this level that data-flow analysis becomes more complex and maybe some new abstraction mechanisms are needed for feasibility. Another opportunity for program analysis is to analyze the conformance to compliance rules as they have similar characteristics as data-flow: the client cannot always check the conformance or may even not observe violations of compliance.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

- [1] *Quantitative Estimates of the Demand for Cloud Computing in Europe and the Likely Barriers to Up-take*, Final Report, IDC Std. D4, July 2012. [Online, retrieved: 03.2013]. Available: http://ec.europa.eu/information_society/activities/cloudcomputing/docs/quantitative_estimates.pdf
- [2] D. Durkee, "Why cloud computing will never be free," *Queue*, vol. 8, no. 4, 2010, p. 20.
- [3] I. Brandic, S. Dustdar, T. Anstett, D. Schumm, F. Leymann, and R. Konrad, "Compliant cloud computing (c3): Architecture and language support for user-driven compliance management in clouds," in *IEEE CLOUD*, 2010, pp. 244–251.
- [4] L. Wei, H. Zhu, Z. Cao, W. Jia, and A. Vasilakos, "Seccloud: Bridging secure storage and computation in cloud," in *Distributed Computing Systems Workshops (ICDCSW), 2010 IEEE 30th International Conference on*, Jun 2010, pp. 52–61.
- [5] "Uniting and strengthening america by providing appropriate tools required to intercept and obstruct terrorism act of 2001 (usa patriot act)," Oct 2001, effective February 1, 2002. [Online, retrieved: 03.2013]. Available: <http://thomas.loc.gov/cgi-bin/query/z?c107:H.R.3162.ENR>
- [6] M. Weissbach and W. Zimmermann, "Controlling data-flow in the cloud," in *The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, W. Zimmermann, Y. W. Lee, and Y. Demchenko, Eds. ThinkMind, 2012, pp. 24–29.
- [7] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *HOTCLOUD*. USENIX, 2009.
- [8] J. Du, W. Wei, X. Gu, and T. Yu, "Runtest: assuring integrity of dataflow processing in cloud computing infrastructures," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. New York, NY, USA: ACM, 2010, pp. 293–304. [Online, retrieved: 03.2013]. Available: <http://doi.acm.org/10.1145/1755688.1755724>
- [9] W. Song, X. Ma, S. Cheung, H. Hu, and J. Lu, "Preserving data flow correctness in process adaptation," *Services Computing, IEEE International Conference on*, vol. 0, 2010, pp. 9–16.
- [10] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6. [Online, retrieved: 03.2013]. Available: <http://dl.acm.org/citation.cfm?id=1924943.1924971>
- [11] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting privacy leaks in iOS applications," in *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS)*, Feb. 2011. [Online, retrieved: 03.2013]. Available: http://www.isoc.org/isoc/conferences/ndss/11/pdf/9_2.pdf
- [12] H. Truong and S. Dustdar, "A survey on context-aware web service systems," *International Journal of Web Information Systems*, vol. 5, no. 1, 2009, pp. 5–31.
- [13] I. Y. L. Chen, S. J. H. Yang, and J. Zhang, "Ubiquitous provision of context aware web services," in *Proceedings of the IEEE International Conference on Services Computing*, ser. SCC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 60–68. [Online, retrieved: 03.2013]. Available: <http://dx.doi.org/10.1109/SCC.2006.110>
- [14] D. Athanasopoulos, A. V. Zarras, V. Issarny, E. Pitoura, and P. Vassiliadis, "Cowsami: Interface-aware context gathering in ambient intelligence environments," *Pervasive Mob. Comput.*, vol. 4, no. 3, Jun. 2008, pp. 360–389. [Online, retrieved: 03.2013]. Available: <http://dx.doi.org/10.1016/j.pmcj.2007.12.004>
- [15] M. Zuidweg, J. Goncalves Filho, and M. van Sinderen, "Using p3p in a web services-based context-aware application platform," in *Proceedings of EUNICE 2003 9th Open European Summer School and IFIP WG6.3 Workshop on Next Generation Networks*, E. Halasz, C. Lukovszki, and T. Marosits, Eds. Budapest: Budapest University of Technology and Economics, Sep. 2003, pp. 238–243. [Online, retrieved: 03.2013]. Available: <http://doc.utwente.nl/66531/>
- [16] H.-L. Truong, L. Juszczak, S. Bashir, A. Manzoor, and S. Dustdar, "Vimoware - a toolkit for mobile web services and collaborative computing," in *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications*, ser. SEAA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 366–373. [Online, retrieved: 03.2013]. Available: <http://dx.doi.org/10.1109/SEAA.2008.42>
- [17] H.-L. Truong, S. Dustdar, D. Baggio, S. Corlosquet, C. Dorn, G. Giuliani, R. Gombotz, Y. Hong, P. Kendal, C. Melchiorre, S. Moretzky, S. Peray, A. Polleres, S. Reiff-Marganiec, D. Schall, S. Stringa, M. Tilly, and H. Yu, "incontext: A pervasive and collaborative working environment for emerging team forms," in *Proceedings of the 2008 International Symposium on Applications and the Internet*, ser. SAINT '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 118–125. [Online, retrieved: 03.2013]. Available: <http://dx.doi.org/10.1109/SAINT.2008.70>
- [18] A. Azeez, S. Perera, D. Gamage, R. Linton, P. Siriwardana, D. Leelaratne, S. Weerawarana, and P. Fremantle, "Multi-tenant soa middleware for cloud computing," in *IEEE CLOUD*, 2010, pp. 458–465.
- [19] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, Jun. 2007, pp. 263–277. [Online, retrieved: 03.2013]. Available: <http://dx.doi.org/10.1504/IJAHUC.2007.014070>
- [20] A. Danylenko, C. Kessler, and W. Löwe, "Comparing machine learning approaches for context-aware composition," in *Proceedings of the 10th international conference on Software composition*, ser. SC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 18–33. [Online, retrieved: 03.2013]. Available: <http://dl.acm.org/citation.cfm?id=2025951.2025954>

A Coordinated Reactive and Predictive Approach to Cloud Elasticity

Laura R. Moore, Kathryn Bean and Tariq Ellahi

SAP Next Business and Technology

SAP (UK) Ltd

Belfast, UK

Email: {laura.moore, kathryn.bean, tariq.ellahi}@sap.com

Abstract—Based on pay-per-use service-oriented architectures, the cloud computing paradigm promises cost-efficient IT solutions. To meet fluctuating demands efficiently, Platform-as-a-Service solutions offer shared environments with on-demand scalability. It remains an open challenge for service providers to implement elastic scalability mechanisms capable of optimally utilizing resource whilst simultaneously guaranteeing that application performance continues to meet Quality of Service metrics. Typically, cloud providers offer only reactive rule-based mechanisms for triggering scaling actions. We introduce a new elasticity management framework that combines reactive and predictive controllers. Our elasticity controller builds predictive models online based on the reactive rules, representing a natural extension to the common offering. We discuss the underlying architecture of the framework and describe how the controllers operate in tandem and complement each other. We present a case study based on real datasets that demonstrates the feasibility of our real-time cloud capacity framework.

Keywords-elasticity; predictive; auto-scaling; platform-as-a-service.

I. INTRODUCTION

Cloud computing, with its promise of cost-effective computing for end-users and improved resource utilization for cloud providers, continues to grow in popularity. A recent Gartner report predicts a compound annual growth rate of 36% for Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) from \$7.6B in 2011 to \$35.5B in 2016 [1]. This increase in user demand, coupled with new technologies, is driving a dramatic increase in cloud infrastructure scale, heterogeneity and complexity [2][3]. To efficiently handle their resources, cloud providers require intelligent methods of automated dynamic infrastructure management.

One of the key features of cloud computing is elasticity. Elasticity refers to the ability of a system to grow and shrink dynamically such that it only uses resources that are necessary to cope with the current load. This paper presents details of the design and current implementation of a real-time cloud capacity framework, Platform Insights. The particular contribution of the paper is the design of an elasticity controller that:

- Couples reactive and predictive elasticity management techniques and coordinates auto-scaling requests
- Can be used without off-line training

- Utilizes multi-timeframe information to allow short-term auto-scaling decisions to be made in the context of the expected longer-term workload demand

The rest of the paper is organized as follows. Section II presents a summary of related work. Section III gives details on the architecture of Platform Insights. Section IV describes the implementation of Platform Insights, giving an overview of the configuration options and the integrated predictive models. Section V presents a case study in which the ClarkNet [4] and the 1998 World Cup data access logs [4][5] are used to simulate driving the SPECjEnterprise2010 benchmark. Resulting QoS statistics and resource provisioning decisions are evaluated. Concluding remarks are given in Section VI.

II. RELATED WORK

Auto-scaling techniques can be classified as either reactive (the system reacts to changes but does not anticipate them) or predictive (the system tries to predict future resource requirements in order to ensure sufficient resource is available ahead of time) [6]. Reactive rule-based methods define scaling conditions based on a target metric reaching some threshold and are offered by several cloud providers such as Amazon [7] or third party tools such as RightScale [8] or AzureWatch [9]. Beyond static thresholds, [10] proposes a regression method to dynamically adapt thresholds to meet QoS targets, but does not predict future workload.

Predictive auto-scaling approaches tend to be based on time series analysis, control theory, reinforcement learning, or queuing theory. One strategy is to use a workload predictor and then use a performance model to determine the number of servers required to service the predicted demand. A variety of performance models has been proposed in the literature. Examples include the use of splines to map the request rate to the observed percentage of slow requests given the number of active servers [11], queuing networks [12], and cost optimization models [13]. The predictive controller component of Platform Insights also uses workload forecasts and performance models: the workload forecast is used to estimate a mid-term trend in demand, which is used as an input to a performance model mapping workload-per-server to future QoS.

Hybrid methods, coupling reactive and predictive controllers, have also been proposed. Proposals include using a predictive method and a reactive method to determine when to provision resource over a long time-scale (hours and days) and a short time-scale respectively [14], or using a predictive controller to control scale in and a reactive controller for scale out [15][16]. In [15] a regression-based method is used. In [16] the authors base their models on queuing theory and they find that SLA violations are reduced by a factor of 2 to 10 compared to a purely reactive controller.

We also implement a hybrid approach, but our reactive and predictive controllers are both capable of triggering scale in and scale out actions. Auto-scaling decisions are coordinated and conservative policies are applied to avoid premature decommissioning of resource.

III. ARCHITECTURE OF PLATFORM INSIGHTS

Typical enterprise applications are composed of a number of services that run on multi-tier architectures. To provide adequate resource to handle client demand, each tier requires monitoring and elasticity. Platform Insights monitors each component of the platform stack individually and evaluates appropriate elastic scaling actions. In Section III-A the architecture of the reactive controller is described, and this is then extended in Section III-B to show how a predictive controller operates in conjunction with the reactive controller.

A. Reactive Elasticity Management

Reactive elasticity management takes place by monitoring scaling rules, which are configured by application architects and administrators. The form of the scaling rules is discussed later in Section IV. This section is dedicated to describing the software agents that make up the reactive auto-scaling alerter component of Platform Insights. Figure 1 shows the steps taken by the system to register a new elasticity rule when it is submitted by an administrator. In its current form, Platform Insights allows rules to be submitted or deleted at any time for running applications, but does not take responsibility for checking that rules do not conflict; this functionality is left to future work. Figure 2 shows the operation of the reactive auto-scaling alerter component. The two figures show the agents that are involved at each stage and their interactions. Each agent is now discussed in turn.

- *The Request Manager* receives requests submitted to the platform by users of the web portal. The web portal has facilities for the application architect to a) monitor resource usage consumed by each instance (real-time or historical), b) configure and manage elasticity rules, c) monitor utilization metrics associated with elasticity rules, and d) receive relevant alerts and/or log messages.
- *The Rule Creator* receives instructions from the *Request Manager* to set up new elasticity rules. When it receives a new scaling rule, it liaises with other components to coordinate the instantiation of the new rule.

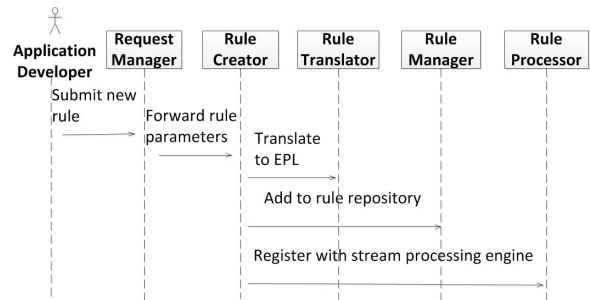


Figure 1. Scaling rule submission sequence diagram.

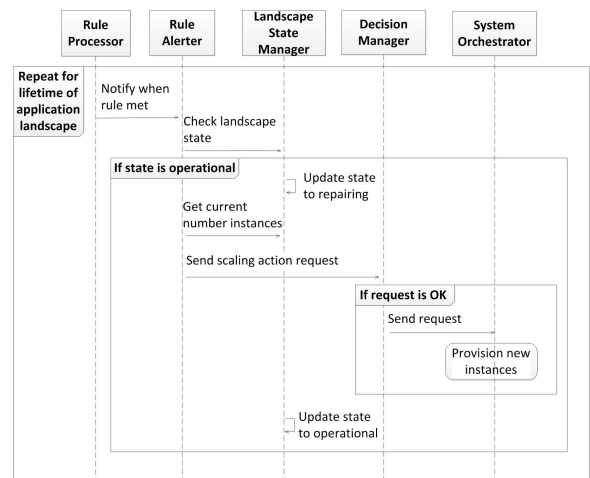


Figure 2. Reactive auto-scaling sequence diagram.

- *The Rule Translator* is responsible for translating the configured attributes of a newly submitted scaling rule into EPL (Event Processing Language) statements that can be monitored by the *Rule Processor*. It maintains a dictionary of pre-set statement templates and parses the incoming data against these templates.
- *The Rule Manager* is responsible for the lifecycle management of elasticity scaling rules. It maintains a repository of scaling rules for each application landscape and of associated EPL statements registered with the *Rule Processor*.
- *The Rule Processor* is based on a complex event processing (CEP) engine. Currently Esper is used as the CEP engine as it is lightweight, can be easily embedded in a Java application and allows new queries to be registered dynamically so that scaling rules can be submitted at any time [17].
- *The Rule Alerter* is responsible for determining what scaling action should be taken upon violation of a scaling rule and broadcasting any relevant information to enable the platform to execute the scaling action.
- *The Landscape State Manager* stores and monitors the application state throughout its entire lifecycle.

Four states are defined for each application landscape: offline, starting, operational and repairing.

- *The Decision Manager* is a centralized component receiving requests from both the *Rule Alerter* and the *Tier Manager* (predictive component, see Section III-B). It is responsible for ensuring coordination of auto-scaling requests; this process is described in more detail in Section IV-D below. Validated requests are broadcast over the messaging bus.
- *The System Orchestrator* listens for the broadcast auto-scaling requests. It is responsible for executing the requests by provisioning and removing server instances.

The particular form of the scaling rules used by Platform Insights is discussed in Section IV below. Rule-based elasticity management only enables the system to scale after some condition has already been met, and so predictive elasticity management is also utilized in Platform Insights.

B. Predictive Elasticity Management

The Platform Insights predictive analysis engine estimates the resource requirements needed by the workload in the near future to satisfy QoS constraints. Platform monitoring data is aggregated on each tier prior to being fed in to the predictive models. Aggregating data at the tier level is acceptable as the platform components are assumed to have load balancers. Esper [17] is used to perform this pre-aggregation of the data (both workload and QoS metrics). More information on the specific data aggregations and predictive models will be given in Section IV-C. Esper listens to the underlying stream of monitoring data, aggregates it as appropriate, and then publishes the aggregated data using the messaging system. Listeners for the predictive models subscribe for all relevant data. The predictive engine comprises the following software agents:

- *The Data Listener* subscribes for relevant data that has been aggregated by Esper and published over the messaging bus and distributes it to the *Data Processor*.
- *The Data Processor* has the responsibility of feeding the data to the appropriate *Model Updater*.
- *The Model Updater* deals with the new data by updating its model and/or doing a prediction using the new data. On the basis of the prediction, it may decide an auto-scaling action is necessary, in which case it sends a request to the *Tier Manager* for assessment.
- *The Tier Manager* evaluates all auto-scaling actions requested for the tier by the *Model Updater*. If the *Tier Manager* agrees that the scaling action is appropriate then a request is sent to the *Decision Manager* (see Section III-A above).

This section has described the underlying architecture of both the reactive and predictive components of the Platform Insights elasticity management framework; the next section gives details on their implementation.

IV. IMPLEMENTATION OF PLATFORM INSIGHTS

This section discusses the nature of the predictive models built from the scaling rules and how they enable auto-scaling decisions to be made. Section IV-A describes the configuration of the scaling rules, Sections IV-B and IV-C describe the implementation of the rule processing and predictive models respectively, and Section IV-D describes how the *Decision Manager* coordinates auto-scaling requests.

A. Configuration of Scaling Rules

The scaling rule strategy is to first perform some aggregation of metric data pertaining to each server instance over some time window. These per-server values are then further aggregated to a single tier value, which is compared against a threshold value. If the rule condition is met then an action is triggered to add or remove some number of instances whilst staying within some limits. The rules are composed of the following elements. *Metric*: one of the metrics exposed on the server and monitored by the system. *Operator*: the comparison operator to be used in evaluating the metric value; allowed operators are 'EQ', 'LT', 'LTE', 'GT' and 'GTE'. *Value*: the value threshold for the metric being observed. *Aggregate Function*: the statistical aggregate function to be used for metric evaluation; allowed values are 'average', 'sum', 'median' and 'raw' (which indicates no aggregation). *Scope*: the metric scope with respect to the all the server instances in the tier; allowed values are 'min', 'max' and 'sum'. *Time Window*: the length of the sliding time window over which to continuously monitor the metrics and evaluate whether or not they meet the scaling rule condition. *Min Time Between Alerts*: the minimum time between auto-scaling actions. *Limit*: specification of the maximum or minimum number of instances allowed in the tier. *Scale By Type*: used to indicate that scaling should be implemented by changing the number of currently running instances by either a set number or by a given percent; allowed values are 'Number' and 'Percent'. *Scale By Value*: the value in units of *Scale By Type* by which to change the number of currently running instances when scaling. *Rule Type*: used to specify whether the rule is based on metric values or on projected values calculated through a linear regression; allowed values are 'Static' and 'LinearRegression'. *Time Ahead*: for rules based on projected metric values, this element defines how far into the future to extrapolate the linear fit.

Scale out rules have a further element, *Use As QoS* with allowed values of true or false. If it is true then the scaling rule is additionally used by Platform Insights to form a QoS condition to be taken account of by the predictive controller, and in this case a predictive rule must also be submitted with the following elements. *Workload Metric*: a metric representing workload demand exposed on the server and monitored by the system. *Aggregate Function*: the statistical aggregate function to be used for workload metric evaluation; allowed values are 'average', 'sum' and 'median'.

Time Window: the length of the batch time window over which the aggregate function is to be applied to the workload metric. *Min Instances* and *Max Instances*: the minimum and maximum number of instances in the tier. *Confidence Level*: the confidence level used by the time series forecaster to compute confidence bounds on the predictions it makes.

B. Rule Processing

Upon submission of a scaling rule, the sequences of events depicted in Figures 1 and 2 take place. If the scaling rule is to be used as the basis of a QoS condition, then a prediction rule is also submitted, which triggers the creation of data aggregation statements and their registration with the Esper engine. These aggregation statements cover the workload metric aggregated over the batch time window requested in the prediction rule and both the workload and QoS metrics aggregated over the sliding time window requested in the scaling rule. The Esper engine again subscribes for relevant data and outputs aggregated data for use as input to the predictive models.

C. Predictive Models

The predictive auto-scaling algorithm currently employed in Platform Insights comprises three models operating on mid-term and short-term timescales. The first model is a time series forecaster that estimates the workload at some future point in time. The model takes as input the total number of requests made by the application (*Workload Metric* is 'number of requests', *Aggregate Function* is 'sum') and predicts the future workload some time later; typically the prediction horizon is set to one hour (*Time Window* is 'one hour'). The *Confidence Level* is set to 95% and used to calculate confidence intervals on the forecast.

By comparing the forecast value against the four-hour moving average value this model also classifies the current workload trend as 'Increasing' (if >10% difference), 'Decreasing' (<-10% difference) or 'Steady' (otherwise). Changes in enterprise workload demand should be observable over a four hour period since typical workload cycles exhibit daily or weekly trends [18].

The second model is an updateable Naive Bayes model that learns the relation between the current workload per server and the current QoS classification. The QoS classification is a binary classification according to whether or not the QoS condition is met or violated. The threshold for making this classification is set to 5% below the actual QoS target value to reduce the risk of under-provisioning, an approach also adopted by others [19]. This model is used to predict mid-term resource requirements taking as input the forecast confidence interval output from the first model.

The third model is also an updateable Naive Bayes model and it maps the total workload per server to a QoS classification some time into the future, typically 30 minutes

as this will allow time to build confidence in any output auto-scaling requests and to provision additional servers. This model takes as input the current workload and the trend output from the first model. It is used to estimate the optimal number of server instances required to handle the short-term workload, by finding the minimum number of servers such that there is less than 5% chance of QoS violation in the next 30 minutes. If this estimate, N_{Est} , differs to the current number of servers, N , then an auto-scaling decision is made to add $N_{Est} - N$ server instances.

The Weka machine learning library is used to implement the predictive models [20]. One of the main reasons for choosing Weka is that it provides classifiers that are updateable incrementally. Because such classifiers can be updated one training instance at a time, in line with the arrival of the new data, this feature is particularly relevant for Platform Insights in analysing steady streams of monitoring data. Platform Insights uses the time series forecasting plug-in and incrementally updateable Naive Bayes models. An in-depth discussion of the implementation and performance of these algorithms is presented in [21]; the focus of this paper is to demonstrate the feasibility of the approach.

D. Coordination of Controllers

When an application starts running, both controllers are activated. The predictive controller can be used without off-line training because it takes advantage of online incremental learning techniques. If the predictive controller is uncertain of what action to take then it does not make any auto-scaling decisions, instead it continues to learn. In these cases auto-scaling decisions can still be made by the reactive controller since it is always running as a stand-by.

Both controllers are capable of triggering scale in and scale out actions. The reactive controller is only able to do so if sufficient time (*Min Time Between Alerts*) has passed since the previous scaling action and if the landscape state is 'operational'. The predictive controller can submit scaling requests at any time. Both controllers act independently but forward their requests to the centralized *Decision Manager* to deliver a coordinated elasticity mechanism.

The *Decision Manager* validates received requests with information it has access to, or can obtain from the *Landscape State Manager*, regarding the current number of instances running, any outstanding scaling requests currently being executed, and specified limits on the number of servers. This validation process may revise the request in several ways. Firstly, the request may be rejected. This can occur if it duplicates a request already being executed, or if it instructs to scale in whilst a scale out action is currently being executed. We choose to implement a conservative policy stipulating that scale out takes precedence over scale in so as to minimize QoS violations. Secondly, the requested number of servers may be modified. This can happen to enforce the specified limits. It will also happen if a) the

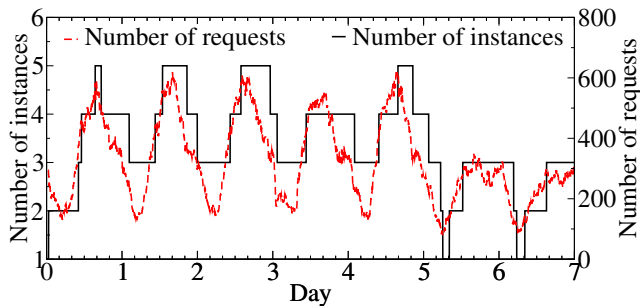


Figure 3. Auto-scaling simulation using ClarkNet traces.

request would result in the number of server instances going outside the mid-term resource requirements predicted from the forecast confidence interval, or b) the request instructs the addition of n_2 servers whilst an earlier scale out action instructing the addition of n_1 servers is still being executed, in which case the request is revised to $\max(n_2 - n_1, 0)$.

Upon successful validation, the *Decision Manager* forwards the (possibly revised) request to the *System Orchestrator* for execution. It also updates the time of the last scaling action and updates the state of the application landscape held by the *Landscape State Manager* to ‘repairing’. The landscape state only returns to ‘operational’ once the *Landscape State Manager* detects the requested change in the number of current running instances, indicating that the request has been successfully carried out.

V. CASE STUDY

To evaluate the performance of the elasticity controller, a simulation of the elasticity of the application server tier has been carried out. Two real datasets, the ClarkNet web server trace logs [4] and the FIFA 1998 Word Cup Access logs [4][5], were used to simulate the incoming load to the system. The log files were summarized to extract the number of requests arriving every 2 minutes and then used to simulate driving the SPECjEnterprise2010 benchmark. The benchmark response times were observed to be in excess of the target time of 2 seconds when the CPU utilization went beyond 80%. A scale out rule was configured as: if the minimum median value of CPU utilization over the past 40 minutes is $> 80\%$ then increase the number of instances by 1. Similarly for scale-in: if the maximum median value of CPU utilization over the past 60 minutes is $< 50\%$ then decrease the number of instances by 1. After auto-scaling, a period of at least the same time window again must pass before the next auto-scaling decision can be made. In the simulation it is assumed that the provisioning of a new instance takes 10 minutes [22]. The QoS condition extracted from the scale out rule was: minimum median CPU utilization over a window of 40 minutes must be $< 80\%$.

The number of requests using the August ClarkNet trace together with the number of simulated running instances are

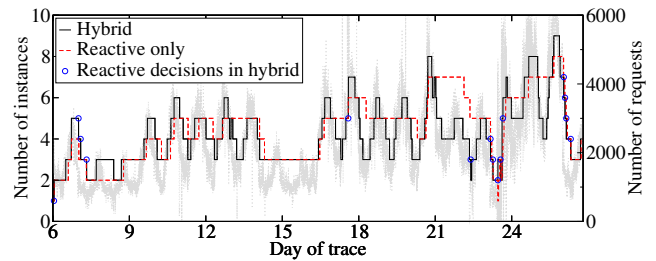


Figure 4. Auto-scaling simulation using 1998 FIFA World Cup trace. Grey line: number of requests in each two minute interval (right axis).

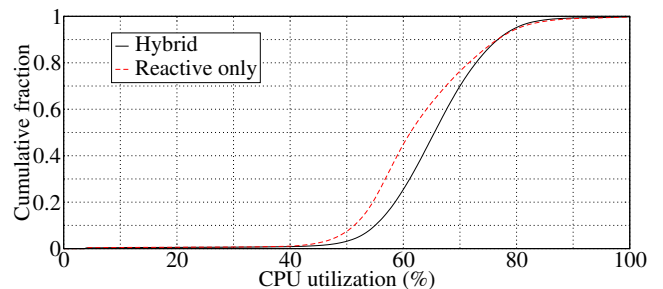


Figure 5. Cumulative distribution of CPU utilization.

shown in Figure 3. In this simulation, the reactive controller was only responsible for the first two scaling actions; all subsequent scaling actions were generated by the predictive controller. The figure demonstrates that the hybrid controller is capable of making appropriate scaling actions and is stable. The QoS metric is monitored throughout the course of the run: less than 1% of all collected QoS values violated the QoS condition. To quantify this fully, there were 28 two minute periods where the QoS metric went above 80%, all of which were at the very start of the run.

The FIFA World Cup access logs exhibit a higher degree of burstiness than do the ClarkNet logs. Figure 4 shows the auto-scaling decisions made by both the hybrid and the purely reactive controllers, for days 6 to 25 of the logs. Day 26 in the figure is a repeat of day 6. The actions initiated by the reactive controller in the hybrid model (blue circles) are generally taken at times when the workload is different to historical workload, and hence the predictive controller has not built sufficient confidence in its online models, or when the workload exhibits more burstiness than normal, highlighting the advantage of operating the reactive and predictive controllers in a coordinated parallel manner. The figure suggests that the hybrid controller dynamically adjusts resource more appropriately, and hence will result in better utilization, than does the purely reactive controller. Figure 5 verifies this: CPU utilization is consistently maintained within the target range of 50% to 80% ($< 50\%$ less than 3% of the time and $> 80\%$ less than 5% of the time). Again, there are only a few QoS violations: 41 in total, which relate to 3 separate incidents characterized by bursty workloads.

VI. CONCLUSION AND FUTURE WORK

In this paper, we described the architecture, design and implementation of a new real-time cloud capacity framework, Platform Insights. Platform Insights is a hybrid elasticity controller employing both reactive rule-based and predictive model-based elasticity mechanisms together in a coordinated manner. The approach has been validated by using traces based on two real datasets to simulate driving a benchmark application. In both cases, Platform Insights was able to provision resource for the application server tier more appropriately than the reactive controller alone, yielding very few QoS violations and maintaining consistently high CPU utilization.

In the short-term, we will carry out further comparisons of our approach with other auto-scaling methods and integrate our framework in a real cloud infrastructure. For future work, we intend to extend Platform Insights to handle multiple QoS objectives at once and to incorporate an algorithm to detect change in workload mix.

ACKNOWLEDGMENT

This research is funded by the CumuloNimbo project in the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. FP7-257993.

REFERENCES

- [1] Gartner, "Market Trends: Platform as a Service, Worldwide, 2012-2016, 2H12 Update," <http://www.gartner.com/id=2188816> Accessed 12th March 2013.
- [2] IDC, "Quantitative Estimates of the Demand for Cloud Computing in Europe and the Likely Barriers to Up, SMART 2011/0045, Final Report," 2012 http://ec.europa.eu/information_society/activities/cloudcomputing/docs/quantitative_estimates.pdf Accessed 12th March 2013.
- [3] L. Schubert and K. Jeffery, "Advances in Clouds: Research in Future Cloud Computing," Expert Group Report, Public Version 1.0, 2012 <http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf> Accessed 12th March 2013.
- [4] The Internet Traffic Archive <http://ita.ee.lbl.gov/html/traces.html> Accessed 12th March 2013.
- [5] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site," Technical Report HPL-1999-35R1, HP Laboratories, 1999. The trace is available from [4].
- [6] T. Lorido-Bostrán, J. Miguel-Alonso and J.A. Lozano, "Auto-scaling Techniques for Elastic Applications in Cloud Environments," Technical Report EHU-KAT-IK-09-12, 2012.
- [7] Amazon Elastic Compute Cloud <http://aws.amazon.com/ec2/> Accessed 12th March 2013.
- [8] RightScale <http://www.rightscale.com/> Accessed 12th March 2013.
- [9] AzureWatch <http://www.paraleap.com/azurewatch> Accessed 12th March 2013.
- [10] D. Breitgand, E. Henis and O. Shehory, "Automated and adaptive threshold setting: enabling technology for autonomy and self-management," in *Proceedings of the 2nd International Conference on Automatic Computing (ICAC '05)*, 2005, pp. 204-215.
- [11] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. Jordan and D. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters," in *Proceedings of the 2009 conference on Hot topics in cloud computing (HotCloud'09)*, 2009, Article No. 12.
- [12] R.N. Calheiros, R. Ranjan and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," in *Proceedings of the 2011 International Conference on Parallel Processing (ICPP '11)*, 2011, pp. 295-304.
- [13] N. Roy, A. Dubey and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, 2011, pp. 500-507.
- [14] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 3, no.1, pp. 1-39, 2008.
- [15] W. Iqbal, M.N. Dailey, D. Carrera and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871-879, 2011.
- [16] A. Ali-Eldin, J. Tordsson and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *Network Operations and Management Symposium (NOMS)*, IEEE, 2012, pp. 204-212.
- [17] EsperTech Event Stream Intelligence <http://esper.codehaus.org/index.html>, a product of EsperTech Inc. Accessed 12th March 2013.
- [18] D. Gmach, J. Rolia, L. Cherkasova and A. Kemper, "Resource pool management: Reactive versus proactive or let's be friends," *Computer Networks*, vol. 53, no. 17, pp. 2905-2922, 2009.
- [19] Z. Gong, X. Gu and J. Wilkes, "Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management (CNSM)*, 2010, pp. 9-16.
- [20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.
- [21] L.R. Moore, K. Bean and T. Ellahi, "Transforming reactive auto-scaling into proactive auto-scaling,". Accepted for publication, 2013.
- [22] A. Li, X. Yang, S. Kandula and M. Zhang, "CloudCmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10)*, 2010, pp. 1-14.

Elastic-TOSCA: Supporting Elasticity of Cloud Application in TOSCA

Rui Han, Moustafa M. Ghanem, Yike Guo*

Department of Computing
Imperial College London
London, UK
{r.han10, mmg, y.guo}@imperial.ac.uk

Abstract— The Topology and Orchestration Specification for Cloud Applications (TOSCA) is an emerging framework aiming at enhancing the portability of cloud applications by standardizing their life cycle management in a vendor-neutral way. TOSCA captures the description of cloud application and infrastructure services, the relationships between parts of the services, and the operational behavior of these services (e.g., deploy, patch, shutdown). However, it lacks support for the equally important aspect of managing elasticity, i.e., managing the dynamic scaling of cloud applications at run-time. In this work we present the Elastic-TOSCA framework, which extends TOSCA to address this issue. We then describe how Elastic-TOSCA can be used to support a variety of analytical model-based approaches for elasticity management in complex cloud applications. We further provide a detailed example describing how Elastic TOSCA can be used to support easily a dynamic scaling approach based on a queueing system model. Using a case study for managing the elasticity of a multi-tier e-commerce service, we demonstrate the effectiveness of both the Elastic-TOSCA framework and the scaling approach used.

Keywords-TOSCA; cloud; elasticity; scaling approaches; queueing system

I. INTRODUCTION

Cloud computing has gained unquestionable commercial success in recent years. Key value propositions promoted by cloud IaaS (Infrastructure-as-a-Service) providers such as Amazon AWS (Amazon Web Services) [1] and GoGrid [2] include the user's ability to scale up or down resources used based on their computational demand, thus letting *application owners* (software service creators or developers) pay only for the resources used. This model is appealing for deploying complex applications that provide services for third parties or end users. Some examples of such services include traditional e-commerce sites, online healthcare applications, gaming applications, and media applications. In such applications, if the workload of the service increases (e.g., more end users start submitting requests simultaneously), the application owner ideally needs to scale up the resources used to maintain the Quality of Service (QoS) offered to the end users. When the workload eases down, the application owner ideally needs to scale down the resources used to reduce the cost incurred for service provision. Within this context, supporting dynamic (on-demand) scaling, also known as elasticity, has become one of

the most important features that need to be supported in a cloud platform.

The Topology and Orchestration Specification for Cloud Applications (TOSCA) is an emerging framework for describing components' dependencies and deployment plans for cloud applications. Proposed by the Organization for Advancement of Structured Information Standards (OASIS) [3, 4], TOSCA is designed to simplify the life cycle management of cloud applications in a vendor-neutral way so as to enhance their portability. Such portability is enabled through specifying the operational behaviours of cloud applications, e.g., how servers are deployed or removed and how they are connected, in a uniform way independent of the cloud platform used. This uniform description provides application owners with flexibility when deploying and migrating their applications and associated components across different IaaS providers.

Currently, TOSCA supports the specification of key activities required for the initial deployment of cloud applications and also the activities required to shut down the application. However, it does not provide support for the equally important aspect of specifying how application elasticity can be managed at run-time, e.g., by enabling the specification of how resources can be added or removed at run-time based on workload variation. Our motivation in this paper is to enrich and extend the existing TOSCA framework to support such elasticity management activities in a vendor-neutral way. In particular, our contributions are summarized as follow:

Elastic-TOSCA: We provide extensions to TOSCA that support the specification of dynamic scaling plans and that enable guiding scale-up/down of cloud applications at run-time.

Supporting model-based application scaling approaches using Elastic-TOSCA: The Elastic-TOSCA framework is generic and can support a wide class of dynamic scaling approaches based on analytical models [1, 2, 5-14]. The key requirement for using the framework is that the implemented scaling approach should be able to access its inputs from the Elastic-TOSCA server templates, and to feed its outputs to the template that enables the auto-scaling of applications. We describe and demonstrate how this can be achieved easily using a scaling approach based on a queueing system model [5].

Example implementation and evaluation: We extend the Imperial Smart Scaling engine (iSSe) [10, 15], an intelligent platform designed to automate the deployment and scaling process of cloud applications, to support the Elastic-

* Please direct your enquiries to the communication author Professor Yike Guo.

TOSCA framework. Using iSSE, we evaluate the effectiveness of both Elastic TOSCA and the proposed scaling approach using a multi-tier e-commerce service as an example application.

The remainder of this paper is organized as follows: Section II presents basic concepts on cloud applications and how auto-scaling of such applications can be achieved using analytical model-based approaches. Section III defines the Elastic-TOSCA framework and describes its key components. Section IV describes how a queueing system model-based approach can be supported by Elastic-TOSCA. Section IV introduces the architecture of the iSSE and describes the extensions implemented to support Elastic-TOSCA. It also provides an experimental evaluation of the proposed approach. Finally, Section VI summarises the work presented in this paper and describes avenues for future research.

II. BACKGROUND

In this section, we first describe the structure of a traditional multi-tier application to illustrate how many applications benefit from dynamic scaling in a cloud environment and then discuss existing dynamic scaling techniques that are based on analytical models.

A. Illustrative Example of Multi-tier Cloud Applications

The main objective of this paper is to investigate the enrichment of TOSCA to support elasticity management of cloud applications. Addressing this issue effectively requires taking a closer look at the structure of common services and applications that can benefit from dynamic scaling when deployed on IaaS clouds so as to cope with varying workloads. Many such services are typically complex multi-tier applications running on distributed software platforms. Figure 1, shows the logical structure of one such application implemented using four tiers of servers: a frontend HAProxy load balancer for accepting and distributing end users' requests, an Apache web server for handling HTTP requests; a middle-tier Tomcat application server for implementing business logic; and a backend database with data store and processing. These servers work together to handle end users' requests. Depending on the application workload, the servers at each tier can be stressed at different times and the implementation ideally needs to scale up or down the resources at the appropriate tier so as to maintain the overall QoS requirement of the application while minimizing the cost of resources used.

Figure 2(a) shows the lifecycle of the e-commerce application as an example of such dynamic scaling. When the application is initially deployed (see Figure 1), five servers are deployed to support a small number of customers. If the demand increases, the application can be scaled up to add new servers. For example, in the scaling up of Figure 2(b), one Apache server and two Tomcat servers are added to maintain performance. Alternatively, if the demand decreases, some servers can be removed to reduce the cost of service provision. For example, in the scaling

down of Figure 2(c), one Tomcat server is removed from Figure 1's initially deployed application.

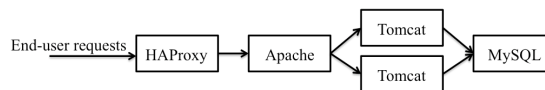


Figure 1. An example multi-tier cloud application.

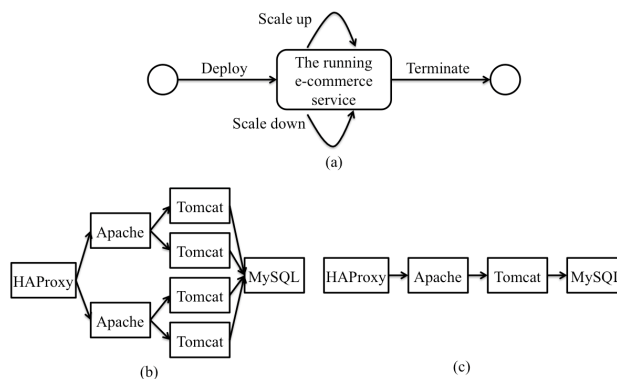


Figure 2. An e-commerce service. We can see (a) the lifecycle of a e-commerce service, (b) the service after a scaling up, and (c) the service after a scaling down.

It should be noted that most scaling approaches for cloud applications, whether used in practice or described in the literature [1, 2, 5-14], are typically based on controlling (increasing or decreasing) the number of Virtual Machine (VM) instances that host the applications' server components. Without loss of generality, we assume that each server component is installed in a stand-alone VM. Accordingly, the scaling up/down the application discussed in this work typically involves adding/removing extra software servers, and hence extra VMs in a cloud environment.

B. Cloud Scaling Techniques Using Analytical Models

A variety of approaches that are suitable for auto-scaling multi-tier applications have been proposed in the literature. Many of those employ analytical modelling techniques based on queueing systems [5]. For example, in [6], Xiong et al. model an application by a network of queueing systems and conduct the performance analysis to show relationships among workloads, number of servers and QoS level. In [7], Bacigalupo et al. model an application by a queueing network with three tiers, namely application, database and database disk tiers. Each tier is then solved to analyse the mean response time, throughput and utilisation of a server. In [8], Bi et al. break down an application's end-to-end response time to each tier. They then calculate the number of servers allocated at each tier subject to constraints on the average response time and arrival rate. In [9], Hu et al. consider two allocation strategies using queueing systems: 1) shared allocation (SA) strategy where all incoming requests have the same queueing; 2) dedicated allocation (DA) strategy where requests with different arrival rate are divided into multiple queues. An algorithm is then proposed to

decide which strategy (SA or DA) results in a smaller number of servers being used to satisfy the QoS requirement. In addition, in [10], Han et al. consider the cost of VMs and introduce cost-aware criteria to detect and analyse the bottlenecks of multi-tier applications. They then present an adaptive scaling algorithm to lower cost by scaling up or down only at the bottleneck tiers. In [11], Pal et al. propose a pricing framework with economic models designed for multiple cloud providers in the marketplace, where each IaaS provider is modelled as a queueing system. Using this queueing system, the framework aims at informing application owners of the available price and its related QoS level.

Various other stochastic model-based approaches have been studied and used. For example, in [12], Ghosh et al. divide an application into three types of sub-analytical model: the resource provisioning decision model, VM provisioning model and run-time model. By iteratively solving each individual sub-model, their analysis obtains two results: response time and service availability. In addition, in [13] Ghosh et al. utilise a stochastic reward net to model an application and provide two analysis results: job rejection rate and response delay. In [14] Li et al. use a network flow model to analyse applications and introduce an approach to assist service providers in making a trade-off between cost and QoS requirements.

It should be noted that the analytical models described here are mainly based on mathematical representations of the application and servers used. Their use in practice requires capturing the structure of the application itself to generate the mathematical representation. Furthermore, their implementation also needs interfacing with the run-time system so as to obtain the parameters used in the models at run-time and also to guide the system in implementing the computed scaling decisions.

III. ELASTIC-TOSCA

In this section, we first introduce the basic TOSCA framework briefly, and then describe how it is extended to define Elastic-TOSCA.

A. Basic Introduction of TOSCA

TOSCA server templates are described in XML and can be used for describing cloud application, including server components and their linking relationships [3, 4]. Figure 3 shows the high-level structure of a TOSCA server template describing an e-commerce service with four sections: Topology template, Node types, Relationship type and Plans. The “Topology template” section specifies the dependency between different server components. The “Node types” section defines the properties of one server, e.g., its owner and the configuration of its hosted VM (CPU numbers, memory size, disk capacity and operating system). A “Relationship type” section specifies the relationship between two servers. In the shown example, an Apache server and a Tomcat server are connected, where the Apache is the source node and the Tomcat is the target node. Finally, the “Plans” section defines the process model for initially

deploying a new application and also for removing a running application.

B. Elastic-TOSCA: Extensions to Support Elasticity

We extend the basic TOSCA framework and enrich it with the information required for guiding dynamic scaling of cloud applications, allowing application owners to specify different scaling strategies. For example, an owner could define a scaling up/down strategy based on performance requirements, budgets and QoS requirements specified in service-level agreements (SLAs).

Using the Elastic-TOSCA framework, we generate a new Elastic-TOSCA-based XML document that includes monitoring information structures and new plans for scaling up/down. Figure 4 shows an example server template in Elastic-TOSCA, including two new sections (“Monitoring Information” and “SLA&Constraints”) as well as extensions to the “Plans” section, corresponding to three components needed for guiding dynamic scaling of an e-commerce service. Note that the specification and extension of these sections follows TOSCA extensibility mechanism [3, 4], which guarantees that the extended sections are independent of cloud IaaS providers.

The “Monitoring Information” section mainly specifies a running application’s current status and underlying infrastructures. In the example fragment in Figure 5, this section records the detected response time and the request arrival rate, as well as the utilisation of resources of the application’s hosted VMs.

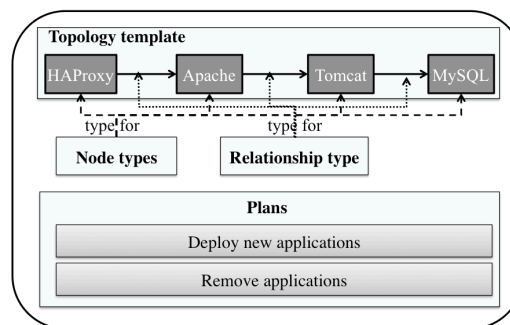


Figure 3. An example server template in basic TOSCA

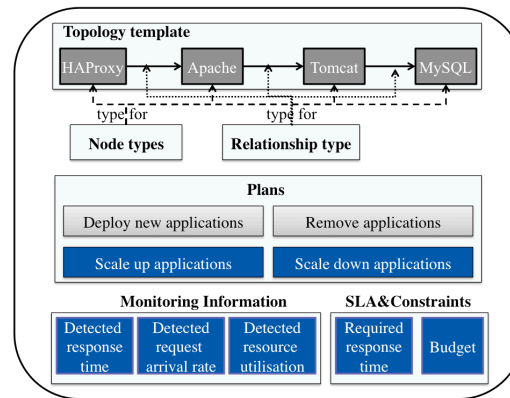


Figure 4. An example server template in Elastic-TOSCA.

```
<MonitoringInformationTemplate id="ApplicationMonitor"
  name="Application Monitor"
  InformationType="Monitor">
  <DetectedResponseTime>1.5Seconds</DetectedResponseTime>
  <RequestArrivalRate>100RequestsPerSec</RequestArrivalRate>
  <ResourceUtilisation>
    <CPU>80Percentages</CPU>
    <Memory>75Percentages</Memory>
    <I/O>50Percentages</I/O>
  </ResourceUtilisation>
</MonitoringInformationTemplate>
```

Figure 5. An example “Monitoring Information” section in Elastic-TOSCA.

The “SLA&Constraints” section describes QoS requirements and any constraints on quality, budget, and other aspects of the application. In the example shown in Figure 6, this section specifies the end users’ required QoS: the maximal response time and the application owner’s constraints: the minimal resource utilisation (a resource is considered as idle if its utilisation is smaller than this requirement) and budget (the maximal cost to support the running of the service).

Finally, we extend the “Plan” section in basic TOSCA to define more types of plans that handle the application’s dynamic scaling cases. Figure 4 shows the Elastic-TOSCA definition of two types of scaling plans — “Scale up applications” and “Scale down applications”. Each type can have multiple scaling plans and each plan describes a specific scaling scenario. For example, Figure 7 shows a fragment of a plan for scaling up an e-commerce service. This plan is used for adding one Apache server and two Tomcat servers to the application.

A scaling plan in Elastic-TOSCA server template defines a list of scaling tasks, where each task corresponds to a deployment action. Based on Elastic-TOSCA, this deployment action is independent of any cloud platform, thus enabling applications an IaaS-neutral scaling process. In Figure 8, we provide two example segments in Elastic-TOSCA server templates for specifying a deployment action in two different cloud platforms. These specifications contain all the parameters needed to call an auto scaling API of IC-Cloud [16] (Figure 8(a)) and Amazon AWS [1] (Figure 8(b)) in order to deploy a new Tomcat server when scaling up. Note that for each scaling case, a scaling plan and its scaling tasks are generated dynamically. The information needed to generate documents describing the scaling tasks (e.g., a server’s user name, password and VM configuration) is obtained from the “Node types” section of Elastic-TOSCA.

```
<SLAAndConstraintsTemplate id="ApplicationSLA"
  name="Application SLA"
  InformationType="SLA">
  <MaximalResponseTime>2Seconds</MaximalResponseTime>
  <MinimalResourceUtilisation>30Percentages</MinimalResourceUtilisation>
  <Budget>10EUR</Budget>
</SLAAndConstraintsTemplate>
```

Figure 6. An example “SLA&Constraints” section in Elastic-TOSCA.

```
<Plan id="ScalingUpAnE-CommerceApplication"
  name="A Scaling Up Plan of E-commerce Website"
  planType="http://docs.oasis-open.org/tosca/ns/2013/01/...."
  languageUsed="http://www.omg.org/spec/BPMN/2.0/">
  <PlanModel>
    <process name="DeployNewApplication" id="p1">
      <task id="t1" name="DeployAnApacheServer"/>
      <task id="t2" name="DeployATomcatServer"/>
      <task id="t3" name="DeployATomcatServer"/>
      <sequenceFlow id="s1" targetRef="t2" sourceRef="t1"/>
      <sequenceFlow id="s2" targetRef="t3" sourceRef="t2"/>
    </process>
  </PlanModel>
</Plan>
```

Figure 7. An example scaling up plan in Elastic-TOSCA.

IV. SUPPORTING SCALING APPROACHES BASED ON ANALYTICAL MODELS USING ELASTIC-TOSCA

In this section, we first explain the basic steps of scaling approaches using analytical models and how these steps are supported by Elastic-TOSCA. We then employ a queueing system as a typical example of analytical model to demonstrate these steps.

A. Analytical Model-based Scaling Approaches Using Elastic-TOSCA

Typically, an analytical model-based approach for scaling an application consists of four steps, which are preformed using information maintained in different sections of Elastic-TOSCA server templates as illustrate in Figure 9.

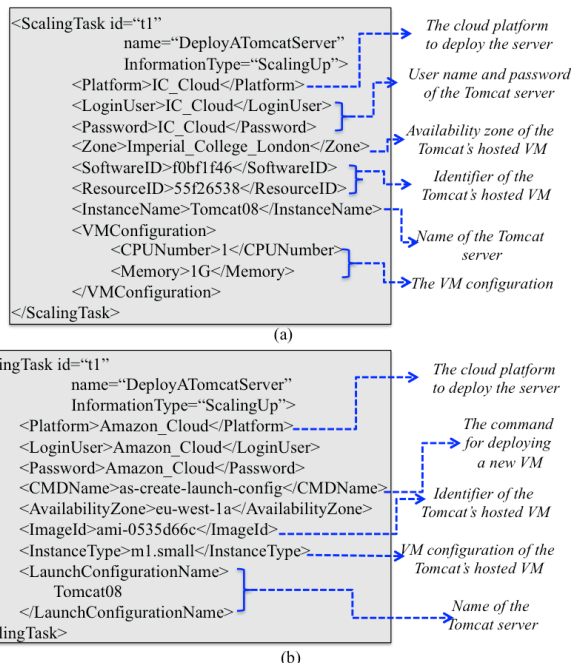


Figure 8. Two example scaling tasks for deploying a new server in two cloud platforms: (a) IC-Cloud and (b) Amazon AWS.

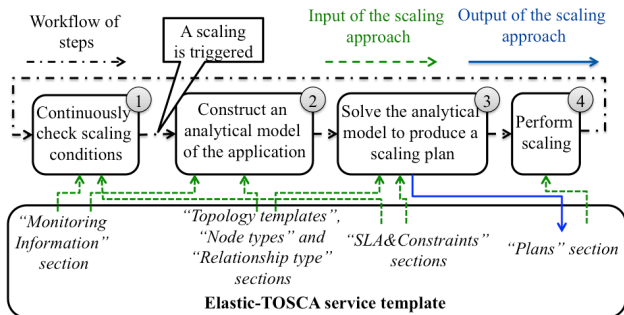


Figure 9. Elastic-TOSCA server templates: for supporting scaling approaches using analytical model.

At step 1, the approach continuously checks the “Monitoring Information” and “SLA&Constraints” sections to check where a scaling up/down is needed. The approach proceeds to step 2 if scaling up/down is triggered. At step 2, an analytical model is constructed according to the application topology, configurations of servers and their linking relationships. At step 3, the approach employs analytical modelling techniques to transform the high level QoS requirements into the number of servers to be deployed, and generates a scaling plan for meeting QoS requirements. Finally, step 4 executes the scaling plan.

B. Basic Introduction to Queueing Systems

Typically, a queueing system can be described using $A/S/n$, where A represents the arrival process, S represents the distribution of service time and n is the number of servers [5]. In the example queueing system of Figure 10(a), $A/S/n = G/G/n$ (G for general). This $G/G/n$ queueing system includes n parallel and independent servers and one request waiting queue, where both requests’ interarrival time λ (the reciprocal of requests’ arrival rate) and servers’ service time follow arbitrary distributions. The $G/G/n$ queueing system of Figure 10(a) is used to model a tier of Tomcat servers. Furthermore, the whole m -tier application is modelled as a network of m $G/G/n$ queueing systems and each queueing system represents a tier in the application. Take Figure 10(b)’s 4-tier e-commerce service as an example, which is modelled by a queueing network of four $G/G/n$ queueing systems. The queueing system of the first tier (HAProxy servers) only receives requests from end users, and the departure requests of one tier are the incoming requests of its following tier.

C. Scaling Approach Using Queueing Systems

Queueing theory [5] has been successfully applied in many cloud scaling algorithms [5-11] to perform capacity planning for dynamic scaling. Typically, the overall scaling approach can be described in Figure 11’s pseudocode. This scaling approach described in this pseudocode corresponds to Figure 9’s four generic steps:

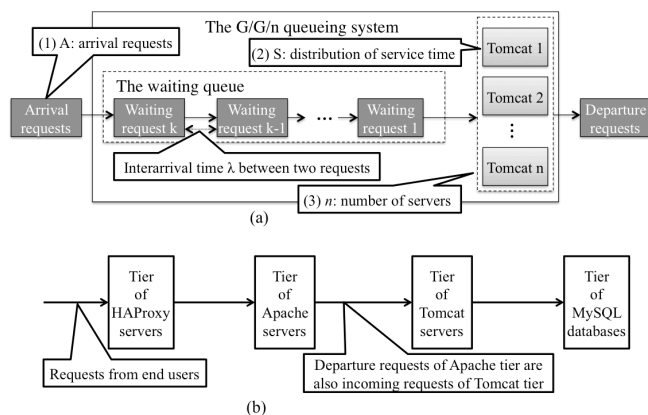


Figure 10. Queueing model and networks. We can see (a) a $G/G/n$ queueing system to describe a tier of Tomcat servers, and (b) a queueing network to describe a e-commerce service.

At step 1 (line 3 and 4), the approach decides whether to trigger a scaling up/down according to the latest monitoring information of the running application maintained in the “Monitoring Information” section of Elastic-TOSCA server templates and QoS requirements and constraints to be satisfied in the “SLA&Constraints” Section. For example, if the monitored response time (e.g., 3 seconds) exceeds the maximal required response time (e.g., 2 second), a scaling up is triggered. In contrast, if the detected resource utilisation (e.g., 20%) is below the minimal resource utilisation (e.g., 30%), a scaling down is conducted to remove some idle servers.

At step 2 (line 6 and 10) and step 3 (line 7 and 11), the approach applies a queueing network to model the application and generate a scaling up or down plan (line 7 and 11). This plan is then added to the “Plan” section of Elastic-TOSCA server templates. Section IV.D explains these two steps in detail.

Finally, at step 4 (line 8 and 12), the approach performs the scaling according to the generated plan.

```

A dynamic scaling approach using queueing systems
Input: A  $m$ -tier application, QoS requirements and constrains.
1. Begin
2. while (the application is not terminated)
3.   Monitor the application’s running status and its underlying infrastructures
4.   Check the QoS requirements and constraints to be satisfied
5.   if a scaling up condition is met, then
6.     Construct a queueing network as the analytical model
7.     Conduct capacity estimation to generate a scaling up plan
8.     Perform the scaling up to add servers
9.   else if a scaling down condition is met, then
10.    Construct a queueing network as the analytical model
11.    Conduct the capacity estimation to generate a scaling down plan
12.    Perform scaling down to remove servers
13. End
    
```

Figure 11. Pseudocode of a dynamic scaling approach using queueing systems.

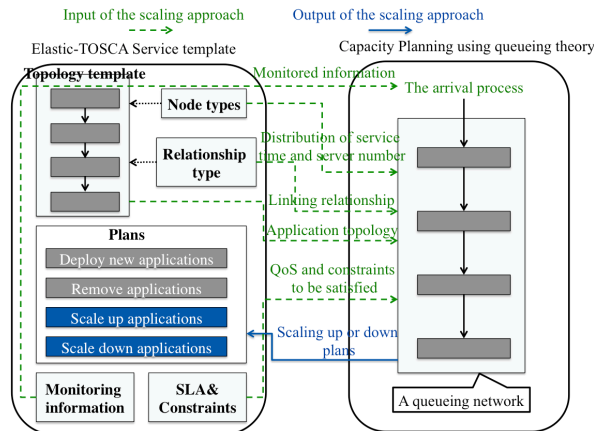


Figure 12. The capacity planning using server templates in Elastic-TOSCA and queueing systems.

D. Supporting the Scaling Approach using Elastic-TOSCA server templates

The key component of a scaling approach using queueing systems as an analytical modelling technique is the capacity planning. This planning involves two steps: constructing a queueing network of a multi-tier application (step 2 in Figure 9) and solving the analytical model to generate a scaling plan (step 3 in Figure 9).

Figure 12 illustrates how the information maintained in a server template of Elastic-TOSCA maps to the corresponding part of a queueing network (dashed lines). The information maintained in the “Topology template” and “Relationship type” sections describe the topology of the queueing network and the linking relationship of different queueing systems. For each queueing system, $A/S/n$, the information in “Monitoring Information” section specifies the arrival process A , the information in the “Node types” section decides distribution of service time S and server number n .

After a queueing network representing the multi-tier application is constructed, the approach applies queueing theory to perform capacity planning using information in the “Monitoring Information” and “SLA&Constraints” sections. This planning estimates the number of servers to be deployed at each tier of the application and generates a scaling plan. The plan is then added to the “Plans” section to guide the scaling of the application (solid lines).

Take the e-commerce service in Figure 1 for example, using information in the “Topology templates” and “Relationship type” sections, the approach first constructs a queueing network of four queueing systems to describe the four tiers of servers in the service, and decides the linking sequence of these four queueing systems. The “Monitoring Information” and “Node type” sections then decide the three components of each queueing system. For example, the queueing system $A/S/n$ of Tomcat servers has arrival requests A with arrival rate 150 requests/second, each Tomcat server has service rate 70 requests/second, and the number of Tomcat servers is 1. Using the constructed queueing network, capacity planning is conducted according to the detected response time (3.5 seconds) in the

“Monitoring Information” section and the required response time (2.0 seconds) in the “SLA&Constraints” section. The detected response time is larger than the required one, so a scaling up plan is generated: the tier of Tomcat should be added two servers and the tier of the Apache should be added one server.

V. IMPLEMENTATION AND EVALUATION

In this section, we first introduce iSse and describe how it has been extended to interact with Elastic-TOSCA and the analytical models. We then describe the experiments conducted to illustrate both the effectiveness of the queueing system based scaling approach and the interaction of iSse with Elastic-TOSCA.

A. Extension of iSse to Support Elastic-TOSCA

We extended iSse (see [10, 15] for detail), an intelligent scaling engine, to support the Elastic-TOSCA framework. As shown in Figure 13, iSse acts as middleware between cloud IaaS providers and application owners. It provides a *Application owner portal* to assist application owners to configure their services, allow them to select servers from the iSse Repository of Servers, define VM configurations, and design their topology. This portal also allows them to specify the required QoS and constraints. To enable interaction with Elastic-TOSCA, the information is stored in the “Topology template”, “Node types”, “Relationship types” sections in Elastic-TOSCA server templates.

The iSse *Monitoring service* monitors each running application using two types of monitors. The first is the entry monitor, which examines the incoming requests over a finite interval (e.g., 60 seconds) and records information such as the requests’ arrival rate and average response time. This information is used to decide whether a scaling up/down is needed. The second is the server monitor installed on each server to monitor its resource usage (e.g., CPU utilisation), and to analyse tier-specific values, such as response time. The collected information is then used to update the “Monitoring information” section in Elastic-TOSCA server templates.

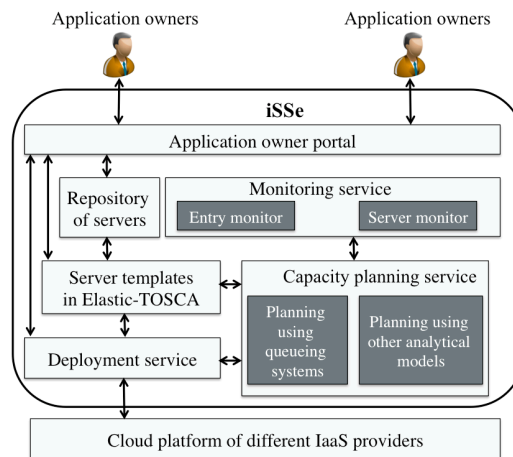


Figure 13. iSse for supporting dynamic scaling using Elastic-TOSCA.

Note that the iSse monitors are generic and independent of the IaaS providers. Existing providers usually provide users with standalone VM images. iSse packages the VM images with pre-developed monitors as server templates that can be deployed in any cloud platform. The iSse monitors hence only depend on the underlying operating systems, e.g., Linux Ubuntu and Centos, supported by iSse. When a scaling is triggered, the iSse *Capacity planning service* applies a scaling approach to generate a scaling up/down plan. When using Elastic-TOSCA, the information is maintained in the server templates. More concretely, the approach estimates the type and number of servers to be scaled, and then updates the “Plans” section by adding the generated scaling plan. Using the generated scaling plan, the iSse *Deployment service* implements the required actions by calling interfaces of the underlying cloud platforms.

B. Evaluation of the Scaling Approach

Our evaluation is designed to illustrate the feasibility and effectiveness of Elastic-TOSCA in enabling dynamic scaling using a queueing system. The scaling approach itself is described in detail in [10], where it had been applied in an iSse version not based on Elastic-TOSCA.

Our experiments are conducted in a data centre running IC-Cloud platform [16]. The configuration used has four physical machines (PMs), each with eight CPUs and 32 GB memory. The version of each processor is Quad-Core AMD Opteron(tm) Processor 2380, with 2.5GHz clock frequency and 512 KB cache size. All four PMs share a 4.1 Tb centralised storage and are connected through a switched gigabit Ethernet LAN with speed 1000mbs.

The e-commerce service in Figure 1 was implemented and its scaling up and down was tested. For convenience, each server of the service is installed on a single dedicated VM running Linux Centos 5.4. In deployment, different servers have different VM configuration details, as listed in Table I. Two versions of the MySQL database (Master and Slave) are implemented to support a data replication model. A MySQL Master is initially deployed and, when the tier of MySQL is scaled up, extra MySQL Slaves are added and configured with replication from the MySQL Master. Given a fixed VM configuration, the deployment of the Tomcat and Apache servers can be completed in a constant time. In the evaluation, the database has a fixed amount of data to be replicated, i.e., the data replication time of MySQL slave is fixed. Thus, the deployment time of MySQL databases is also a constant time.

TABLE I. FIVE TYPES OF SERVERS’ VM CONFIGURATIONS

Service name	CPU	RAM (GB)	Software version
HAProxy	2	2	haproxy-1.4.8
Apache	2	2	Apache 2.2.20
Tomcat	1	1	Tomcat 7.0.22
MySQL Master	4	4	MySQL 5.5
MySQL Slave	1	1	MySQL 5.5

We used a client emulator to simulate a number of concurrent end users. Each end user continuously generates a sequence of requests to stress the server-side application. We divide the test into nine periods, where each period lasts 600 seconds. The first five periods of simulations stepwise increase in the number of end users so as to initiate scaling up. The remaining four periods gradually decrease this number to trigger scaling down. More concretely, the number of simulated concurrent users in the nine periods are: 200, 400, 600, 900, 1200, 900, 600, 400, and 200, respectively. This variance of end user numbers denotes the changing workload volume. The first testing period starts at time = 0 second. During the whole testing period, the application is monitored once every 60 seconds and Figure 14(a) displays the observed arrival rates of incoming requests. These observed arrival rates can be used to derive the mean and variance values of the request’s interarrival time used in the queueing system.

Figure 14(b) lists the numbers of servers at each tier during scaling. Note that the numbers of HAProxy servers and MySQL Master database do not change. For the first period (the number of concurrent users is 200), the e-commerce service is initially deployed with one HAProxy, Apache, MySQL Master server and two Tomcat servers. When the concurrent users increase to 400 at time = 600 seconds and saturate the Apache and Tomcat tiers, dynamic scaling is triggered and one Apache and two Tomcats servers are added. When the number of concurrent users is increased at time=1200, 1800 and 2400 seconds, the cycle repeats. In contrast, when this number decreases at time =3000, 3600, 4200 and 4800 seconds, the service is scaled down by removing idle servers.

Note that, once scaling up or down is triggered, the construction of the application’s queueing network model and executing the capacity planning are completed within few seconds to generate a scaling plan. Using the plan, servers are added or removed in parallel using the iSse Deployment service. In the IC-Cloud platform [16], the deployment actions are completed within 1 or 2 minutes.

In the evaluation, we checked the monitoring information (request arrival rate and response time) every 60 seconds. We can observe in Figure 14 that there are 10 observation values for response time in each test period of 600 seconds. Typically, in each scaling up the first and second observed response time values violate the required constraint because scaling up is not yet completed. In other words, response recovery can be detected only after 1 to 2 minutes.

Figure 14(c) demonstrates the fluctuation of the end-to-end response time observed in the nine testing periods. In the first five periods, the response time is violated whenever the number of concurrent users is increased. For instance, when the number of users is increased to 400 at time = 600 seconds it saturates the Apache and Tomcat servers. Scaling up is then triggered and two Tomcat and one Apache servers are added. In contrast, in the last four periods, the scaling approach scales down the service while meeting the required response time.

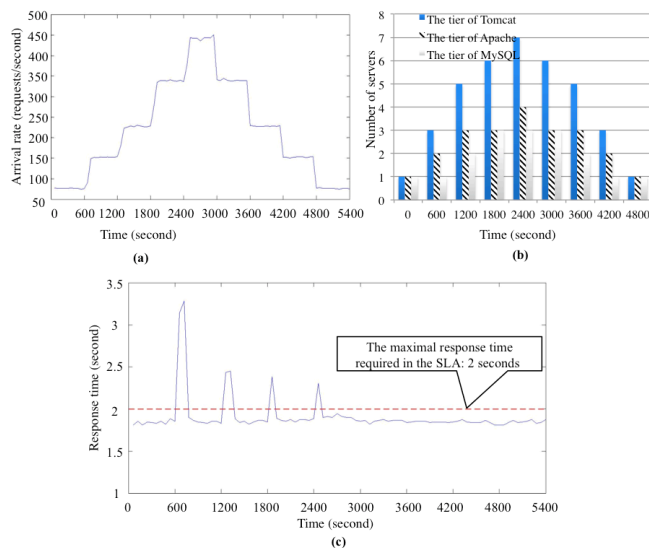


Figure 14. Evaluation of the scaling approach: (a) requests' arrival rate, (b) number of servers in each tier, (c) the end-to-end response time.

Result. The Elastic-TOSCA framework is able to support the scaling approach based on queueing systems for dynamically scaling up and down cloud applications to meet their QoS requirements.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented extensions to the TOSCA framework to enable platform-independent specification of dynamic scaling for cloud applications. The extensions covered three sections, corresponding to three types of information used in guiding dynamic scaling. The Elastic-TOSCA framework is generic and supports a wide class of analytical mode-based scaling algorithms. We illustrated the effectiveness of Elastic-TOSCA framework by using a scaling approach based on a queueing system model. We also described how the framework can be supported easily in a scaling engine and conducted experiments to demonstrate the practicality of the framework using an example e-commerce service.

Our direct future work is to evaluate supporting Elastic-TOSCA on different cloud platforms and also to evaluate using other scaling techniques and approaches such as lightweight scaling at the VM level itself (CPUs, memory, I/O, etc) [17]. We will also test our approach using more complex scenarios such as considering the amount of data to be replicated in the MySQL slave databases in the e-commerce application, as well as by using other multi-tier applications.

REFERENCES

[1] Amazon Web Services (Amazon WS): <http://aws.amazon.com/ec2/> [retrieved: 03, 2013]
 [2] GoGrid: <http://www.gogrid.com/> [retrieved: 03, 2013].

[3] Binz, T., Breiter, G., Leyman, F. and Spatzier, T., "Portable Cloud Services Using TOSCA," *Internet Computing, IEEE*, vol. 16, pp. 80-85, 2012.
 [4] Topology and Orchestration Specification for Cloud Applications (TOSCA), OASIS specification: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca [retrieved: 03, 2013].
 [5] R. B. Cooper, *Introduction to queueing theory*, second edition ed. New York: North-Holland, 1981.
 [6] K. Xiong and H. Perros, "Service performance and analysis in Cloud computing," in *2009 Congress on Services - I*, Los Angeles, CA 2009, pp. 693-700.
 [7] D. A. Bacigalupo, et al., "Managing dynamic enterprise and urgent workloads on clouds using layered queuing and historical performance models," *Simulation Modelling Practice and Theory*, vol. 19, pp. 1479-1495, 2011.
 [8] Bi, jin, Zhu, Zhiliang, Tian, Ruixiong and Wang, Qingbo, "Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in Cloud Data Center," in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD'10)*, Miami, Florida, 2010, pp. 370-377.
 [9] Hu, Y., Wong, J., Iszlai, G. and Litoiu, M., "Resource provisioning for cloud computing," in *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research (CASCON '09)*, 2009, pp. 101-111.
 [10] R. Han, M. Ghanem., L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Generation Computer Systems*, 2012, pp. 1-17, doi:10.1016/j.future.2012.05.018.
 [11] R. Pal and P. Hui, "On the Economics of Cloud Markets," , Technical Report, University of Southern Californi, Los Angeles, 2011, pp. 1-7.
 [12] Ghosh, R., Trivedi, K.S., Naik, V.K. and Kim, D.S., "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing*, Tokyo, Japan, 2010, pp. 125-132.
 [13] Ghosh, R., Longo, F., Naik, V.K. and Trivedi, K.S., "Quantifying resiliency of IaaS cloud," in *2010 29th IEEE Symposium on Reliable Distributed Systems*, New Delhi, Punjab India, 2010, pp. 343-347.
 [14] J. Z. Li, "Fast Optimization for Scalable Application Deployments in Large Service Centers," Doctor of Philosophy, Department of Systems and Computer Engineerin, Carleton University, Ottawa, Ontario, 2011.
 [15] R. Han, L. Guo, Y. Guo, and S. He, "A Deployment Platform for Dynamically Scaling Applications in The Cloud," in the *3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'11)*, Athens, Greece, 2011, pp. 506-510.
 [16] L. Guo, Y. Guo, X. Tian, "IC Cloud: A Design Space for Composable Cloud Computing," in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD'10)*, Miami, Florida, 2010, pp. 394-401.
 [17] R. Han, L. Guo, M. Ghanem., and Y. Guo, "Lightweight Resource Scaling for Cloud Applications.," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012)*, Ottawa, Canada, 2012, pp. 644-651.

OpenStack Cloud Security Vulnerabilities from Inside and Outside

Sasko Ristov, Marjan Gusev and Aleksandar Donevski
 Ss. Cyril and Methodius University,
 Faculty of Information Sciences and Computer Engineering,
 Skopje, Macedonia

Email: sashko.ristov@finki.ukim.mk, marjan.gushev@finki.ukim.mk, aleksandar.donevski@outlook.com

Abstract—As usage of cloud computing increases, customers are mainly concerned about choosing cloud infrastructure with sufficient security. Concerns are greater in the multi-tenant environment on a public cloud. This paper addresses the security assessment of OpenStack open source cloud solution and virtual machine instances with different operating systems hosted in the cloud. The methodology and realized experiments target vulnerabilities from both inside and outside the cloud. We tested four different platforms and analyzed the security assessment. The main conclusions of the realized experiments show that multi-tenant environment raises new security challenges, there are more vulnerabilities from inside than outside and that Linux based Ubuntu, CentOS and Fedora are less vulnerable than Windows. We discuss details about these vulnerabilities and show how they can be solved by appropriate patches and other solutions.

Keywords-Cloud Computing; Security Assessment; Virtualization.

I. INTRODUCTION

Infrastructure as a Service (IaaS) is the most offered cloud service layer by public cloud service providers and also the most used by customers. There are lots of open source cloud solutions that offer building IaaS framework over Internet. Selecting a proper IaaS framework is a difficult task since the customers have different requirements and all IaaS frameworks offer various advantages [1]. System administrators mostly care about easy deployment, scalability, supporting different operating systems, hypervisors, and licensing. However, the main concern of cloud computing customers is the security. New challenges arise due to multi-tenancy, virtualization, data and application transfer to third party.

Building a private cloud is a good approach that might solve most of the security challenges, since it mitigates the security risks. However, private clouds lack scalability and elasticity [2]. Therefore, most customers will make their decisions in favor of public clouds, since they offer scalability, elasticity and cost reduction. For example, public clouds reduce the cost up to 85% for disaster recovery compared to on-premise resources [3]. It could provide better Recovery Point Objective (RPO) due to layered backup strategy and also Recovery Time Objective (RTO) due to built-in geographic redundancy. Nevertheless, the cloud service provider maybe has not defined these objectives or they do not meet

with the customer's one [4]. Most discussions and related papers conclude that the main obstacle for public cloud solutions is the security [2]. Confidentiality, integrity and availability are the biggest security concerns faced by the customers in public cloud solutions [5].

Security evaluation of the cloud architecture and cloud service provider should be realized before migrating the customer virtual machines in public cloud. Traditional security incident handling procedures are applicable for cloud computing with some modification to function optimally [6]. Security assessment and comparison of commercial clouds might be a difficult task because of the limited access rights. Therefore, many public cloud service providers use open source clouds.

In this paper, we are interested in analyzing the security vulnerabilities from private or public networks both on virtual machine instances and OpenStack [7] cloud nodes. We focus on OpenStack open source cloud since it is a scalable solution and more than 60 leading companies participate in its development. The goal of this research is to check the validity of the following hypotheses:

- H1 The cloud solution is more vulnerable from inside than outside. Inside vulnerabilities subsume the outside vulnerabilities;
- H2 The multi-tenant environment raises new security vulnerabilities risks from inside the cloud, both for the tenants and the OpenStack cloud provider; and
- H3 Windows based virtual machine instances are more vulnerable than Linux based CentOS [8], Ubuntu [9] and Fedora [10].

The hypotheses are set since the tenants in the cloud are exposed not only from outside, but also from inside the cloud. That is, there is a threat from other tenants, but also from the cloud provider. The cloud provider has also threats from inside, i.e., the tenants. The systems are more vulnerable if the attacker is in the same LAN [11].

We installed the OpenStack cloud with default installation, where the virtual machine instances are installed with default operating systems. Our goal is to determine all possible risks that arise from inside and outside the OpenStack cloud, the OpenStack cloud architecture vulnerabilities, as well as to propose measures to mitigate the security risks by securing detected vulnerabilities. Our analysis is focused

toward operating system vulnerabilities of virtual machine instance hosted in the cloud and the cloud controller where the OpenStack cloud services are deployed. We assess the OpenStack product weaknesses, possibilities of unauthorized access, ensuring data confidentiality, integrity and availability, risk of DoS (Denial-of-Service) or even Distributed DoS (DDoS) attacks, man-in-the-middle attack, etc.

The rest of the paper is organized as follows. Related work is presented in Section II. Section III briefly describes the OpenStack cloud architecture and its components. The methodology for security assessment is presented in Section IV. In Section V, we present assessment results both for inside and outside vulnerabilities. We discuss and conclude our work, and present future work in Section VI.

II. RELATED WORK

There are a lot of open source cloud solutions to build a private cloud with IaaS cloud service layer. Voras et al. [12] devise a set of criteria to evaluate and compare most common open source IaaS cloud solutions. Mahjoub et al. [13] compare the open source technologies to help customers to choose the best cloud offer of open source technologies. Most common open source cloud computing platforms are scalable, provide IaaS, support dynamic platform, Xen virtualization technology, linux operating system and Java [14]. However, they have different purposes. For example, Eucalyptus [15] fits well to build a public cloud services (IaaS) with homogeneous pool of hypervisors, while OpenNebula [16] fits well for building private/hybrid cloud with heterogeneous virtualization platforms [17].

Many authors have analyzed the cloud security challenges and propose methodologies for security evaluation of the cloud solutions. Cloud Security Alliance (CSA) announce Cloud Control Matrix Version 1.3 [18] which can assist the potential cloud customers to assess the overall security risk of a cloud service providers classifying the security controls according to cloud service layer and architecture. A methodology for security evaluation of on-premise systems and cloud computing based on ISO 27001:2005 [19] is proposed in [20]. The authors in [4] evaluate ISO 27001:2005 control objective importance for on-premise and the three cloud service layers IaaS, PaaS (Platform as a Service) and SaaS (Software as a Service). International Organization for Standardization (ISO) is developing new guidelines ISO/IEC WD TS 27017 [21] that will recommend relevant security controls for information security management system (ISMS) implementation in cloud computing. Eucalyptus and CloudStack [22] have integrated the maximum security level in front of OpenNebula and OpenStack open source cloud solutions [23].

III. THE OPENSTACK CLOUD ARCHITECTURE

Open source clouds have similar architecture [24]. Each open source cloud has, at minimum:

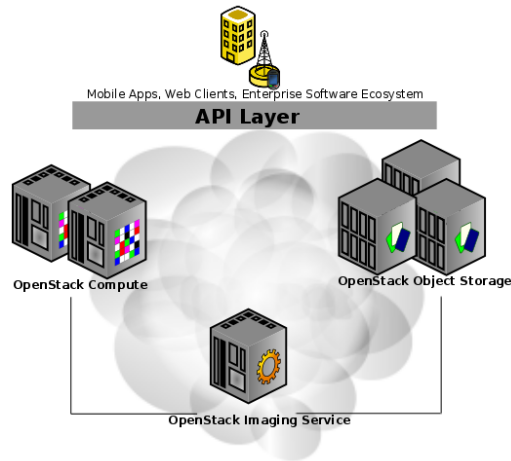


Figure 1. The three components of OpenStack cloud [7]

- *cloud controller* - several services are deployed on this server that control the system, network, schedule the virtual machine instances and act as administrator interface; and
- *cloud node* - this server hosts the virtual machine instances of virtual machines. It communicates with the cloud controller.

This section briefly describes the architecture of the newest Folsom release of OpenStack cloud, its components, networking and features.

A. OpenStack Components

Figure 1 depicts the three main components of OpenStack cloud: Compute, Object Storage, and Image Service.

Compute Infrastructure (Nova) is the core part of the OpenStack cloud that manages instances of virtual machines and networking. *Object Storage* is the subsystem that stores the objects in a massively scalable, large capacity system. It backs up and archives data, stores secondary or tertiary static data, stores data when predicting storage capacity is difficult, and creates the elasticity and flexibility of cloud-based storage for customer web applications. *Image Service* is lookup and retrieval subsystem for virtual machine images.

B. OpenStack Deployment

OpenStack can be deployed and runs on Linux Ubuntu, CentOS and RedHat operating systems. It supports KVM [25], Xen [26], UML [27], and Hyper-V [28] hypervisors. Nova services can be deployed either on the same physical server or they can be installed on separate servers. The OpenStack cloud can be deployed in three different modes:

- **Single Node:** All nova-services are deployed on only one physical server which hosts also all the virtual machine instances.

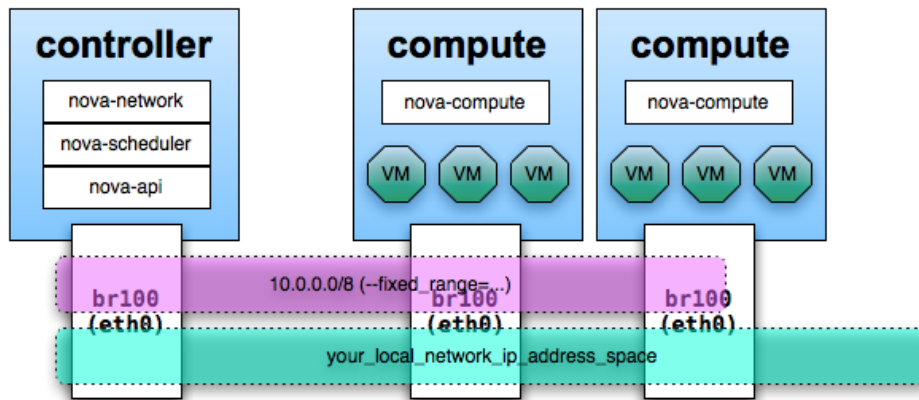


Figure 2. OpenStack networking example [7]

- Dual Node: This deployment consists of two physical servers, i.e., the *Cloud Controller Node (CCN)* and the *Compute Node (CN)*. The former is used as cloud controller which runs all the nova-services except for nova-compute. The latter is deployed with nova-compute to instantiate virtual machine instances.
- Multiple Node: Particular number of CNs can be installed resulting in a multiple node installation. Volume controller and a network controller can be added as separate nodes in a more complex, multiple node installation.

We deployed the OpenStack in the Single Node since we are not interested in performance, but for security. The choice for Single Node is based on the fact that the security vulnerabilities of OpenStack services do not depend on the number of physical nodes.

C. OpenStack Networking

OpenStack network consists of two networks, public and private, as depicted in Figure 2. IP addresses from the public network are associated with virtual machines instances to be accessed from the public Internet. The private network is used for internal cloud web service communication.

In this paper, we are interested in analyzing the security vulnerabilities from private or public networks both on virtual machine instances and OpenStack cloud node deployed in Single Node.

IV. METHODOLOGY FOR SECURITY ASSESSMENT

This section presents the methodology for security assessment on OpenStack cloud and virtual machine instances. It is based on two assessments with two groups of test cases for different targets. The goal of the assessments is to determine the vulnerabilities of the OpenStack cloud nodes (Compute and Controller deployed in one physical server) and virtual machine instances with different operating systems, both from inside and outside the OpenStack cloud.

A. The Targets

Two different target groups will be assessed. The first target group covers the assessment of physical OpenStack server node which is installed with Ubuntu Server 12.04 64-bit operating system. The second target group covers the assessment of virtual machine instances hosted in the cloud with operating systems:

- Windows 2008 R2 Standard 64 bit;
- CentOS 6 64 bit;
- Ubuntu 10.04 Server Edition 64 bit; and
- Fedora 17 64 bit.

The virtual machine instances are installed with default configuration in order to detect all possible vulnerabilities. We will address which vulnerabilities can be secured after implementing additional patches or reconfigurations.

B. Security Assessment Plan

The security assessment basic goal is to determine the security vulnerabilities of the targets from inside and outside the OpenStack cloud. Therefore, we realize two different assessments using Nessus 5 vulnerability and configuration assessment scanner [29] using External Network Scan policy. Nessus scans all TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) ports, as well as the vulnerabilities of the services that work on certain opened port.

Each vulnerability is rated as derived from the associated Common Vulnerability Scoring System (CVSS) [30] score:

- *Info* if CVSS score is 0;
- *Low* for CVSS score $\in \{1, 2, 3\}$;
- *Medium* for CVSS score $\in \{4, 5, 6\}$;
- *High* for CVSS score $\in \{7, 8, 9\}$; and
- *Critical* if CVSS score is 10.

Figure 3 depicts the test cases of security assessment from inside and outside the OpenStack cloud, i.e., on private and

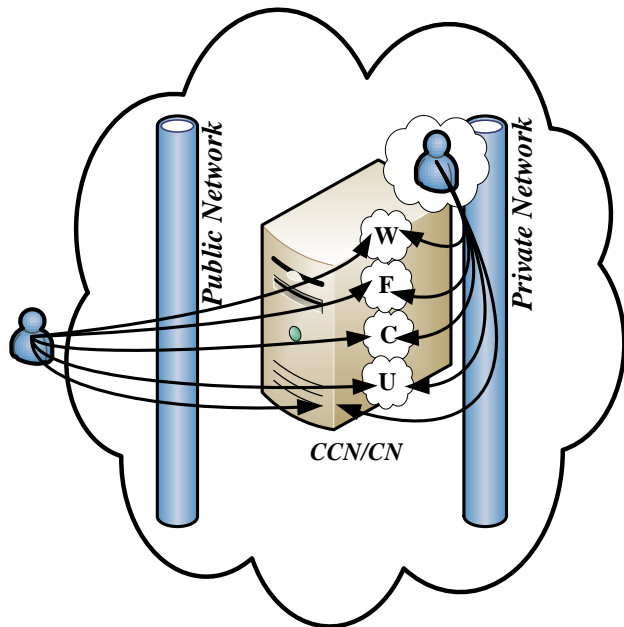


Figure 3. Inside and outside security assessment

public network. U denotes the Ubuntu operating system, while F, C and W denote the Fedora, CentOS and Windows operating systems, correspondingly.

1) *Inside Security Assessment:* The Nessus client is deployed on one virtual machine instance. It scans the four virtual machine instances with different operating systems and cloud physical server node (both CCN and CN are the same physical server in our case). This assessment from inside simulates the tenant and its goal is to assess the vulnerabilities that arise from the cloud multi-tenancy. OpenStack private network is used to communicate among the target inside virtual machine instances, the cloud physical node and the virtual machine instance with Nessus client.

2) *Outside Security Assessment:* The Nessus client is deployed on a workstation outside the OpenStack cloud, i.e., on a public network. It also scans the same four virtual machine instances with different operating systems hosted in the OpenStack cloud and the cloud physical server node. This assessment goal is to assess the vulnerabilities that arise for virtual machine instances and the OpenStack cloud services outside the cloud. OpenStack public network and floating IP addresses are used for communication with virtual machine instances and cloud physical server node.

V. THE RESULTS OF THE ASSESSMENT

This section presents the results of both assessments for both target groups defined in previous Section IV. We omit the results of the assessments with CVSS score 0 since they are informative, rather than real vulnerabilities. The values

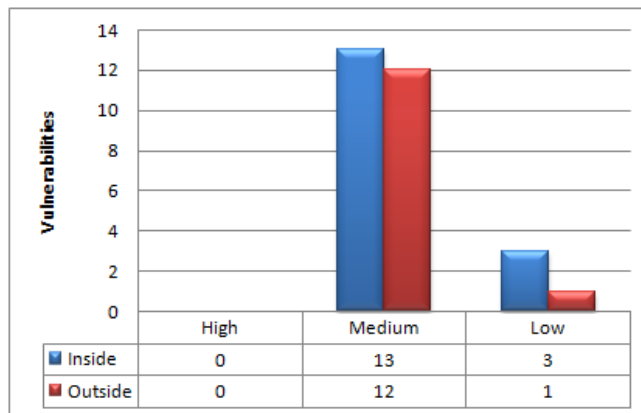


Figure 4. Summary results of OpenStack security assessment

for critical vulnerabilities are also omitted since we have not found any critical vulnerability during the assessments.

A. OpenStack Node Vulnerabilities

Figure 4 depicts the summary results of the security assessment of the cloud node.

The results confirm the hypothesis H1 that there are more inside vulnerabilities which subsume the outside vulnerabilities. 13 medium and 3 low vulnerabilities are detected from inside and only 1 low and 12 medium vulnerabilities are detected from outside. High vulnerabilities are not detected neither from outside, nor from inside.

Let us assess detected vulnerabilities in more detail. 6 Web Server Generic XSS (Cross-site scripting) and 6 Web Server Generic Cookie Injection vulnerabilities (medium) are detected by both assessments on several ports. We conclude that the web server is prone to cross-site scripting and cookie injection attacks. Therefore, new patches must be developed in order to secure two assessed vulnerabilities. Common low vulnerability is the usage of plain text authentication forms which should be transmitted encrypted over secured HTTPS.

Assessment of inside vulnerabilities detected 1 additional medium vulnerability, i.e., the DNS (Domain Name System) server is vulnerable to cache snooping attacks. DNS software vendor should fix it.

Two additional low vulnerabilities are detected, as well. DHCP (Dynamic Host Configuration Protocol) server may expose information about the associated network and applying filtering will keep the information off the network and mitigate the risk of this vulnerability. The web server leaks a private IP address that is usually hidden behind a NAT (Network Address Translation) Firewall or proxy server. However, this is not a real vulnerability since our private IP address will be a public IP in real world scenario.

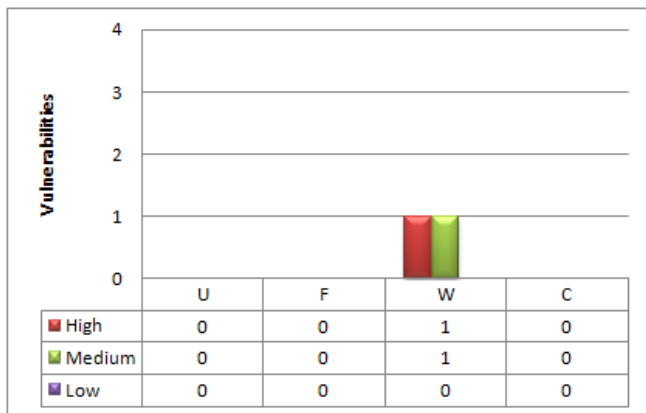


Figure 5. Summary results of outside security assessment on instances

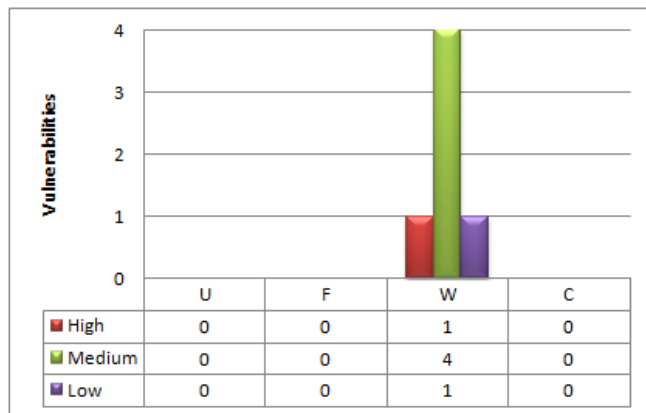


Figure 6. Summary results of inside security assessment on instances

B. Virtual Machine Instance Vulnerabilities

In this section we present and analyze the results of the assessment of the four instances, each with different operating system, both from inside and outside the OpenStack cloud.

1) *Vulnerabilities from Outside:* Figure 5 depicts the summary results of the outside security assessment on virtual machine instances. U denotes the Ubuntu operating system, while F, C and W denote the Fedora, CentOS and Windows operating systems, correspondingly.

The Nessus client has not detected any vulnerability neither on Ubuntu, nor on Fedora, nor on CentOS operating system. 1 high and 1 medium vulnerabilities are detected on Windows operating system with the assessment from outside the OpenStack. Windows could allow arbitrary code execution (high vulnerability) in the implementation of the Remote Desktop Protocol (RDP). The problem with Windows lies in the requirement to activate remote desktop to connect to Windows, instead of secured SSH (Secure Shell) protocol to connect on Linux based operating systems. However, installing the existing patch will secure the vulnerability. Network Level Authentication (NLA) on the remote RDP server is not configured (Low vulnerability) by default and should be enabled.

2) *Vulnerabilities from Inside:* Figure 6 depicts the summary results of the inside security assessment on virtual machine instances hosted in OpenStack.

Linux based operating systems are not detected with any security vulnerability from outside the OpenStack cloud, as well. The same 1 high and 1 medium vulnerabilities are detected from inside the virtual machine instance with Windows operating system. However, 3 additional medium vulnerabilities are detected. The first, Windows is using weak cryptography by default for RDP and changing RDP encryption level to "High" or "FIPS Compliant" will mitigate this vulnerability. The second, the virtual machine instance is vulnerable to a man-in-the-middle attack. Forcing

SSL (Secure Sockets Layer) or RDP with NLA will secure the vulnerability. The last detected medium vulnerability is "man-in-the-middle attack against the Server Message Block (SMB) server" which can be secured by enforcing message signing.

FIPS-140 incompliance for terminal services encryption level is the additional low vulnerability which can be secured changing RDP encryption level to "FIPS Compliant".

VI. CONCLUSION AND FUTURE WORK

We have realized security assessments of OpenStack cloud services and four virtual machine instances with different operating systems Fedora, Ubuntu, CentOS and Windows. The experiments addressed the security vulnerabilities both from inside and outside the OpenStack cloud.

The results of the assessments proved hypothesis H2 that cloud multi-tenant environment raises new security vulnerabilities risks from inside the cloud, both for the tenants and the OpenStack cloud provider. Inside vulnerabilities subsume the outside vulnerabilities for the cloud node and each operating system, which proves the hypothesis H1.

Vulnerabilities on Linux operating systems are not detected, neither from outside, nor inside. The assessment of Windows operating system shows additional 1 low and 3 medium security vulnerabilities, which proves the hypothesis H3. All these vulnerabilities are not detected from outside since the OpenStack cloud denies all TCP and UDP ports from outside by default. They still exist because of the Windows default installation (configuration) and the requirements of creating Windows image.

Although Windows based virtual machine instances with default configuration are less secure than Linux based instances, all Windows vulnerabilities can be secured by implementing existing patches or reconfiguration. Only then RDP port 3389 should be opened to outside.

OpenStack cloud is also more vulnerable from inside the cloud with additional 1 medium and 2 low vulnerabilities

which can be secured with reconfiguration. However, we detect that OpenStack cloud has 2 medium vulnerabilities on 6 different ports that can not be secured with reconfiguration, but with new patches that should be developed. All detected OpenStack security vulnerabilities do not depend on creating different virtual machine images, but they exist with default OpenStack deployment.

This paper realizes the security assessment of OpenStack open source cloud and virtual machine instances hosted with different operating systems. We will continue the security assessment on the other open source clouds and bring relevant conclusions about their security vulnerabilities. This will help the customers to select the most appropriate cloud solution regarding the security.

REFERENCES

- [1] G. von Laszewski, J. Diaz, F. Wang, and G. Fox, "Comparison of multiple cloud frameworks," in *Cloud Computing (CLOUD)*, 2012 IEEE 5th International Conference on, June 2012, pp. 734–741.
- [2] M. Shtern, B. Simmons, M. Smit, and M. Litoiu, "An architecture for overlaying private clouds on public providers," in *8th Int. Conf. on Network and Service Management, CNSM 2012*, Las Vegas, USA, 2012.
- [3] T. Wood, E. Cecchet, K. K. Ramakrishnan, P. Shenoy, J. van der Merwe, and A. Venkataramani, "Disaster recovery as a cloud service: economic benefits & deployment challenges," in *Proc. of the 2nd USENIX conf. on Hot topics in cloud comp., ser. HotCloud'10*, USA, 2010, pp. 8–8.
- [4] S. Ristov, M. Gusev, and M. Kostoska, "Cloud computing security in business information systems," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 4, no. 2, March 2012, pp. 75–93.
- [5] S. Chaves, C. Westphall, C. Westphall, and G. Geronimo, "Customer security concerns in cloud computing," in *Proceedings of the 10-th Int. Conf. on Networks, ser. ICN 2011. IARIA*, 2011, pp. 7–11.
- [6] A. TaheriMonfared and M. G. Jaatun, "As strong as the weakest link: Handling compromised components in OpenStack," in *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, ser. CLOUDCOM '11*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 189–196.
- [7] O. C. Software. Openstack cloud. [Retrieved: March, 2013]. [Online]. Available: <http://openstack.org>
- [8] T. C. Enterprise, "Centos," [Retrieved: March, 2013]. [Online]. Available: <http://www.centos.org/>
- [9] Canonical, "Ubuntu," [Retrieved: March, 2013]. [Online]. Available: <http://www.canonical.com/>
- [10] A. R. H.-S. C. Project, "Fedora," [Retrieved: March, 2013]. [Online]. Available: <http://fedoraproject.org/>
- [11] H.-C. Li, P.-H. Liang, J.-M. Yang, and S.-J. Chen, "Analysis on cloud-based security vulnerability assessment," in *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*, Nov. 2010, pp. 490–494.
- [12] I. Voras, B. Mihaljevic, and M. Orlic, "Criteria for evaluation of open source cloud computing solutions," in *Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on*, June 2011, pp. 137–142.
- [13] M. Mahjoub, A. Mdhaffar, R. B. Halima, and M. Jmaiel, "A comparative study of the current cloud computing technologies and offers," in *Proceedings of the 2011 First International Symposium on Network Cloud Computing and Applications, ser. NCCA '11*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 131–134.
- [14] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *Proceedings of the 2009 Second International Symposium on Information Science and Engineering, ser. ISISE '09*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 23–27.
- [15] Eucalyptus. Eucalyptus cloud. [Retrieved: March, 2013]. [Online]. Available: <http://www.eucalyptus.com/>
- [16] OpenNebula. Opennebula cloud software. [Retrieved: March, 2013]. [Online]. Available: <http://Opennebula.org>
- [17] T. D. Cordeiro, D. B. Damalio, N. C. V. N. Pereira, P. T. Endo, A. V. de Almeida Palhares, G. E. Goncalves, D. F. H. Sadok, J. Kelner, B. Melander, V. Souza, and J.-E. Mangs, "Open source cloud computing platforms," in *Proceedings of the 2010 Ninth International Conference on Grid and Cloud Computing, ser. GCC '10*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 366–371.
- [18] CSA, "Cloud security alliance," [Retrieved: March, 2013]. [Online]. Available: <http://cloudsecurityalliance.org/>
- [19] ISO/IEC, "ISO/IEC 27001:2005, Information Security Management Systems - Requirements," [Retrieved: March, 2013]. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=42103
- [20] S. Ristov, M. Gusev, and M. Kostoska, "A new methodology for security evaluation in cloud computing," in *MIPRO, 2012 Proc. of the 35th Int. Convention, IEEE Conference Publications*, 2012, pp. 1808–1813.
- [21] ISO/IEC, "WD TS 27017, Guidelines on information security controls for the use of cloud computing services," [Retrieved: March, 2013]. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43757
- [22] CloudStack. Cloudstack opens source cloud computing. [Retrieved: March, 2013]. [Online]. Available: <http://cloudstack.org>
- [23] S. Ristov, M. Gusev, and M. Kostoska, "Security assessment of openstack open source cloud solution," in *Proceedings of the 7th South East European Doctoral Student Conference (DSC2012)*, 2012, pp. 577–587.

- [24] C.-H. Ng, M. Ma, T.-Y. Wong, P. Lee, and J. Lui, "Live deduplication storage of virtual machine images in an open-source cloud," in Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware, ser. Middleware'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 81–100.
- [25] KVM, "Kernel based virtual machine," [Retrieved: March, 2013]. [Online]. Available: http://www.linux-kvm.org/page/Main_Page
- [26] C. Systems, "Xen hypervisor," [Retrieved: March, 2013]. [Online]. Available: <http://www.xen.org/>
- [27] UML, "User-mode linux kernel," [Retrieved: March, 2013]. [Online]. Available: <http://user-mode-linux.sourceforge.net/>
- [28] Microsoft, "Microsoft hyper-v server 2012," [Retrieved: March, 2013]. [Online]. Available: www.microsoft.com/hyper-v-server/
- [29] Tenable, "Nessus 5," [Retrieved: March, 2013]. [Online]. Available: <http://www.tenable.com/products/nessus>
- [30] N. V. Database, "Common vulnerability scoring system," [Retrieved: March, 2013]. [Online]. Available: <http://nvd.nist.gov/cvss.cfm>

Seamlessly Enabling the Use of Cloud Resources in Workflows

Michael Gerhards, Volker Sander

Faculty of Medical Engineering & Technomathematics
FH Aachen, University of Applied Sciences
Jülich, Germany
{M.Gerhards|V.Sander}@fh-aachen.de

Adam Belloum

Institute of Informatics
University of Amsterdam
Amsterdam, The Netherlands
A.S.Z.Belloum@uva.nl

Abstract—The hosting of large on-premise computational resources is common practice. Cloud Computing offers a promising, alternative infrastructure for using scalable on-demand off-premise resources. However, outsourcing whole applications is not a cost optimal solution in some scenarios, because the already existing on-premise resources are not considered. A flexible integration of additional resources from the cloud to compensate a shortage of suitable on-premise resources is a tradeoff between costs and efficiency. This paper provides a light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack. The approach is evaluated by running an example workflow.

Keywords—cloud economics; dynamic resource allocation; cloud computing; cross-cloud workflows; on-demand computing model; service oriented architecture; workflow; workflow orchestration.

I. INTRODUCTION

Refactoring on-premise computational resources to form a computer center is common practice. However, it is not reasonable to provide a solution for all requested resource types in such a center. First of all, the initial purchase costs are very high. For small and medium enterprises (SME) it is nearly impossible to bear these costs alone. Even after a purchase the disadvantages still occur, mainly due to the operational costs. The hosting company is bound to the resources for many years, even if the computational power is no longer required. The old hardware does not benefit from new technologies, which were developed in the meantime. If specific resources are used with unbalanced load, there is the risk of underuse. An overprovisioning is also required for load peaks which also increase the costs.

Cloud computing offers a promising alternative infrastructure for using scalable on-demand resources. Providers such as Amazon allow users to allocate virtualized computational resources. Of course, those providers allow for porting the full application. However, this might not be the most cost-effective solution, because the already existing on-premise resources are not considered. Therefore, for many scenarios it appears to be opportune to integrate cloud resources with easy-scale and dynamic provisioning into the local environment for the execution of computation intensive application parts whereas the other application parts are

executed on local available general-purpose computational resources. An example is a highly parallelized application which could use a Graphics Processing Unit (GPU) in the cloud, while the remainder of the program is executed locally.

This paper will briefly present existing complex software stacks which combine on-premise resources with cloud resources. Then it introduces our light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack. The paper will focus on workflows because the division of applications into parts is natively supported. The basic ideas apply to a much broader application domain.

The paper is organized as follows: The second section presents the cloud-enabled workflow environment. It introduces the challenges for such an environment and provides solutions. The third section evaluates the presented solutions by describing a run of an example workflow in a specific workflow management system under the use of cloud resources. The last section concludes the lessons learned and provides future work. For simplicity reasons we omitted to refer to related work in an isolated section. Instead we provide references when the according context is discussed.

II. CLOUD-ENABLED WORKFLOW ENVIRONMENT

Many publications deal with cloud computing since it is the greatest IT hype of the last ten years. Surprisingly the combination of cloud computing with workflows is little addressed. "With the emerging of the latest cloud computing paradigm, the trend for distributed workflow systems is shifting to cloud computing based workflow systems [1]." In comparison to the mobile smart domain, approaches like CloneCloud already exists to dynamically partition applications between weak devices and clouds [2]. Nephele is another approach that claims to be "the first data processing framework to explicitly exploit the dynamical resource allocation offered by today's compute clouds for both, task scheduling and execution [3]." Nephele itself is focused on performance in full cloud environments but does not consider available on-premise resources which results in a lower performance but a cost reduction. A tradeoff between costs and performance is missing.

Workflows in cloud computing are addressed by several EU projects. “BREIN takes the e-business concept developed in recent grid research projects, namely the concept of so-called “dynamic virtual organizations” towards a more business-centric model, by enhancing the system with methods from artificial intelligence, intelligent systems, semantic web etc. [4].” BREIN can enhance some cloud features like automatic resource allocation and outsourcing of resources to third party. The approach presented in this paper also focuses on resource allocation and outsourcing but from a more technical sight by combining existing lightweight technologies. It does not consider collaboration between companies. The required components of the overall architecture are similar: A workflow framework with service broker and registry.

A. Service layers and deployment models

The National Institute of Standards and Technology (NIST) distinguishes the three service layers: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) and four different deployment models: Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud [5]. The cloud-enabled workflow environment differs dependent of the used service layer and deployment model. A detailed comparison of the different service models and deployment models is given in [6]. The rest of the paper will therefore focus on Public Cloud IaaS resources to assume the minimum of requirements. This should not limit the generic aspects of the proposed solution since other service layers and deployment models can be used instead with less effort.

B. Security and Governance

This paper assumes that the workflow management system runs on-premise or in a private cloud and is used only by users of a single organization. This assumption simplifies the security handling since the organization is interfacing with the cloud service providers as a whole. Cross-organizational environments can be addressed by applying the concept of virtual organizations [7].

While incurred costs would be billed against the organization, the actual costs still have to be mapped to cost units within the organization. Therefore, an AAAA (Authentication, Authorization, Admission control, and Accounting) is required. Actually, an AAAAA mechanism is demanded, i.e. an additional auditing mechanism like described in Section II.L.

During application runtime off-premise cloud resources will access on-premise data for calculations. To protect the data against unauthorized access credentials are required. These credentials are entered by the user at the start of the application. If a native support is not guaranteed, the credentials can be entered during a WS-HumanTask, which stores credentials in a secured short-lived repository with limited life time [8]. This procedure is used by our approach. The integration of tasks is detailed in Section II.F.

To assure authentication and authorization, we extend the idea of using WS-HumanTask for credentials and propose an architecture we presented in the context of our publication of

a security framework for our WS-HumanTask implementation. This publication “presents a generic framework that supports a pull-based work distribution strategy in distributed environments with the help of a task repository that mediates tasks between resources and workflow instances [9].” It provides an implementation for Role Based Access Control (RBAC) based authorization. To provide a certificate repository, we follow the concept of MyProxy which is an authentication technology from the grid domain which lets the workflow impersonating the user [10].

C. Conditions on applications

A condition for executing different parts of the same application on different premises is an application which is divisible into parts. Modeling a complex application as workflow supports its division into simpler individual parts that are executed as interacting tasks by a workflow management system that takes care of the individual tasks’ progress and dependencies [11].

The Generic Workflow Execution Service (GWES) is an open source workflow management system which was developed by Fraunhofer-Gesellschaft for the management and the automation of complex workflows in heterogeneous environments [12]. GWES was originally developed basing on grid technologies like Globus Toolkit as Grid Workflow Execution Service (also GWES) and was then adjusted to the cloud domain. To conclude GWES is a specific workflow management system with an own workflow description language.

In contrast, the interoperable approach presented in this paper bases on an extension for existing arbitrary workflow management systems by its loosely coupled connection to a cloud broker to enable the use of additional cloud resources. By choosing a workflow management system independent approach the benefit of using the already known system is given for the end-user.

AMOS is “a system that combines grid and cloud technologies in a novel way to support on-demand execution of e-Science applications [13].” The e-Science applications handled in this paper are also modeled as workflow and executed in the cloud. The main idea is the creation of a “transient grids by automatically installing and configuring grid middleware on the purchased resources“. In contrast the approach of this paper provides a light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack.

“OPTIMIS deliverables will enable clouds to be composed from multiple services and resources. It will support service brokerage via interoperability, and is architecture-independent [14].” It provides “a toolkit for supporting service provisioning using Cloud eco-systems consisting of multiple Cloud infrastructures from different providers with guaranteed Quality of Service (QoS)”. A direct integration of workflows is not part of the project but as a future work the usage of OPTIMIS as underlying cloud infrastructure in combination with the workflow tools of this paper could be tested.

D. Resource independent modeling of workflows

The various tasks from a workflow are of different task types. Most task types like control flow or script tasks are executed on the workflow management system's computer. But service tasks are computation demanding and therefore executed as service - or other remote procedure call (RPC) - on suitable hardware resources. The workflow will run in a Service Oriented Architecture (SOA) in a combination of services, which are deployed on-premise and in the cloud. The invocation of the cloud services must be protected against unauthorized usage using a system like described in Section II.B. The data flow for large data sets is not integrated into the workflow but in the service software directly. Since pushing the data in a web service invocation message results in bad performance through marshaling the data, only the data location and an authorization ticket is sent to the service. The service then loads the data using a third party high performance file transfer mechanism like GridFTP. The security aspect is handled in Section II.B. This paper presents how these script tasks can be executed on enabled cloud resources without workflow modification. If additional cloud resources are enabled is decided during runtime.

The concept of considering only physical resources is gone in the cloud vision of elastic resources, which can be instantiated on-demand. Therefore, workflows are modeled independently of specific resources by abstracting service endpoints as service names. This enables the easy exchange of an on-premise endpoint with an off-premise endpoint, e.g., in the cloud. The binding of workflow tasks to endpoints is done at runtime by dissolving the service names. The service registry contains assignments between all service names to available service endpoints independent if the endpoint is located on-premise or off-premise in the cloud. In Figure 1 both tasks "T1" and "T2" fetches their endpoints from the service registry. A so modeled workflow can be executed in the usual way without disadvantages.

Enterprise service buses (ESB) like Mule or Fiorano are also able to manage dynamic endpoints independently of the endpoint location [15]. However, compared to our solution, ESBs are rather heavyweight software products which increase the complexity of the architecture. Connectors between workflows and ESB are application dependent.

This paper provides a light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack. Such an approach lowers the entry barrier. This empowers workflow users to benefit from the cloud in an easy way.

E. Enabling cloud resources using a broker

In cloud economics, resources are frequently provided following a pay-per-time billing structure. The time is billed when they are available even when the resources are not used. Therefore these resources are shut down when idling. If a shutdown resource is required at the service registry the resource must first be instantiated. According to the National Institute of Standardization (NIST) Cloud Computing Reference Architecture [5], the dynamic allocation of cloud

resources is done by a cloud broker. The cloud broker is "an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between cloud providers and cloud consumers [5]." The cloud broker publishes endpoints of instantiated cloud resources at the service registry.

F. Connection between workflow and cloud broker

The connection between workflow management system and cloud broker can be established at different locations in the overall workflow environment. Possible locations are tasks, called functions of tasks, the workflow, and the workflow management system itself is the source code is available. The advantages and disadvantages of the different connection locations are discussed in [1].

To not change the workflow management systems source code, the cloud broker connection is integrated into the workflow template itself. The workflow template can be seen as the source code of the workflow but not of the invoked services. A preprocessor creates a new extended workflow template out of the original workflow template. It consists of all original tasks in the given order but with interposed administrative tasks to handle the cloud broker connection for service tasks which should be executed in the cloud. The preprocessing process is also used to customize the workflow execution like described in Section II.G and to feed the provenance service of Section II.L.

The additional administrative tasks are similar to ESB adapters or cloud connectors. This new extended workflow is executed instead [16]. In Figure 1 the administrative task "AT" connects to the cloud broker to enable the cloud resource before its service is invoked by the service task "T2".

G. Identification of cloud tasks

Before the start of the workflow, the scheduler has to check if enough suitable on-premise resources are available. To realize this task, a resource description language like the Job Submission Description Language (JSDL) can be used to describe the different requirements for each individual task [17]. If not enough suitable local resources are available some tasks have to be redirected to cloud resources. Here, the scheduler must have all information about all constraints that apply to tasks that might be handled by cloud resources.

The user has the ownership of the data and decides which individual tasks are allowed for execution on integrated cloud resources. One possibility to model that is the usage of JSDL task annotations in the workflow template. This is similar to MAUI where developers annotate which methods of an application can be offloaded for remote execution [18]. If annotations are not supported in the workflow modeling language, another possibility is outsourcing the annotations to a workflow or task dependent configuration in a separate file with references to the original workflow template. Figure 1 illustrates the input of the scheduler and the annotation files together with the original workflow template to the preprocessor, which forms the extended workflow template. The administrative tasks of Section II.F are customized evaluating the annotations described above.

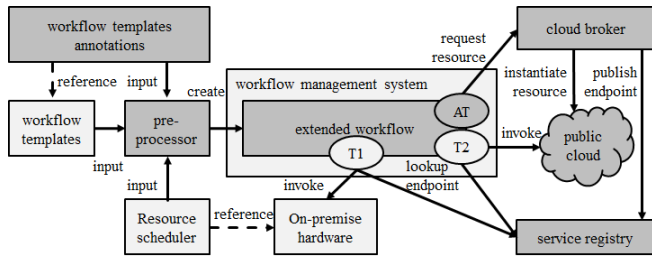


Figure 1. Cloud-enabled workflow environment. Components with bright background are the legacy system and components with dark background are extensions.

All tasks that will stay on-premise for execution are called local tasks whereas the tasks executed off-premise in the cloud are called cloud tasks. The cloud tasks get administrative predecessors and successors to connect to the cloud broker to enable the cloud resources. So all tasks are now arranged in one of these two categories. Since cloud tasks cause administrative overhead, they should only be used for computation intensive tasks like service tasks.

H. Endpoint selection strategy

At this point the workflow itself is prepared for an execution across organizational boundaries. The binding of service tasks to service endpoints is done at runtime by dissolving the service names at the service registry. Since the number of idling active cloud resources is kept to a minimum to avoid costs it is not guaranteed that the service registry holds an entry for the required service. The decision making plan to select an endpoint is illustrated in Figure 2 and explained in the following paragraphs.

The simplest case is illustrated in the first two branches: The service is already available and registered at the service registry. This is common if it is deployed on on-premise resources or in the cloud, e.g., from a previous run or as SaaS solution.

If the required web service is not available at the service registry, the service broker checks if a suitable underutilized or idling resource is running which represents the 3rd branch of Figure 2. The cloud broker re-installs the required software from a repository on that resource and publishes the new endpoint at the service registry. The installation process is described in Section II.I. This procedure is most suitable for workflows with different cloud tasks that can then be executed in a pipeline on the same cloud instance. It also reduces the data movement.

If neither suitable service nor resource is available a new resource representing the last branch of Figure 2 must be instantiated.. This process is presented in Section II.J. The instantiation takes time during provisioning and software installation which pause the task execution. It also causes new costs for renting an additional cloud resource.

Independent of the endpoint provisioning variant, the endpoint is now available and registered at the service registry. Like illustrated in Figure 1 the service tasks fetches their endpoints from the service registry and invokes the service directly. This proceeding is implemented in the workflow management system in its natural way.

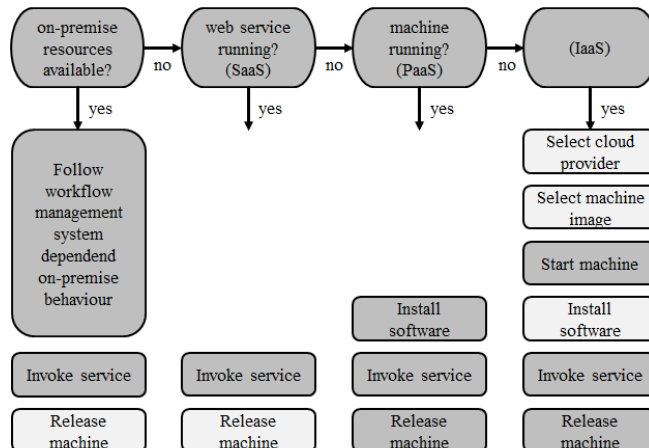


Figure 2. Endpoint selection decision process. Steps with bright background are optional and depend on the implementation.

I. Deployment of software on a running machine

The deployment of the web service including its required container can be done simply by using scripting (SCP / SSH | PowerShell). Password prompts can be suppressed using public/private key based authentication. The required keys are stored by the user in a secure key repository as provided for file transfer. The workflow is empowered to read these key using the mechanism described in Section II.B. A more sophisticated solution in comparison to scripting is to use cloud agnostic interfaces such as the Open Cloud Computing Interface (OCCI) or the compute API tool of jclouds [19][20]. The OCCI Working Group has highlighted the need for machine-readable Service Level Agreements (SLAs) associated with the dynamic provisioning of cloud computing resources.

J. Instantiation process of a new cloud resource

Preconfigured machine images contain only the required software for immediate use to speed up the instantiation. Each abstract cloud task uses its own machine image which is identified evaluating the abstract task’s description in the workflow template. The cloud instance loads its machine image from its storage system. After startup, the web service endpoint is published to the service registry. An alternative is the use of a generic machine image which only contains the rudimentary software and is customized at runtime by additional software installation like described in Section II.I.

The billing period of a public cloud provider would start now together with the instantiation of the cloud resource instance.

K. Cloud Provider selection strategy

The flexible enabling of resources of the most suitable cloud provider for each individual task is an optimization to form a cross-cloud workflow with intra- and inter-cloud communications. The selection process can be modeled similar to the three-phase cross-cloud federation model described in [21]. In the discovery phase, the cloud broker collects information about assured properties offered by the

cloud providers. Each abstract cloud task specifies its requirements. “Each object is characterized by a set of properties/attributes; each property is a tuple (name, value), with name a string of characters and value [22].” In the match-making phase, the cloud broker compares the cloud task’s requirements with the cloud providers’ assured properties. The cloud providers that assure all requirements of the requesting task are potential task owners. In the authentication phase, the cloud broker selects the cheapest potential owner as the current owner for each cloud task. Matchmaking between requirements and properties was already handled in the grid domain. A “formal definition of matchmaking, overview algorithms to evaluate different matchmaking expressions, and develop a matchmaking service for an intelligent grid environment” is presented in [22].

One challenge arises if the workflow execution depends on large data because the data movement costs and time have to be considered. In [23], “a Network and Data Location Aware job scheduling has been proposed for data intensive jobs. The proposed scheduling algorithm takes into account network characteristics, disk read speed of data sources, and data locations of input files, as well as other computational factors (CPU power, memory, CPU load, etc.) when making scheduling decisions.”

L. Provenance

The importance of auditing the outcome of computation processes is a fundamental quality characteristic to many application domains. The automated tracking and storing of provenance information during workflow execution could satisfy this requirement [24]. The required data can be pushed out of the workflow by the administrative tasks introduced in Section II.F. Provenance traces enable the users to see what has happened during the execution of the workflow. This enables failure analysis and future optimization. Provenance becomes even more important in distributed environments because workflow tasks are loosely bound to computational resources. Using provenance in the cloud-workflow domain enables the identification of task to cloud assignments so that it is visible where the cloud task has been executed and where its data have been stored.

Provenance also shows at which time the cloud instance was running and therefore causing costs. Based on provenance traces, statistics can be created showing which workflows cause which costs, which users cause which costs, which clouds cause which costs, which users instantiate which workflows, which clouds execute which cloud task, etc.. A detailed comparison of two possible provenance models is done in [25].

III. EVALUATION

The prototype of [26] following the ideas of Section II is evaluated in this section. First an example workflow was modeled. Then required software products were chosen and deployed together with the self-developed cloud broker to form the cloud-enabled environment illustrated in Figure 1. Finally the example workflow was executed in the established testing environment. This evaluation shows how

the lightweight system works basing on an example workflow. Not all components of the prototype were ready when this paper was written. Therefore, some are simulated using a mock like indicated at the corresponding place.

One advantage of combining on- and off-premise resources is a cost reduction attributable to the performance. Since cost structures vary they are not considered in this evaluation.

A. Example Workflow

The example BPMN 2.0 workflow illustrated in Figure 3 is taken from [26] where additional information like the source code is given. It solves a linear equation system. To not repeat previous work, only the minimum required information to understand this paper is given here.

The workflow consists of two script tasks, two service tasks, two parallel gateways, and the start as well as the end. The arrows indicate the task dependencies and the data flow which define the execution order of the tasks. A task can only start its execution after its predecessor has finished its own execution. The two script tasks are executed on the local computer. The two service tasks are executed on high-performance computation resources which can be on-premise or off-premise, e.g., in the cloud. The two parallel gateways split and merge the service tasks “Gauss” and “LuDecomposition”. That means that they can be executed independent of each other in an arbitrary order with no dependencies between them or even in parallel on different computers.

B. Used Software

The open-source flexible Business Process management (BPM) Suite jBPM of the JBoss community was used to evaluate the approach by running the example workflow of Figure 3. It provides an application server, a workflow engine to run workflows, an Eclipse Integrated Development Environment (IDE) with a Business Process Model and Notation 2.0 (BPMN 2.0) conform editor as plugin to model workflows, a data base to persist workflow runs, and a WS-HumanTask implementation to integrate human interactions into workflows in a standard conform way [8].

OpenNebula is an open-source software toolkit that enables the creation of Private, Public, and Hybrid Clouds [27]. This evaluation uses OpenNebula for local tests to simulate a Public Cloud provider on local resources to avoid expenses.

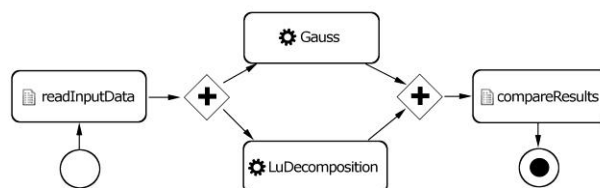


Figure 3. The example workflow consists of two script tasks, two service tasks, and two parallel gateways.

RESERVOIR is a FP7 project which bases on OpenNebula. “RESERVOIR’s open-source approach supports the definition of open standards for Cloud computing in order to break the lock-in imposed by vendors today and allowing any organization to build its own local or public cloud infrastructure [28].” It allows building “on-demand infrastructure services, reducing investment and operational costs, increasing energy efficiency and elasticity while ensuring security and Quality of Service” (QoS). Future versions of our prototype could replace OpenNebula with RESERVOIR to get access to a more advanced toolkit and to integrate public cloud infrastructure resources in a standard conform way.

The clients for the equation solver web services are created by Java API for XML Web Services (JAX-WS) using the Java interface, the web service endpoint, and the web service description language (WSDL) file.

The software implementation to extend workflows is presented in [16]. The cloud service broker was self-developed following the prototype described in [26].

C. Workflow run

Before the instantiation of the workflow, the preprocessor requests the workflow template, the workflow annotations, and the information about available resources of the scheduler. The workflow annotations allow both service tasks to be executed off-premise. The scheduler was configured to indicate only enough available resources for one of the service tasks, the “LuDecomposition”. That means that the “Gauss” task must be executed in the cloud which resources will be enabled during workflow runtime. The preprocessor then inserts the two administrative tasks “create” and “destroy” as predecessor and successor of the “Gauss” script task into the workflow template as only communication points between workflow and cloud broker. This new modified workflow template is then forwarded to the workflow management system for execution. The first script task reads the input data and forwards it to both service tasks. The “LuDecomposition” service task requests its service endpoint from the service registry. Since the endpoint is available on on-premise resources, the execution behavior of this service task is not influenced by the new architecture’s components. The merge control flow task stops the execution branch until the “Gauss” service task finishes execution. The administrative “create” task connects to the cloud broker and forwards the execution requirements of its assigned “Gauss” service task. The cloud broker performs the decision making algorithm described in Section II.H. Suppose neither a service nor a computer is available. So the cloud broker selects the best cloud provider, instantiates a resource, and deploys the software. In this example only the private OpenNebula cloud was available and therefore chosen. The cloud broker requests the endpoint of the cloud resource and publishes it at the service registry. Now the “create” administrative task finishes execution. The “Gauss” service task first requests the endpoint from the service registry to invoke the service. The service task does not know that it is executed off-premise because of the design decision to abstract endpoints with service names,

which are replaced during runtime. After the service returns the result to the workflow, the administrative task “destroy” notifies the cloud broker, that the service is no longer needed. The cloud broker terminates the cloud resource because no future cloud requests are predicted. Now all execution branches finished and the merge task starts the final script task which compares both results on the local computer.

IV. CONCLUSION AND FUTURE WORK

This paper presented a general concept for the hybrid execution of workflows by enabling Cloud resources to compensate a shortage of on-premise resources. The proposed prototype has the advantage that it neither depends on a particular workflow management system nor on a particular workflow description language. It follows the approach of automatically modifying workflow templates to incorporate the steps for dynamically enable the appropriate off-premise resources in a flexible manner. The cloud broker automatically selects the most suitable cloud resource to guarantee the fulfillment of all task requirements. The end users’ interfaces are not changed so that workflows can be used the same way as before.

Next steps of work will be an analysis of an according selection metric for the cloud broker to select the most suitable cloud service provider. The incurred costs of a partial off-premise execution will be compared with the costs of a full off-premise execution to calculate a costs reduction ratio and a cost-performance tradeoff. The time overhead for migrating tasks across cloud and organizational boundaries has to be measured for different providers and set it into relation with the avoided costs. Additionally, in the meantime developed technologies will be analyzed for a possible integration to benefit from related work.

ACKNOWLEDGMENT

This work was carried out in the context of HiX4AGWS [29]. HiX4AGWS is supported of the Federal Ministry of Education and Research in Germany. Grant No.: 17N3409.

REFERENCES

- [1] X. Liu, D. Yuan, G. Zhang, W. Li, D. Cao, Q. He, J. Chen, and Y. Yang, “The Design of Cloud Workflow Systems,” SpringerBriefs in Computer Science, November 2011
- [2] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “CloneCloud: Elastic Execution between Mobile Device and Cloud,” Proceedings of the sixth conference on Computer systems (EuroSys11), pp. 301-314, April 2011
- [3] D. Warnecke and O. Kao, “Nephele: Efficient Parallel Data Processing in the Cloud,” Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS09), pp. 8:1-8:10, November 2009
- [4] Business objective driven REilable and Intelligent grids for real busiNess (BREIN) FP7 project <http://www.eu-brein.com/> [retrieved: March, 2013]
- [5] P. Mell and T. Grance, National Institute of Standards and Technology (NIST), “The NIST Definition of Cloud Computing”, Special Publication 800-145, September 2011
- [6] M. Gerhards, V. Sander, and A. Belloum, “About the flexible Migration of Workflow Tasks to Clouds: Combining on- and off-premise Executions of Applications,” Proceedings of the

- Third International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012), pp. 82-87, July 2012
- [7] I. Foster and C. Kesselman, *The Grid 2*, ISBN 1-55860-933-4
- [8] Web Service HumanTask V1.1 Committee Specification, August 2010, <http://docs.oasis-open.org/bpel4people/ws-human-task-1.1.pdf> form [retrieved: March, 2013]
- [9] M. Gerhards, S. Skorupa, V. Sander, P. Pfeiffer, and A. Belloum, "Towards a Security Framework for a WS-HumanTask Processor," 7th International Conference on Network and Service Management (CNSM 2011), pp. 1-5, October 2011
- [10] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, M. Goode, and K. Keahey, "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy," 5th Annual PKI R&D Workshop, April 2006
- [11] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," *Journal of Grid Computing*, Vol. 3, Issue 3-4, pp. 171-200, September 2005
- [12] Generic Workflow Execution Service (GWES) <http://www.gridworkflow.org/kwfguid/gwes/docs/> [retrieved: March, 2013]
- [13] R. Strijkers, W. Toorop, A. van Hoof, P. Grosso, A. Belloum, D. Vasuining, C. de Laat, and R. Meijer, "AMOS: Using the Cloud for On-Demand Execution of e-Science Applications," Sixth International Conference on e-Science (e-Science), pp. 331-338, December 2010
- [14] OPTIMIS FP7 project <http://www.optimis-project.eu/> [retrieved: March, 2013]
- [15] R. Woolley, "Enterprise Service Bus (ESB) Product Evaluation Comparisons", October 2006
- [16] M. Gerhards, A. Belloum, F. Berretz, V. Sander, and S. Skorupa, "A History-tracing XML-based Provenance Framework for Workflows". The 5th Workshop on Workflows in Support of Large-Scale Science (WORKS), New Orleans, pp. 1-10, November 2010
- [17] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, *Job Submission Description Language (JSDL) Specification, Version 1.0*, 7 November 2005
- [18] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture", Applications, and Approaches, *Wireless Communications and Mobile Computing*, Oktober 2011, DOI: 10.1002/wcm.1203
- [19] Open Grid Forum (OFG), *Open Cloud Computing Interface (OCCI)*, June 2011
- [20] jclouds <http://www.jclouds.org/documentation/gettingstarted/what-is-jclouds/> [retrieved: March, 2013]
- [21] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation", 3rd International Conference on Cloud Computing (CLOUD), pp. 337-345, 2010
- [22] X. Bai, H. Yu, Y. Ji, and D. Marinescu, "Resource Matching and a Matchmaking Service for an Intelligent Grid", *World Academy of Science, Engineering and Technology 1*, pp. 666-669, 2005
- [23] S. Kumar and N. Kumar, "Network and Data Location Aware Job Scheduling in Grid: Improvement to GridWay Metascheduler", *International Journal of Grid and Distributed Computing*, Vol. 5, No. 1, pp. 87-100, March 2012
- [24] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in e-Science", *SIGMOD RECORD*, vol. 34, pp. 31-36, 2005
- [25] M. Gerhards, V. Sander, T. Matzerath, A. Belloum, D. Vasunin, A. Benabdelkader, "Provenance Opportunities for WS-VLAM: An Exploration of an e-Science and an e-Business Approach", *The 6th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pp. 57-66, November 2011
- [26] M. Gerhards, M. Jagodzinska, V. Sander, and A. Belloum, "Realizing the flexible Integration of Cloud Resources into Workflows", *Systemics and Informatics World Network (ISSN 2044-7272)*, Special Issue on Cloud Computing and Services, Dezember 2012 (in-press)
- [27] OpenNebula Enterprise Cloud and Datacenter Virtualization <http://www.opennebula.org> [retrieved: March, 2013]
- [28] RESERVOIR (Resources and Services Virtualization without Barriers) FP7 project <http://www.reservoir-fp7.eu/> [retrieved: March, 2013]
- [29] History-tracing XML for an Actor-driven Grid-enabled Workflow System (HiX4AGWS), <http://www.fh-aachen.de/en/research/projekt-hixforagws/> [retrieved: March, 2013]

Collaborative Autonomic Resource Management System for Mobile Cloud Computing

Ahmed Khalifa^{1,2}, Mohamed Eltoweissy¹

¹ The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia, USA

² Switching Department, National Telecommunication Institute, Cairo, Egypt

e-mail: {akhalifa, toweissy}@vt.edu

Abstract— Mobile cloud computing promises more effective and efficient utilization of the ever-increasing pool of computing resources available on modern mobile devices. To support mobile cloud computing, we propose a Collaborative Autonomic Resource Management System (CARMS), which automatically manages task scheduling and resource allocation to realize efficient cloud formation and computing in a dynamic mobile environment. CARMS utilizes our previously proposed Global Resource Positioning System (GRPS) to track current and future availability of mobile resources. In this paper, we present CARMS architecture and its associated Adaptive List-based Scheduling and Allocation Algorithm (ALSALAM) for adaptive task scheduling and resource allocation. ALSALAM uses the continually updated data from the loosely federated GRPS to automatically select appropriate mobile nodes to participate informing clouds, and to adjust both task scheduling and resource allocation according to the changing conditions due to the dynamicity of resources and tasks in an existing cloud. Our simulation results show that CARMS offers effective and efficient support for mobile cloud computing that has not yet been adequately provided by prior research.

Keywords- Mobile cloud computing; Resource management; Dynamic resource maps; Autonomic computing; Collaborative computing.

I. INTRODUCTION

Cloud computing enables the delivery of computing resources as a utility. This utility concept is expected to drastically bring down computing costs. Moreover, the computation resources of mobile devices are increasing, for example quad-core platforms and significantly enhanced storage and memory capabilities. Recently, principles of cloud computing have been extended to the mobile computing domain, leading to the emergence of Mobile Cloud Computing (MCC). A MCC system (MCCS) has been defined from different views in the literature [1]. One of these perspectives defines a MCCS as a way of outsourcing the computing power and storage from mobile devices into an infrastructure cloud of fixed supercomputers. Here, a mobile device is simply a terminal which accesses services offered in the cloud. Another view defines a MCCS as an infrastructure-less cloud that is formed locally by a group of mobile devices, sharing their computing resources to run applications. This paper adopts and extends the latter definition as follows: A MCCS is a shared pool of configurable computing resources that are harvested from available or potentially available local or remote nodes that

are either mobile or fixed over a network to provide on-demand computational services to users.

Mobile devices in MCCS are expected to have reasonably powerful capabilities, for example exploiting the virtually unlimited power supply in our vehicles making them good candidates for housing powerful on-board computers augmented with huge storage devices that may act as networked computing centers on wheels [15].

MCC has a dynamic nature as nodes, usually having heterogeneous capabilities, may join or leave the formed cloud varying its computing capabilities. Also, the number of reachable nodes may vary according to the mobility pattern of these nodes. Resource management systems for MCCS should support this dynamicity, hide the heterogeneity of resources, provide users with unified access, evaluate and predict the availability and performance of resources, and guarantee the quality of service to meet users' requests.

Research in resource management systems and algorithms for mobile cloud computing is still in its infancy. In [4], authors proposed a preliminary design for a framework to exploit resources of a collection of nearby mobile devices as a virtual ad hoc cloud computing provider. In [5], a mobile cloud computing framework was presented. Experiments for job sharing were conducted over an ad-hoc network linking a user group of mobile devices. The Hyrax platform [6] introduced the concept of using mobile devices as resource providers. The platform used a central server to coordinate data and jobs on connected mobile devices. Task scheduling and resource allocation algorithms were reported in [7-11]. These algorithms used cost, time, reliability and energy as criteria for selection.

Most of the existing resource management systems [4-6] for MCC were designed to select the available mobile resources in the same area or those follow the same movement pattern to overcome the instability of the mobile cloud environment. However, they did not consider more general scenarios of users' mobility where mobile resources should be automatically and dynamically discovered, scheduled, allocated in a distributed manner largely transparent to the users. Additionally, most current task scheduling and resource allocation algorithms [7-11] did not consider the prediction of resource availability or the connectivity among mobile nodes in the future, or the channel contention, which affects the performance of submitted applications. Consequently, there is a need for a solution that effectively and autonomically manages the high resource variations in a dynamic cloud environment. It should include autonomic components for resource

discovery, scheduling, allocation and monitoring to provide ubiquitously available resources to cloud users.

In an apparent departure from previous work, our Collaborative Autonomic Resource Management System (CARMS) provides a more general distributed solution to cloud formation and management based on dynamic calendars of available or potentially available resources. Our main contributions in this paper are:

(1) The CARMS architecture which provides system-managed cloud services such as configuration, adaptation and resilience through collaborative autonomic management of dynamic cloud resources, services and membership; and

(2) Adaptive task scheduling and resource allocation algorithm to map applications' requirements to the currently or potentially available mobile resources. This would support formed cloud stability in a dynamic resource environment.

The rest of the paper is organized as follows. Section II presents the architecture of CARMS. Section III presents ALSALAM; our proposed adaptive task scheduling and resource allocation algorithm. Section IV discusses the performance evaluation. Finally Section V concludes the paper and outlines future work.

II. COLLABORATIVE AUTONOMIC RESOURCE MANAGEMENT SYSTEM (CARMS)

In [12], we proposed the PlanetCloud concept to enable MCC to tap into the otherwise unreachable resources, which may be located on any opt-in reachable node, rather than being exclusively located on a static cloud service providers' side. A key PlanetCloud component was the Global Resource Positioning System (GRPS) that we presented in detail in [13]. GRPS adopts a spatiotemporal calendaring mechanism with real-time synchronization to support dynamic real-time recording and tracking of idle mobile or fixed resources. The calendar consists of records including data about time, location, and computing capabilities of GRPS participants. GRPS also forecasts the availability of resources, anytime and anywhere. GRPS makes use of the analysis of calendaring data coupled with data from other sources such as social networking to improve the prediction accuracy of resource availability. In addition, the GRPS provides hierarchical zone architecture with a synchronization protocol between different levels of zones to enable scalable resource-infinite computing.

In this paper, we describe and evaluate our CARMS integral to PlanetCloud. In PlanetCloud, a cloud application comprises a number of tasks. At the basic level, each task consists of a sequence of instructions that must be executed on the same node. Tasks of a submitted application are represented by nodes on a directed acyclic Graph (DAG) which is addressed in the next section. The set of communication edges among these nodes show the dependencies among the tasks. The edge $e_{i,j}$ joins nodes v_i and v_j , where v_i is called the immediate predecessor of v_j , and v_j is called the immediate successor of v_i . A task without any immediate predecessor is called an entry task, and a task without any immediate successors is called an exit

task. Only after all immediate predecessors of a task finish, that task can start its execution.

CARMS manages clouds of mobile or hybrid resources (resources of mobile and fixed nodes). A CARMS-managed cloud consists of resources on virtual nodes that meet the cloud applications' requirements. Each virtual node is emulated by a subset of the real physical mobile nodes. The subset locally stores the state of the emulated virtual node. The real nodes perform the tasks assigned to their emulated virtual node. If a mobile node fails or leaves the cloud, it ceases to emulate the virtual node; a mobile node that joins the cloud attempts to participate in the emulation. CARMS attempts to provide each subset with a sufficient number of real mobile nodes, such that in case of failure, a redundant node can be ready to substitute the failed node.

A Cloud Agent, as a requester to form a cloud, manages the formed cloud by keeping track of all the resources joining its cloud using the updates received from the GRPS.

We design our CARMS architecture using the key features, concepts and principles of autonomic computing systems. As shown in Fig. 1, components of the CARMS and GRPS architectures interact with each other to automatically manage resource allocation and task scheduling to affect cloud computing in a dynamic mobile environment.

CARMS interacts with the information-base which maintains the necessary information about a requested cloud. The information-base includes user information, e.g., personal information and subscribed services, etc. Also, it contains information about the formed cloud, e.g., SLAs, types of resources needed, the amount of each resource type needed, and billing plan for the service.

CARMS comprises two primary types of nodes: Cloud Agent and participant nodes. CARMS performs all required management functions using the components detailed below.

1) *Controller*: In order to obtain a self-controlled operation, a controller is needed to automatically take appropriate actions. These actions are taken according to results of the evaluation received from the Performance Analyzer, described below, due to variations in the performance and workload in a cloud environment. The Controller manages interactions to form, maintain and disassemble a cloud. Besides, it makes decisions according to the applied policies. The Controller provides both policy and participant control functions. The policy control function prevents conflicts and inconsistency when policies are updated due to change in the demands of a cloud. In addition, it distributes policies to other CARMS components. On the other hand, the participant control function manages the interaction between a cloud requester and resource providers, the cloud participants, to perform a Service Level Agreement (SLA) negotiation. Once the negotiation is successful, the participant control function updates the billing information and SLA of a participant in the information-base. Then, the Controller sends a cloud activation request to a Cloud Manager component.

2) *Cloud Manager*: decomposes the requested application, in a Cloud Agent, upon receiving a cloud activation request, to a set of tasks. This component can create some policies on the fly and assign a set of virtual resources to these tasks according to the received SLA information from the controller component. Then, the information of the virtual resources is sent to the resource manager component for the appropriate real mobile resources allocation or de-allocation.

3) *Resource Manager*: Real mobile resources need to be allocated to the requested application. On the other hand, tasks of a requested application need to be scheduled. The Resource Manager component handles the resource allocation and task scheduling processes on real mobile nodes. The Resource Manager consists of two main units:

a) *Resource Allocator*: allocates local real resources for a task. Also, the resource allocator obtains the required information about the available real resources from (potential) participants by interacting with a GRCS of GRPS system. The Resource Allocator interacts with the registry of Cloud Agent to store and retrieve the periodically updated data related to all participants within a cloud.

b) *Task Scheduler*: distributes tasks to the appropriate real mobile nodes and keeps a copy of these tasks in an image registry to retrieve them as needed such as in case of

failure.

4) *Monitoring Manager*: consists of the following two units:

a) *Performance Monitor*: monitors the performance measured by monitoring agents at resource providers. Then, it provides the results of these measurements to the Performance Analyzer component.

b) *Workload Monitor*: The workload information of the incoming request is periodically collected by the Workload Monitor component.

5) *Performance Analyzer*: continually analyzes the measurements received from the Monitoring Manager to detect the status of tasks and operations, and evaluate both the performance and SLA. The results are then sent to both the Controller and the Account Manager.

6) *Account Manager*: In case of violation of SLA, adjustments are needed to the bill of a particular participant. These adjustments are performed by the Account Manager component depending on the billing policies negotiated by the requester of cloud formation.

III. ADAPTIVE TASK SCHEDULING AND RESOURCE ALLOCATION ALGORITHM

A. Application Model

For simplicity, we start with a basic application model. The load of submitted application is defined by the following

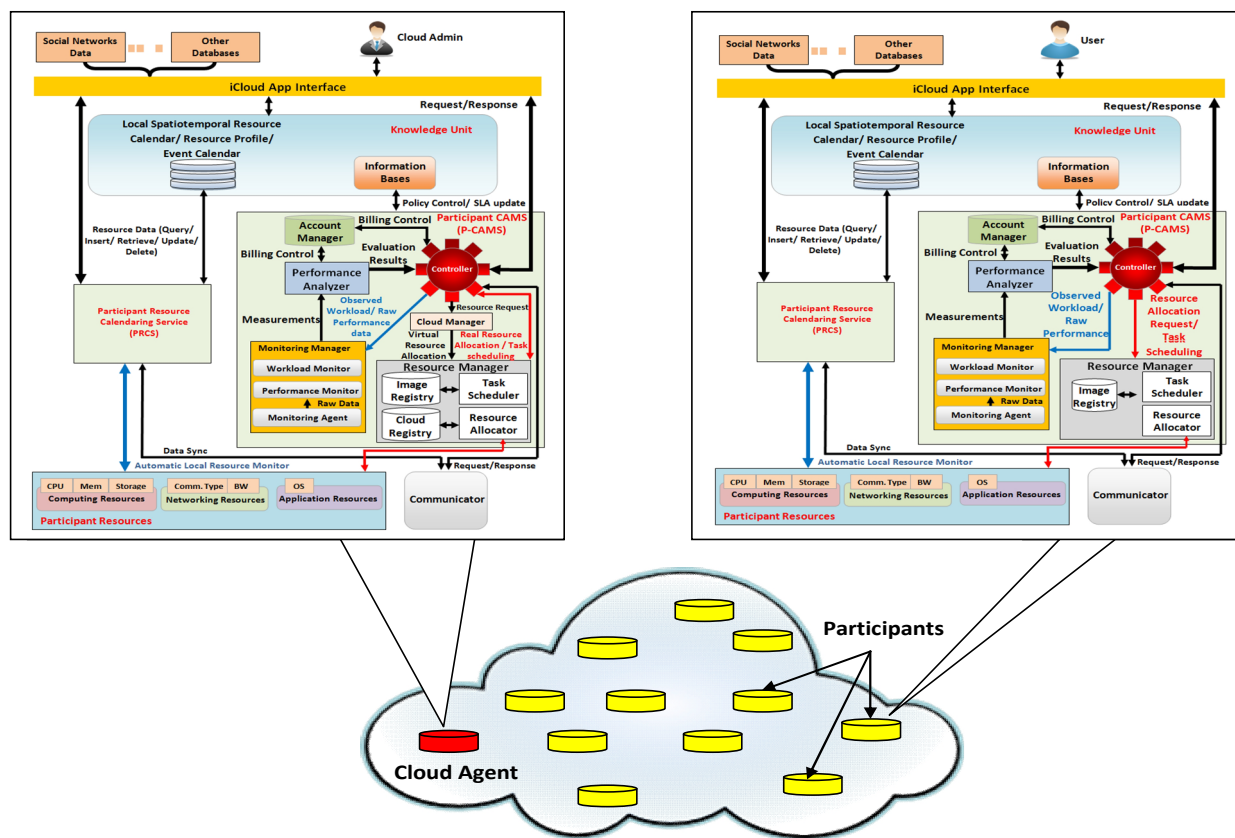


Figure 1. CARMS Architecture.

parameters: the number of submitted applications, the number of tasks per application, and the settings of each task. For example, the input and the output file size of a task before and after execution in bytes, the memory and the number of cores required to execute this task, and the execution time of a task.

Based on the criteria for selection, we mainly define two matrices: Criteria costs matrix, C , of size $v \times p$, i.e., $c_{i,j}$ gives the estimated time, cost, or energy consumption to execute task v_i on participant node p_j ; and a R matrix, of size $p \times p$, which includes criteria costs per transferred byte between any two participant nodes. For Example, time or cost to transfer n bytes of data from task v_i , scheduled on p_k , to task v_j , scheduled on p_l .

As an example of time-based selection criteria, a set of unlisted parent-trees is defined from the graph where a critical-node (CN) represents the root of each parent-tree. A CN refers to the node that has zero difference between its earliest start time (EST) and latest start time (LST). The EST of a task v_i is shown in (1). It refers to the earliest time that all predecessor tasks can be completed. ET is the average execution time of a task.

$$EST(v_i) = \max_{v_m \in pred(v_i)} \{EST(v_m) + ET(v_m)\} \quad (1)$$

Where $ET(v_m)$ is the average execution time of a task v_m , and $pred(v_i)$ is the set of immediate predecessors of v_i . The LST of a task v_i is shown in (2).

$$LST(v_i) = \max_{v_j \in succ(v_i)} \{LST(v_m)\} - ET(v_i) \quad (2)$$

Where $succ(v_i)$ is the set of immediate successors of v_i .

B. Resource Model

Our cloud system represents a heterogeneous environment since the mobile nodes have different characteristics and capabilities. The total computing capability of the real mobile nodes, hosts, within a cloud is a function of the number of hosts within a cloud and the configuration of their resources, i.e., memory, storage, bandwidth, number of CPUs/Cores, and the number of instructions a core can process per second.

C. Proposed Algorithm

We propose a generic GRPS-driven algorithm for the task scheduling and resource allocation: Adaptive List-based Scheduling and Allocation Algorithm (ALSALAM) for mobile cloud computing. ALSALAM supports the stability of a formed cloud in a dynamic resource environment. Where, a certain resource provider is selected to run a task based on resource discovery and forecasting information provided by the GRPS. The algorithm consists of two phases as follows.

1) Initial static scheduling and assignment phase

After, the information of virtual resources is sent to the Resource Manager for the appropriate real mobile nodes' resource allocation, the Resource Manager uses its Resource Allocator unit, which interacts with the GRPS to find the available resources of every possible node a Cloud Agent could reach. The information of location, time and the computing capabilities of these resources, which match the

application requirements, are obtained from GRPS. This information affects matrices of criteria for node selection. Based on the next waypoint, a destination obtained from GRPS, of each mobile node and the updated location of the Cloud Agent, we can estimate which mobile nodes will pass through the transmission range of the Cloud Agent.

A priority is assigned to a node depending on the criteria of selection. For example, in a time-based approach, we may select a host such that the highest priority is given to the nodes which are located inside the transmission range of a Cloud Agent, followed by the nodes which are located outside this transmission range and will cross it, and finally to the rest of the nodes. Within each group, nodes are listed in descending order according to the available computing capabilities, e.g. their number of cores or central processing units (CPUs). Nodes, with the same computing capabilities, are listed in descending order according to the time they will spend in the transmission range of a Cloud Agent. This could minimize the overall execution and communication time. As a result, a host list, H , is formed based on the priorities as shown in Algorithm 1 presented in Appendix.

The Cloud Agent sends the cloud formation requests, through its Communicator unit, to all resource providers to in the list of hosts H . According to the (earliest) responses received about resource available time from all responders and the criteria of selection, the responders' IDs are pushed by the Resource Manager in increasing order of parameters which reduce their costs. For example, CPUs in use in time-based approach, i.e. the responding node, R_{min} , with maximum free CPUs is on the top of responders stack RS , $top(RS)$. This could reduce the queuing delay and therefore enhance the overall execution time.

The Task Scheduler unit of the resource manager assigns and distributes the task at the top of the list of tasks L , $top(L)$ to the host at the top of responders stack RS , $top(RS)$.

2) Adaptive scheduling and reallocation phase

The actual measures, e.g., time, cost or energy, required to finish a task may differ from the estimated due to both the mobility of hosts and the resource contention. For example, the mobility of hosts affects the actual finish time of a task due to the delay a host takes to submit task results to other hosts in a MCCS.

The Estimated Finish Time of a task v_i on a node p_j , $EFT(v_i, p_j)$, is shown in (3), where ERAT is the earliest resource available time.

$$EFT(v_i, p_j) = ERAT(v_i, p_j) + ET(v_i, p_j) \quad (3)$$

We propose an adaptive task scheduling and resource allocation phase to adjust the resource allocation and reschedule the tasks dynamically based on both the updated measurements, provided by the Monitoring Manager, as well as the evaluation results performed by the Performance Analyzer. The Monitoring Manager aggregates the information about the current executed tasks periodically, as a pull mode. Due to the dynamic mobile environment, hosts of a cloud update the Monitoring Manager with any changes in the status of their tasks, as a push mode. Also, hosts periodically update the cloud registry of a Cloud Agent with any changes in the status of resources. Consequently, the

Performance Analyzer could re-calculate the estimated measures of the submitted tasks. As a result, tasks and resources could be rescheduled and reallocated according to the latest evaluation results and measurements.

IV. EVALUATION

To simulate the MCCS environment, we have extended the CloudSim simulator [14] to support the mobility of nodes by incorporating the Random Waypoint (RWP) model. A mobile node moves along a line from one waypoint W_i to the next W_{i+1} . These waypoints are uniformly distributed over a unit square area. At the start of each leg, a random velocity is drawn from a uniform velocity distribution.

In our evaluation model, an application is a set of tasks with one primary task. Each task, or cloudlet, runs in a single virtual machine (VM) which is deployed on a mobile node. VMs on mobile nodes could only communicate with the VM of the primary task node and only when a direct ad-hoc connection is established between them. For simplicity, a primary node collects the execution results from the other tasks which are executed on other mobile nodes in a cloud. There is only one cloud in this simulation. For scheduling any application on a VM, first-come, first-served (FCFS) is followed. We only considered the initial static scheduling and assignment phases through this part of the evaluation.

For calculating the collision delay, we consider the worst case scenario, a saturation condition, where each node has a packet to transmit in the transmission range.

A. Assumptions

- Communication between nodes is possible within a limited maximum communication range, x (km). Within this range, the communication is assumed to be error free and instantaneous.
- The distribution of speed is uniform.
- Every mobile node can always function well all the time with high reliability and does not fail.

B. Metrics and Parameters

Preliminarily, the evaluated metric is the average application execution time, which is the time elapsed from the application submission to the application completion.

We set parameters in the simulation according to the maximum and minimum values shown in Table I. The number of hosts represents the mobile nodes that provide their computing resources and participate in the cloud.

C. Experiments

We started our evaluation by studying the effect of collision delay due to channel contention on the performance of the submitted application. In this evaluation, all nodes have the same computing capabilities, i.e. homogeneous. Fig. 2 shows the average execution time of an application at a different number of nodes, ranging from 4 to 24 nodes, in a unit square area. The average speed of a mobile node equals 10 (m/sec). We set the transmission range to be 0.8 (km), which has been obtained from an evaluation not presented here due to space limitation. At this value, we can neglect the effect of the connectivity, i.e. a node is almost always

TABLE I. PARAMETERS

Parameters	Values	Parameters	Values
Density of nodes	4-40 (Nodes/Km ²)	Communication range	0.1-1 (km)
Number of Hosts/Cloud	4-24	Application Arrival Rate (Poisson distribution)	7 (Applications/sec)
Number of hosts /Application	2-20	Expected execution time for a task	800 (Sec)
Number of tasks/Application	4 – 140	Number of CPUs/Cores per host (Uniform distribution)	1-8
Number of applications/Cloud	1 – 14	Average Node Speed (Uniform distribution)	1.389,10,20 (m/sec)

connected with others. Fig. 2 shows that the worst performance is obtained when a host has a minimum number of cores, i.e. 1 core, and at a maximum number of tasks per application, i.e. 30. This is because at a small number of nodes, e.g. 4, most of the submitted tasks will be queued in a waiting list since just one core is available per task. The more the available nodes participate in the formed cloud, the more available cores to execute these tasks. Consequently, the average execution time of an application decreases with the increase of the number of mobile nodes. The collision delay should increase with node density, while results show that the collision delay is negligible if we compare it with the queuing delay. The results at 1 and 8 cores per host are very close to each other at a small number of tasks per application, at 4 tasks/application, since there is no effect of the queuing delay. Noticeable differences between these results and the others appear at a higher number of submitted tasks/application equals 15, at a number of cores/host equals 8, due to the significant effect of the mobility of hosts. The reason is that these tasks are assigned to more nodes in the formed cloud, and this leads to increase in the communication time until the primary node collects results from the other nodes. These results show that the collision delay is also negligible if we compare it with the communication delay. Conversely, the average execution time of an application decreases when the number of nodes

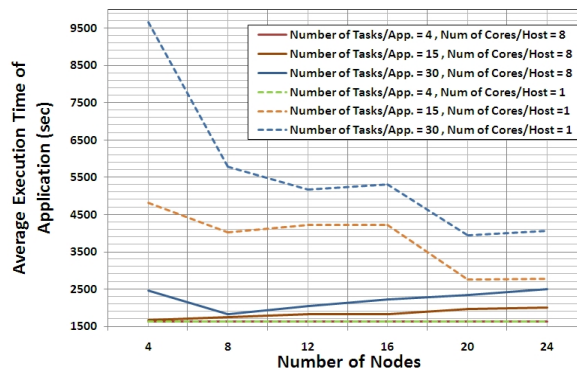


Figure 2. Average Execution Time of Application Vs Number of nodes at different number of submitted tasks/application and number of cores/host.

increases from 4 to 8 at a number of tasks per application equals 30, and at a number of cores/host equals 8. This is because the more the number of hosts, the more cores to execute these tasks. This reduces the queuing delay.

In the next experiments, we compare results of two cases: Using ALSALAM algorithm, which is based on the information obtained from GRPS, e.g. location and available processors, in resource scheduling and assignment and the random-based algorithm, which does not use this information, where a random mobile nodes are selected to execute the submitted application.

Let all 40 mobile nodes have a random number of cores, heterogeneous resources, ranging from 1 to 8 cores. Fig. 3 shows that the average execution time of an application when we consider one application is submitted to be executed. Each node has a transmission range equals 0.4 km, and its average speed equals 1.389 (m/sec). As expected, this evaluation provides significant differences between results of the two cases, with/without using the ALSALAM. The results of this figure show that executing the application on a smaller number of nodes, e.g. 8 hosts, has better performance in terms of average execution time of an application than in case of results at a larger number of hosts, i.e. 24 hosts. The higher number of submitted tasks per application leads to make some tasks waiting the previous ones in a waiting list to be executed. The total delay becomes higher if these tasks are distributed on a higher number of nodes, e.g. 24 hosts. This is because the communication delay is dominant.

We repeat our evaluation at a different number of hosts equals 4, 8 and 24 hosts, and at a different value of transmission ranges equals 0.4, and 1 (km). Fig. 4 shows that the average execution time of an application at a transmission range equals 1 (km) almost has a better performance than the case of a transmission range equals 0.4 (km) at the same number of hosts. Also, we can see that at a small transmission range, e.g. 0.4 (km), and a large number of hosts, e.g., 24 hosts, a worst performance is obtained. While, it has a better performance, at a number of hosts equals 8, than in case of a number of hosts equals 4. This observation is quite obvious because at this large number of tasks, greater than the total computing capabilities of the selected 4 hosts, the queuing delay is dominant. On the other hand, the larger the value of a number of hosts, at a high transmission range equals 1 (km), the better average

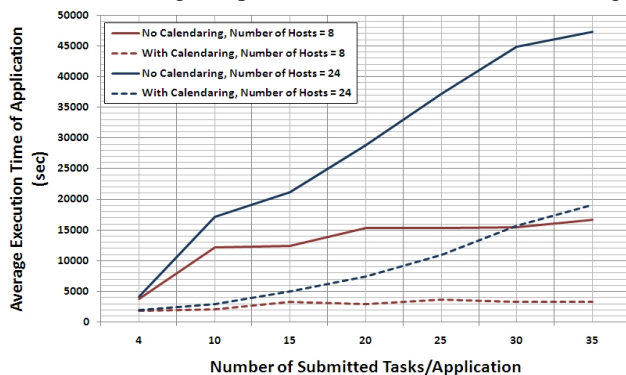


Figure 3. Average Execution Time of Applications Vs number of submitted tasks at different number of hosts.

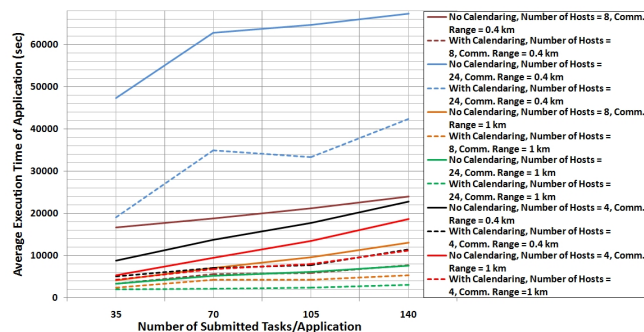


Figure 4. Average Execution Time of Applications Vs number of submitted tasks at different number of hosts and Comm. Ranges.

execution time of an application is, e.g. at 24 hosts.

The results of Fig. 5 show that the smaller the number of submitted applications, e.g. 7 applications, the better performance is obtained. Applications arrive into the system following a Poisson process with arrival rate 7. Also, the results show that the execution of submitted applications on a smaller number of hosts, e.g. 2 hosts/application, has a worst performance than of executing them on larger number of hosts, e.g. 8 hosts/application. This is because at a small number of hosts, e.g. 2, the queuing delay is dominant. The more the available number of hosts participated in the formed cloud the more available cores to execute these tasks. Consequently, the average execution time of an application decreases with the increase of a number of mobile nodes, e.g. 8 hosts/application. On the other hand, the larger the value of a number of hosts/application, the worst average execution time of an application is, e.g. at 20 hosts/application. This is because the communication delay is dominant.

D. Findings

Our findings can be summarized as follows.

1) There is a tradeoff between the communication delay and the queuing delay as a number of hosts per submitted application is varied. The higher number of hosts per an application, the higher total computing capability within the cloud is. Therefore, the queuing delay of a task is decreased. While, this leads to increase the time until the primary node collects results from other resource provider nodes, and therefore this increases the communication delay.

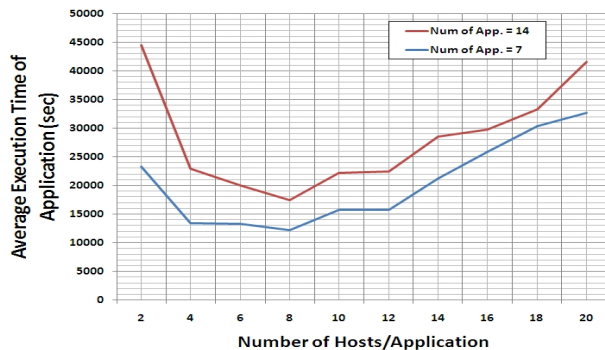


Figure 5. Average Execution Time of Applications Vs number of hosts per application at different number of applications.

2) A better performance may be obtained, at a shorter transmission range, if we select the smallest number of hosts that have computing capabilities which minimize the queuing delay to participate in a cloud. While at long transmission range of nodes, where the communication delay could be neglected, we have to select the highest number of hosts to maximize the computing capability and reduce the queuing delay.

3) The average execution time of an application is impacted by the connectivity among hosts of a cloud, the load of submitted applications, and the total resources, computing capabilities, confined in these hosts. The major factors affecting connectivity are hosts' transmission range, node mobility, and node density. The mobility is impacted by the hosts' speed and movement direction (relative to primary nodes).

V. CONCLUSION AND FUTURE WORK

We presented CARMS, a distributed autonomic resource management system to enable resilient dynamic resource allocation and task scheduling for mobile cloud computing. In addition, we proposed the GRPS-driven ALSALAM, an adaptive scheduling and allocation algorithm implemented in the resource manager of CARMS to enable efficient selection of cloud participants and to provide a stable cloud in a dynamic resource environment. Results have shown that CARMS enables effective and efficient cloud formation and maintenance over mobile devices.

Our ongoing research extends CARMS to enhance the resilience and cost efficiency of cloud management by considering the reliability and security aspects of mobile resources in the selection of cloud nodes while minimizing the execution and communication costs.

REFERENCES

[1] I. Chandrasekaran, "Mobile computing with cloud," *Advances in Parallel Distributed Computing, Communications in Computer and Information Science*, vol. 203, 2011, pp. 513-522.

[2] Y. Yuan and W. Liu, "Efficient resource management for cloud computing," *International Conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM)*, China, 2011, pp.233-236.

[3] Z. Liu, W. Tong, Z. Gong, J. Liu, Y. Hu, and S. Guo, "Cloud Computing Model without Resource Management Center," *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2011, pp.442-446.

[4] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," *Proc. 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, California, USA, 2010, pp.1-5.

[5] N. Fernando, S.W. Loke, and W. Rahayu, "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," *Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, Australia, 2011, pp.281-286.

[6] E. Marinelli, "Hyrax: cloud computing on mobile devices using MapReduce," *Master thesis, Carnegie Mellon University*, 2009.

[7] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid

clouds," *Journal of Internet Services and Applications*, vol. 2, Dec 2011, pp. 207-227.

[8] C. Lin, S. Lu, "Scheduling scientific workflows elastically for cloud computing," in *IEEE 4th International Conference on Cloud Computing*, USA, 2011, pp. 746 - 747.

[9] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," *5th International Conference on Cloud Computing*, USA, 2012, pp. 638-645.

[10] B. Yang, X. Xu, F. Tan, and D. H. Park, "An utility-based job scheduling algorithm for cloud computing considering reliability factor," *International Conference on Cloud and Service Computing (CSC)*, Hong Kong, 2011, pp. 95-102.

[11] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," *Proc. 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID'10*, Australia, 2010, pp. 368-377.

[12] A. Khalifa, R. Hassan, and M. Eltoweissy, "Towards ubiquitous computing clouds," in *The Third International Conference on Future Computational Technologies and Applications*, Rome, Italy, September, 2011, pp. 52-56.

[13] A. Khalifa and M. Eltoweissy, "A global resource positioning system for ubiquitous clouds," in *Eighth International Conference on Innovations in Information Technology (IIT)*, UAE, March, 2012, pp. 145-150.

[14] S. K. Garg and R. Buyya, "NetworkCloudSim: modelling parallel applications in cloud simulations," *Proc. 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011)*, Melbourne, Australia, Dec. 2011, pp.105-113.

[15] M. Eltoweissy, S.Olariu, and M.Younis, "Towards autonomous vehicular clouds," *Ad Hoc Networks, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 49, 2010, pp 1-16.

APPENDIX

Algorithm 1	Initial task scheduling and assignment based on priorities
1:	The EST of every task is calculated.
2:	The LST of every task is calculated.
3:	Empty list of tasks L and auxiliary stack S.
4:	Push tasks of CN tree into stack S in decreasing order of their LST.
5:	while the stack S is not empty do
6:	If there is unlisted predecessor of top(S) then
7:	Push the predecessor with least LST first into stack S
8:	else
9:	enqueue top(S) to the list L
10:	pop the top(S)
11:	end if
12:	end while
13:	while the list L is not empty do
14:	dequeue top(L).
15:	Send task requests of top(L) to all participant nodes in the list of hosts H which match the task requirements.
16:	Receive the earliest resource available time responses for top(L) from all responders.
17:	Empty auxiliary responders stack RS.
18:	Push IDs of hosts which respond to requests into responders stack RS in increasing order according to their CPUs in use.
19:	while the host stack RS is not empty do
20:	find the responder R_{min} with less CPUs in use.
21:	assign task top(L) to responder R_{min} .
22:	remove top(L) from the list L.
23:	end while
24:	end while

SLA Template Filtering: A Faceted Approach

Katerina Stamou, Verena Kantere and Jean-Henry Morin
Institute of Services Science, University of Geneva - HEC, Switzerland
Email: aikaterini.stamou, verena.kantere, jean-henry.morin @unige.ch

Abstract—With the commoditization of cloud computing, more and more companies prefer to outsource IT resources into virtual infrastructures. Service Level Agreements (SLAs) can be helpful to make the right investment decision. A SLA template represents the pre-agreed SLA state. A service provider proposes the SLA content and submits the template to a marketplace for customer consideration. Customers use SLA template views as "What You See Is What You Get" (WYSIWIG) snapshots prior to service selection and before agreement initialization. The paper proposes a filtering framework that is based on a faceted approach and that uses SLA templates to guide marketplace customers through available services. The framework design is presented along with the data-model of SLA templates. We report the results from testing the faceted filtering with two different SLA storage approaches and evaluate their appropriateness for the web application layer.

Keywords-cloud marketplaces; SLA templates; SLA modularity; faceted filtering; document database.

I. INTRODUCTION

A Service Level Agreement (SLA) accurately depicts how a service is going to be provisioned. Its explicit definition is necessary for both providers and consumers to measure and assess actual consumption of resources during service execution. The SLA description allows customers to have a clear idea before service commitment on how resources will be served. Hence SLAs can be helpful to make more informed investment decisions. Customers of service marketplaces can use SLA templates as "What You See Is What You Get" (WYSIWIG) snapshots when they navigate through available offers. We consider a SLA snapshot as a high level summary of a pre-agreed SLA.

Our discussion begins with the current role of SLAs in cloud marketplaces and with research challenges whose completion can advance the SLA utilization for IT services. The paper continues with the presentation of our filtering framework that uses the SLA template content to provide a multi-faceted navigation tool for customers. We position the framework within a service marketplace. A customer can filter views of available offers according to provisioning requirements. The goal of the faceted filtering is to gradually lead a customer to a reduced service offer list that is in accordance with customer requests, thus helping in the final service selection activity.

We describe the SLA template data model and the filtering framework design. Our analysis concentrates on how SLA

information is stored and managed by a marketplace to help customers orient their navigation according to their provisioning requirements. To examine the applicability of the proposed framework, we simulate its operation using two different data storage approaches and evaluate their appropriateness for the web marketplace setting.

The paper is organized as follows: in Section II we formalize our problem setting and elaborate on SLA research challenges that we consider towards a large scale reality of efficient SLA content manipulation. Section III presents the data model for SLA templates, their construction process, the design of the filtering framework and the proposed database schemas. Section IV describes our experimentation and reports on preliminary results. Section V acknowledges related scientific work that tries to answer relevant research questions around SLA manipulation. The paper concludes with on-going work.

II. PROBLEM FORMALIZATION

A. SLA and SLA template role in cloud markets

In the following, the terms 'SLA template' and 'service offer' are used interchangeably. A Service Level Agreement (SLA) identifies the exact measurement and enables the auditing of described resource parameter values. The SLA definition provides an explicit view on how the provisioning of a service is planned. It also indicates precise bounds of service levels that a provider can supply.

Providers use SLAs during service execution to monitor service measurable attributes. Currently, SLAs hardly appear in cloud marketplaces. Promotion of IT offers to customers relies primarily on high-level service descriptions. The role of SLAs is peripheral and they are often materialized by documents of "terms-and-conditions" that typically do not involve functional service aspects.

In the literature, a SLA template represents a pre-instantiated agreement that is submitted by a service provider to a marketplace for customer consideration. The SLA template describes the agreement content that a provider is willing to accept during communications with customers. Thus a template describes precisely a provider's resource availability and provisioning plan. To decide which provisioning is more suitable for their needs, customers review SLA templates as service offers and proceed with either agreement initialization or negotiation with one or more providers.

We consider SLA templates as dynamic information that is updated at frequent time intervals. A marketplace or equivalently a service aggregator platform can use such templates as customer drivers for service selection since SLA templates enclose all details on how services are to be provisioned. SLA templates can be efficiently manipulated given that they follow a modular structure. Template content modularity allows viewing service offer sections as facets.

According to [10], a facet represents a category of ordered information data. It may contain flat or hierarchical information and can be divided into subcategories or sub-hierarchies. Moreover, a facet is described by attributes. In [10] the authors analyze how they have used hierarchical faceted categories (HFC) to organize the information structure of a navigational interface [5] for large data collections. Faceted navigation is a design pattern that enables flexible browsing through a web interface. Big market vendors have employed this pattern as it allows friendly navigation through multiple data hierarchies simultaneously. The ordering of information in multi-hierarchies makes the faceted-navigation pattern suitable to use with SLA templates also, since the native SLA structure is represented in the literature as a tree hierarchy [2], [9]. Figure 1 illustrates a high-level overview of the SLA schema proposed by [2].

In a SLA tree structure, facets represent SLA branches that describe ordered aspects of provisioning details. Representation of SLA facets can be combined with filters to facilitate the customization of facet attributes. In addition, filters generate new SLA facets by following selected traversing routes in the SLA tree path. Motivated by the faceted navigation pattern and its noticeable suitability with the SLA tree structure, we provide a framework that manipulates modular SLA templates to enable service customer navigation through available service offers.

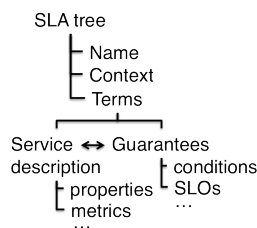


Figure 1. SLA tree structure according to WS-Agreement specification

Prior to service selection and agreement initialization, customers search through submitted offers to find services that match their business needs. The goal of SLA faceted filtering is to enable flexible service navigation that is driven by customer (either users or automated processes) provisioning requirements. The filtering process narrows down service offer views to only desired ones that fulfill requested provisioning parameters. A faceted navigation tool should provide filters that help customers indicate their

service provisioning requirements according to existing offer availability. Filtered navigation facilitates rapid traversing between different offer views.

B. SLA manipulation challenges

In the scientific literature, SLAs are hardly viewed as end user documents, but merely as automated processes that assist the monitoring and scheduling of resources. In contrast, cloud marketplaces treat SLAs as static documents that do not allow for any processing. One challenge is to find the right equilibrium between these two orthogonal aspects and combine machine-readable with user friendly SLAs into a uniform process that can be used by both backend systems and front-end web services.

SLAs represent nested tree structures that include heterogeneous characteristics and are unbounded in terms of length and content. In the cloud business setting, diversified services are offered. Description characteristics and provisioning guarantees vary considerably, even if they describe similar services in different contexts. Providers from different business domains use customized terminology to describe service parameters, metric functions and guarantee definitions. Terms like "availability", "throughput" or "performance" are usually included in ambiguous ways in service descriptions, which may be confusing for service customers. Various vocabularies of provisioning terms represent a primary cause for SLA heterogeneity. On a wide scale, SLA semantic and structural heterogeneity represents a challenge because it complicates SLA template comparison thus hindering any attempt to efficiently manipulate SLAs in open marketplaces.

SLA formulation highly depends on resource availability. Hence to manipulate SLA templates for customer interest, we need to first ensure that the template content can be updated dynamically. As the SLA depth is unbounded, frequent updates may cause performance delays in the information exchange between customers and providers. Thus, the storage schema of SLA templates represents a challenge. On one hand, one may argue that since SLA templates represent dynamic information objects, they should not be stored at all. Instead, they should be kept in-memory for as long as they are valid and then be immediately replaced. On the other hand, a modular SLA data model allows to persist SLA templates for longer time periods and to run frequent content updates according to provider resource capacity and provisioning availability. In this paper we work on the latter aspect.

Viewed as a tree hierarchy, the SLA content may include nested branching, which may lead to alternative information content. A challenge for the manipulation of SLA templates is to select a content structure that facilitates quick traversing within nested information routes. A modular structure provides independence between inner SLA components thus helping the exploitation of finer grained information. Mod-

ularity allows for categorization of SLA parts and indicates data management structures that may apply for diverse types of SLA formats.

SLA content heterogeneity addresses issues that deal with scientific opportunities for data management, information retrieval and language processing research. In addition, it highlights the need for SLA standardization. Currently, SLA formalization is not supported by a standard to allow classification of key performance indicators (KPIs) or to mandate inclusion of specific functions per business domain.

The scientific computing community has primarily used the WSLA [9] and WS-Agreement [2] language specifications to express SLAs. The GRAAP working group proposed the WS-Agreement specification [2] as a language and a protocol to conduct SLAs. The WSLA [9] language specification has been proposed by IBM research on utility computing. Both approaches denote SLA language semantics in XML notation. According to [9], a SLA complements a service description. Moreover, both specifications suggest the use of customer and provider templates for the exchange of counter offers in the process of agreeing on service levels.

III. FILTERED NAVIGATION AND TEMPLATE REPOSITORY

We propose a filtering process that is different from direct comparison and exchange of SLA templates. The process uses provider templates to construct filters, based on which customers express their provisioning preferences. The outcome of the filtering process does not represent the final selection decision of a customer, but rather a subset of available service offers that satisfy the imposed filters.

We assume homogeneity of template structure with respect to the ordering of sections and terms as proposed by [2]. Filters are created according to SLA facets. The following paragraphs describe the SLA template construction, the design of the filtering framework and the SLA template storage schema.

A. SLA template construction

[2], [9] propose that a SLA consists of three primary sections:

- (i) service description,
- (ii) guarantees or obligations and
- (iii) an informative section regarding involved parties and/or the provisioned service

[2] names the latter section SLA context. To construct SLA templates, we follow the WS-Agreement guidelines, but express the template content in JSON [8] notation. The SLA template construction steps can be summarized as following:

- 1) Parse XML sample into JSON
- 2) Use (1) to create SLA template data model
- 3) Create database schema according to (2)
- 4) Retrieve service descriptions from marketplace
- 5) Order data from (4) into service types
- 6) Create fictional information, order according to (5)

- 7) Shuffle information from (5) and (6) into randomly generated data lists
- 8) Load (7) into CSV files
- 9) Load (8) into database

The native WS-Agreement format comes in XML notation. Hence we initially parse a WS-Agreement template sample from XML to JSON. We use the JSON sample to create the data model for our SLA templates. From the native WS-Agreement specification, we employ the proposed sections of guarantees, description terms and agreement context, but order them accordingly to address the filtering need for modularity. Moreover, we extend the context section that we refer to as non-obligation attributes, and add information regarding the provider infrastructure and the customer data-storage location. We keep the service description joined with associated metrics and guarantees. Furthermore, we add a separate section for guarantees that apply to the overall service and that may or may not be measurable. We use this section to include customer monitoring options and provider obligations that indicate QoS bounds, e.g., service helpdesk availability. Customers typically need to be aware of such options before service commitment. Figure 2 illustrates the deduced SLA data model that we use to create the database schema for our templates.

The proposed SLA data model exploits information granularity by categorizing data into distinct SLA modules. This ordering allows for isolation of internal SLA root components, without depriving their inner depth in terms of nested branching. Nesting within a SLA template module depends on the information content. For example, non-obligation terms do not contain additional branches and remain consistent for all templates, regardless of service type. Service description and associated guarantees expand to multiple branches. The depth-level of nesting is of interest as it affects the template storage schema and hence the filtering flow process. Moreover, the suggested data model allows expanding the SLA content into distinct themes. Figure 2 depicts SLA data modularity with the letter N to indicate granularity of themes.

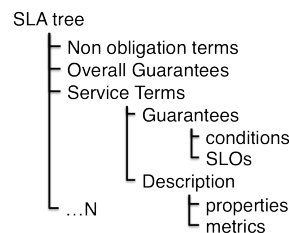


Figure 2. SLA data-model

Following, we retrieve information of service descriptions from the Amazon WS marketplace [1]. In particular, we derive service profiles that relate to storage, network and virtual machines. We order this information into nested

lists according to service type. Wherever necessary, we reformulate the retrieved data and complete them with fictitious information to cover the content of our SLA template.

We iteratively call a Python process to generate lists whose elements are assigned randomly from the classification of ordered data. We intentionally apply variations in the nesting depth of lists. Generated data are not skewed. Still, we acknowledge that in real market conditions services do not share the same level of popularity, thus customer preference. The Python process loads the generated data lists into comma separated value (CSV) files and from there to the database management system (DBMS) in use.

In this manner we create template sets for all three derived service profiles. Generated templates follow the proposed SLA data model, but differ in depth level and content. The template construction procedure simulates provider submissions of service offers into a cloud marketplace.

B. Filtered navigation through service offers

Figure 3 illustrates our proposed filtering framework. The framework design consists of two layers. One tier depicts all possible combinations of cloud stack layer [4], service type and offer validity as a three dimensional Cartesian coordinate system. The other tier presents the proposed SLA tree as a cube, where cubic sides indicate root-facets of filtering and service offer views. The cube selection is indicative of the proposed data model because a template may contain up to n SLA root-themes. The multidimensional structure depicts the inner-depth volume and interconnections of nested SLA information.

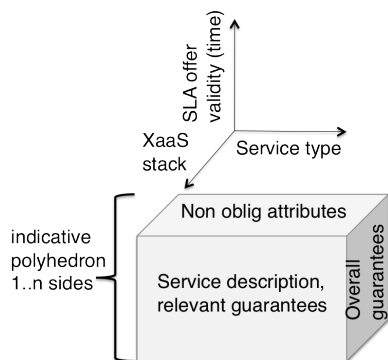


Figure 3. SLA filtering framework

Parameter combinations from the two filtering tiers indicate navigation and filtering options. For the suggested cubic representation, we point out the following entry-points:

- (a) Select a combination of cloud stack layer, service type and time interval. We take into account interdependencies that may exist between cloud stack layer and service type, since several services are mapped to a single cloud layer.

- (b) Select (a) and combine with filtering of non-obligation attributes.
- (c) Filter only non-obligation attributes.
- (d) Select (a) and combine with filtering of high-level service description terms.
- (e) Select (a) and combine with filtering of overall guarantee parameters.
- (f) Filter only overall guarantees.

The selection of entry-points designates one or more conditional queries that are processed transparently from customer actions, on the backend. The instantly returned result facilitates further navigation from a refined subset of existing offers, where deeper-level filtering options are provided. The navigation process gradually leads to a minimal set of preferred service offers that satisfy provisioning requirements according to submission of customer parameters. The method can be also deployed as an incremental process, where the system keeps track of customer selections on each step and accordingly regulates the flow of results.

The inherent modularity of the proposed SLA data model and its representation as a multidimensional structure allows for quick and selective navigation through designated nested information. At any point the navigation route can change by either selecting a different combination of SLA facet or by re-arranging filter values. The approach provides flexibility to navigate through available service offers from the provisioning aspect that a customer is mostly interested in.

Thus a customer may directly navigate through service profile attributes and associated provisioning guarantees by selecting the type of service and by filtering initial parameters from the description category. Alternatively, a customer may first look into non-obligation attributes if, for example, there is a provider or a data-location preference. Moreover, a customer may simply search for particular guarantee attributes that are irrelevant of service type.

Entry navigation and filtering points can be extended accordingly to the SLA structure branches or respectively SLA facets. Special facets can be introduced to illustrate provisioning guarantees that deal with service provider and customer concerns about, for example, energy efficiency or environmental impact.

C. SLA template storage

Filters in faceted navigation translate customer choices into conditional queries. In this work, we consider and experiment the faceted filtering with two different data management approaches.

In one case, SLA templates are stored and manipulated in a relational DBMS. Service offer information is kept into distinct tables and at a granular level of detail according to the template data model. In [2] a unique identifier (uid), located into the SLA context section, accompanies every SLA template. To resemble this relationship in the relational

database schema, we set a uid as the primary key (PK) of the non-obligation attributes table. Similar to [2], this PK acts as a reference key for the identification and matching of any incoming template. Moreover, the uid serves as foreign key (FK) to service description and overall guarantees tables that are associated with a specific template instance. This relationship resembles the native SLA tree structure of [2]. Figure 4 shows the relational database schema of our design.

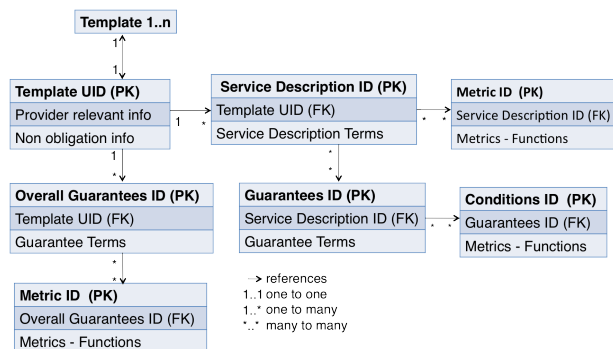


Figure 4. SLA template - relational database schema

To illustrate inner content branching, the PK of the service description table acts as a FK to associated rows of metric and guarantee tables. Similarly, the overall guarantees table is associated to metrics and function definitions for the measurement of referenced guarantees. In this order, the relational schema offers an alternative to the native XML structure proposed by [2]. This database design achieves the necessary granularity in terms of parameter details to allow for conditional queries on term and metric values.

We express and manipulate SLA templates using the JSON format. This choice was driven by our objective to test faceted filtering with a structured query language of a relational database and with a NoSQL data processing system, which in this work represents a document database. Since SLAs are machine-readable documents, a NoSQL DBMS may prove suitable for the marketplace scenario that typically operates over HTTP.

The document database design follows a nested dictionary structure. Compared to the relational schema, document collections represent tables and respectively documents represent records (table rows). The database design looks a lot like the relational one, but SLA templates are stored as nested documents. Although, schema conformance is not a pre-requisite for a document database, every stored document follows a generic SLA template structure and accurately corresponds to the information stored in the relational database. Figure 5 illustrates the NoSQL schema design.

Every stored document is accompanied by a unique identifier and embeds dictionaries (or sub-documents) to map FK

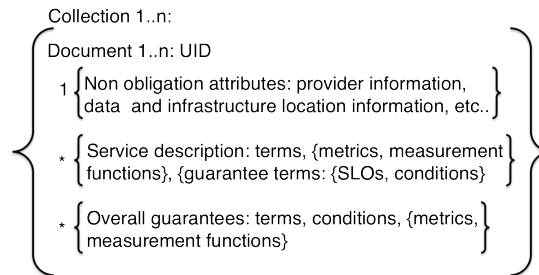


Figure 5. SLA template - document database schema

relationships from the relational database to the document schema. Each document contains one dictionary that holds non-obligation service attributes and one or more dictionaries to present service description parameters and overall guarantees. Similar to the relational model, description terms and overall guarantees enclose associated service attributes and respectively metrics, which in turn embed additional nested information.

IV. EXPERIMENTATION

A. Filtering simulation setup

We simulate the faceted filtering operation using the entry navigation-points analyzed in Section III. We assume that an IT marketplace provides SLA faceted navigation as an interaction tool for customers to submit their criteria and guide their browsing of service offers through provisioning requirements. We emulate customer instances and the SLA faceted filtering in a client - server architecture. Our goal is to measure the server response time to incoming customer requests and the scalability of the filtering operation as the number of simultaneous requests increase.

We setup the simulation environment on a 24-processor computing machine. The model of each processor is Intel Xeon and every processor runs at 2.50 GHz. The computing machine includes 128GB of RAM and operates on Ubuntu 12.04, Linux version 3.2.0. We deploy the Tornado web server [14] that is natively written in Python, to represent the server side of the simulated environment.

Filtering is accomplished by simultaneous processing of queries and our tests target the parallel handling of client requests. We prepare multithreaded Python scripts that use data from SLA facet attributes, generate random parameter values and pass them as HTTP GET requests to the web server. Randomly generated parameters simulate the customer filtering input. We keep the values of generated parameters within the value range of existing SLOs and description terms. This configuration does not guarantee that customer requests are always satisfied, because every incoming request submits a diverse number of SLA requirement values, whose combinations may not map to an existing SLA offer.

In every run the server receives parameter values from each incoming HTTP request, generates a conditional statement and sends it for processing to a DBMS. A server process reads the returned result set from the DBMS and updates the customer view with matching available offers. The server handles client requests with the help of common gateway interface (CGI) Python scripts, which are multi-threaded to assist the concurrent request serving.

SLA templates are submitted and updated on-demand, transparently from customer activities. Section III describes the process of creating our SLA templates and loading their content into a DBMS. In our simulation the marketplace uses a centralized data repository for the SLA template storage. Our datasets are derived from the stored template content.

We use MySQL DBMS [13] for the relational database and MongoDB [12] for the document database. Both DBMS are deployed on the same machine as Tornado to reduce TCP communication overhead. Measurements are derived from testing with each database separately. Each table in the MySQL database is loaded with approximately 150,000 records. In MongoDB this number amounts to 35,000 documents with an average document size of 1289.44 kb.

B. Experimental setup

The experimentation simulates the process of sending and handling concurrent client requests and returning the results over HTTP. The entry points that we introduced in Section III designate the main use cases of our testing. Every entry point represents a number of query parameter values that are passed to the server and from there to the respective database. Incoming parameters represent SLA facet attributes. Their number depends from the facet type and its nesting depth. We range incoming submissions between 2, 10 and 20 parameters.

We start from the upper, more generic, tier of the filtering framework (Figure 3) and submit 2 parameters to represent an initial choice of service type and offer expiration time. We gradually combine filtering attributes from both framework layers and reach nested template information. Our testing deals with different customer use cases. We simulate the case, where a customer has a provider and a data storage location preference and hence filters only attributes of the non-obligation facet, which in our case represents a submission of 5 up to 10 parameters.

We also consider a customer, who wants to look into service offers with specific description characteristics and explicit guarantee values. The customer selects a desired service type and filters attributes of the service description facet. Submission to the server ranges from 10 up to 20 parameters. We use the same parameter range to deal with the submission of overall guarantee criteria and to combine filtering attributes from different SLA facets.

We run the same number of experiments for both databases and categorize them in three test suites. In the

first test set we measure the total time of the faceted filtering operation over HTTP. The total time starts from the point a client request reaches the server up to the point the server returns the result to the client. Timings include HTTP and backend processing overhead.

The second test suite includes faceted filtering runs that are processed locally on the machine where the server and the two databases reside to avoid additional network overhead. The third test suite is also based on local runs, but measurements combine the query processing from filtering and database updates. We prepare an extra set of update statements for both databases to measure their potential overhead on the filtering operation. In each run update queries are processed in parallel to filtering requests and account for an extra 10% of workload on the total database processing. Local communication between server and DBMS is achieved via Unix sockets for the MySQL database and over localhost for MongoDB.

Queries in each DBMS are similar in terms of number and type of conditions, but the values of conditional parameters are randomly generated for every query. For the MySQL case, conditional queries take the form of SELECT statements, where the number of conditions varies according to the incoming parameters. For MongoDB queries are represented in binary JSON (BSON) format. The MongoDB alternative for SELECT statements is the formation of queries with the find() method. For every filtering point, we repeat the same test for 10 runs and take the average time from their accumulation. We also gradually increase the number of submitted HTTP requests. We begin with 100 simultaneous requests and reach up to 100,000 concurrent requests for both MySQL and MongoDB.

C. Observations and evaluation of results

The graph in Figure 6 shows the results for both MySQL and MongoDB from running with 2, 10 and 20 requested parameters over HTTP. The y-axis represents the average total time for each performed run and the x-axis indicates the gradually increased number of incoming requests that the web server receives. The average time is close to constant for both MySQL and MongoDB. Derived curves for all runs are fitted to highlight the small range of fluctuations in the query processing results.

The filtering operation over HTTP takes approximately 1.87 seconds less for queries that are processed in MongoDB compared to the average time in MySQL. This approximate time difference prevails for all HTTP runs regardless of the number of incoming requests or the number of simultaneously processed queries. The difference can be justified by the fact that when MongoDB retrieves a document from a collection, the whole document is loaded into memory along with any embedded dictionaries. Thus, retrieving information from any nested dictionary comes at a minimal cost as soon as the root document is loaded into memory.

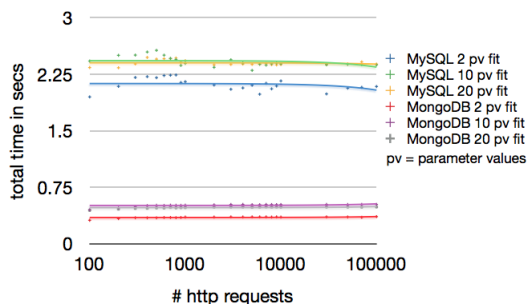


Figure 6. Average total filtering processing time over HTTP: MySQL and MongoDB

The embedding feature of MongoDB provides an alternative for MySQL JOINS [12].

Figures 7 and 8 present the results from the locally executed test sets. The average query processing time for MySQL is illustrated in Figure 7 and for MongoDB in Figure 8. For both graphs the y-axis represents the average query processing time in seconds and the x-axis the gradual increase in the number of submitted queries.

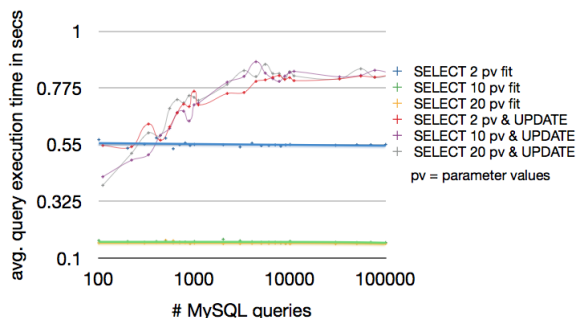


Figure 7. Average query processing time MySQL via UNIX sockets

The average query processing time for local faceted filtering (both SELECT and find() statements) is almost constant and in fact identical (0.16 seconds average) for both DBMS. The only exception is the MySQL SELECT query with 2 conditions, where the average time is nearly 0.38 seconds more than the SELECT queries with 10, 20 conditions and the respective find() statements in MongoDB. In both graphs, the curves that illustrate the average query processing time of the local faceted filtering are fitted to designate the small fluctuation range of the result set.

Figures 7 and 8 also illustrate the results from local runs that combine updates and faceted filtering. For both DBMS, updates are executed in randomly selected tables (respectively collections) and with randomly generated conditions. Updates affect multiple records of one or more tables (collections) but not those, where the SELECT/find() query operates. For both databases the results from the mixed

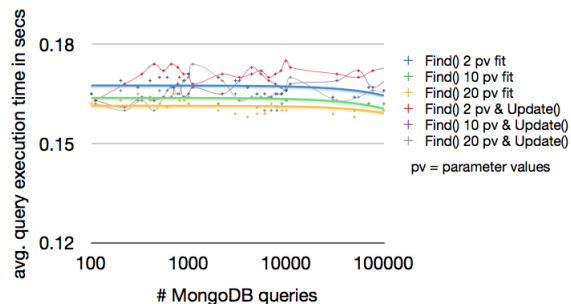


Figure 8. Average query execution time MongoDB over localhost

processing are not linear due to the random factor that affects the volume of updates. Compared to local faceted filtering, the cost of the update operation for MongoDB is negligible. For MySQL the cost is nearly constant at 0.57 seconds, with the exception of the SELECT statement with 2 conditions.

Our overall testing indicates that possibly a NoSQL approach like the MongoDB DBMS fits better for the web scenario, where SLA offers are manipulated over HTTP. In the local running mode, both DBMS share comparable performance. Still, the combination of updates with SELECT statements appears to be more expensive for MySQL than for MongoDB.

V. RELATED WORK

In [3], the authors propose an approach for automated matching of customer and provider templates by discovering semantically equivalent SLA parameters. The authors highlight that the absence of SLA standardization inevitably leads to variations in the definition of semantically related terms. They use a machine learning methodology to illustrate their matching comparison. The authors assume the existence of a knowledge repository that is responsible for managing incoming SLA templates and template mappings. As their work is focused on the comparison of SLA terms from diverse templates, the authors do not go into detail about the repository structure or the exposure of SLA parameters through a web interface.

In [11], a decision-support framework is proposed to assist the selection of infrastructure resources and the migration of services from local to virtual platforms. Although the approach is not explicitly directed towards SLA manipulation, the decision-support operation uses service attributes that are derived from provisioning parameters. The authors do not deal with customer navigation in a marketplace, but assume submission of service requirements by potential customers. Service attributes are structured in hierarchies. The authors apply the decision-support framework into a realistic use case to prototype the filtering of customer requirements on available service parameters. Still, they do not elaborate on how retrieved information is either stored or managed.

The work described in [6], [7] is a motivating schema for SLA-aware service-oriented infrastructures. In the proposed architecture, customer-provider interaction takes place over a service registry. The model can be extended to current conditions of service provisioning. SLA templates in the form of service offers are included in marketplaces and customers select services according to their provisioning preferences. A marketplace can then expose SLA offers in the same way a registry exposes service descriptions.

VI. CONCLUSIONS & ON-GOING WORK

The scope of the presented work has been to promote SLA aspects from post-agreement monitoring instruments to pre-agreement manipulation objects. SLA templates represent pre-initialized agreements and describe provisioning plans of service providers. We presented our SLA data model that assumes structure homogeneity and is based on the WS-Agreement language specification. Our data model supports modularity of internal components as this feature enables the extraction of SLA facets by categorizing information into distinct themes.

We described how we constructed SLA templates and used them with a faceted filtering framework that enables customers to browse through available services according to their provisioning requirements. Service customers utilize facet attributes as filters to express their objectives and to get views of preferred provisioning arrangements. We demonstrated use cases of filtering according to facet preferences that customers would like to be aware of before service commitment. The approach can be extended to include additional filtering criteria that influence provisioning expectations and are derived from non-SLA related objectives (e.g., risk, security, energy efficiency).

For the filtering experimentation we used two different DBMS approaches, a relational one represented by MySQL and a document one represented by MongoDB. We assumed that customer requests arrive concurrently and need to be served immediately. Both databases share their tradeoffs. MySQL is a seasoned DBMS, possibly suitable for back-end processing of SLA data. MongoDB represents a new product that appears to more efficient in terms of query processing time on web operations. Our results indicate that the MongoDB approach seems more suitable for SLA manipulation on the HTTP layer, where client requests reach the web server in large-scale mode and need to be handled simultaneously.

We continue the refinement of the SLA data model and the filtering framework experimentation with alternate modes of template persistence. An alternative to the NoSQL document approach is a database system that supports the Resource Description Framework (RDF) data structure. RDF encoding enables the representation of information in a graph form, where connections between nodes indicate semantic relationships. This attribute is of particular interest for SLAs, as

it supports the classification of SLA modules and promotes the creation of semantic vocabularies that can be associated in a distributed sharing mode.

Our next challenge is to extend the filtering framework into a recommendation mechanism that provides customer-tailored SLA suggestions by using a given user profile. The filtering framework can be considered as a pre-requisite of the recommendation system because it provides a tool to keep track of customer navigation behavior and filtering preferences.

ACKNOWLEDGEMENT

This work is supported by the Swiss National Science Foundation (SNSF), grant number 200021E-136316/1

REFERENCES

- [1] "Amazon WS Marketplace," accessed Sep. 2012, <https://aws.amazon.com/marketplace>.
- [2] A. Andrieux *et al.*, "Web Services Agreement Specification (WS-Agreement)," retrieved Oct. 2011, from <http://www.ogf.org/documents/GFD.192.pdf>, Open Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group.
- [3] C.Redl, I.Breskovic, I.Brandic, and S.Dustdar, "Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets," in *Proc. of the 2012 ACM/IEEE 13th International Conference on Grid Computing (GRID '12)*. IEEE Computer Society, 2012, pp. 85–94.
- [4] Fang Liu *et al.*, "SP 500-292 Cloud Computing Reference Architecture," retrieved Oct. 2011, from http://www.nist.gov/manuscript-publication-search.cfm?pub_id=909505, National Institute of Standards and Technology (NIST), Sep 2011.
- [5] "The Flamenco Search Interface Project," accessed Oct. 2012, <http://flamenco.berkeley.edu>.
- [6] H.Ludwig, "WS-Agreement Concepts and Use of Agreement-Based Service-Oriented Architectures," IBM Research, Tech. Rep., 2006.
- [7] H.Ludwig, A.Dan, and R.Kearney, "Cremona: an architecture and library for creation and monitoring of WS-agreements," in *Proc. of the 2nd International Conference on Service Oriented Computing (ICSOC '04)*. ACM, 2004, pp. 65–74.
- [8] "JavaScript Object Notation," accessed Aug. 2012, <http://www.json.org>.
- [9] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification," retrieved Oct. 2011, from <http://www.research.ibm.com/>, IBM Corporation, Tech. Rep., Jan 2003.
- [10] M.A.Hearst, "Clustering versus faceted categories for information exploration," *Commun. ACM*, vol. 49, no. 4, pp. 59–61, Apr. 2006.

- [11] M.Menzel and R.Ranjan, "CloudGenius: decision support for web server cloud migration," in *Proceedings of the 21st international conference on World Wide Web (WWW '12)*. ACM, 2012, pp. 979–988.
- [12] "MongoDB manual," accessed Sep. 2012, <http://docs.mongodb.org/manual>.
- [13] "MySQL," accessed Sep. 2006, <http://www.mysql.com>.
- [14] "Tornado web server," retrieved Sep. 2012, from <http://www.tornadoweb.org>.

CPU Utilization while Scaling Resources in the Cloud

Marjan Gusev, Sasko Ristov, Monika Simjanoska, and Goran Velkoski
 Faculty of Information Sciences and Computer Engineering
 Ss. Cyril and Methodius University
 Skopje, Macedonia

Email: marjan.gushev@finki.ukim.mk, sashko.ristov@finki.ukim.mk, m.simjanoska@gmail.com, velkoski.goran@gmail.com

Abstract—CPU utilization in a virtual machine instance directly impacts the overall cost for the cloud service provider since it generates costs for power consumption and cooling. We are interested to determine the total CPU utilization behavior while scaling the number of CPU cores using the same server load. The experiments are based on two simple web services to utilize the virtual machine instance varying the number of concurrent messages and their size. The goal is to check if the total CPU utilization while scaling will be sublinear (smaller than the number of cores), and if it is greater than the CPU utilization when executed without scaling (using only one CPU core) due to task scheduling, coherence, etc. The experiments prove only that the total CPU utilization will be sublinear. We observe a region (workload with smaller number of concurrent messages) where the total CPU utilization decreases while scaling, compared to the case without scaling. We also determine the correlation between the CPU utilization with message size and the number of concurrent messages.

Keywords—Cloud Computing; Performance; Web Services; Web Server.

I. INTRODUCTION

Cloud computing is a recent technological trend in which resources, such as CPU and storage, are provided as general utilities that can be leased and released on-demand by users according to their requirements [1]. The cloud is a promising approach for delivering ICT services by improving the utilization of data centre resources [2]. Scalability and elasticity are quality features in the cloud, since the cloud adjusts itself to achieve better performance whenever it detects a change in the environment [3]. Scaling the performance for growing problem size is an imperative [4], [5]. However, the resulting performance is not always acceptable for all applications hosted in the cloud [6].

While the cloud customer cost depends on the resources leased time, the cloud service provider cost mostly depends on CPU utilization of the active (leased) resources. That is, greater CPU utilization will increase not only the cost for power electricity, but also for cooling. Activating and utilizing more computing resources will increase the monthly costs of cloud data-center (approx.40% of costs are generated by power electricity and cooling). Reallocation of virtual machines and switching off the idle servers will save substantial energy [7]. Optimal resource allocation can

improve the performance using the same resources in the cloud [8]. Saleh et al. [9] have demonstrated that using some CPU utilization threshold to autoscale the resources is not an accurate measure since it can provide high cost and poor resource utilization.

Scaling the resources will reduce the CPU utilization per core, but we are interested if total CPU utilization will be also reduced or increased. We have set two hypotheses which we would like to check:

- H1 the total CPU utilization while scaling is sublinear (smaller than the number of cores); and
- H2 the total CPU utilization while scaling is greater than the CPU utilization when executed without scaling due to task scheduling, coherence, etc.

That is, we expect that the total CPU utilization will be in the range of $(U_1, U_1 \cdot n)$, where U_1 denotes the CPU utilization of virtual machine instance with one CPU allocated.

We realize several experiments to find the behavior of CPU utilization when scaling is applied, i.e., more powerful virtual machine instances (using more processor cores) are activated. The experiments are based on measurement of the CPU utilization while scaling from 1 to 2 and 4 CPU cores in a virtual machine instance. We use two simple web services to load the web server in virtual machine instances, i.e., *Concat* and *Sort*. The former concatenates two strings and the latter sorts the concatenation of two input strings. Both are memory demanding, and the second is also computationally intensive. We analyze the CPU utilization by varying the server load with different number of concurrent messages and input string size.

The rest of the paper is organized as follows. Related work is presented in Section II. In Section III, we describe the methodology used for testing. The experiments and the results are discussed in sections IV and V. In Section VI, we derive conclusion and we present future work.

II. RELATED WORK

Several papers analyzed CPU utilization on-premise and in the cloud, while loaded the same web services with various number of concurrent messages and message size. Gusev et al. [10] determined that the number of concurrent messages impacts directly to the CPU utilization for memory

demanding web services (*Concat*), while both the number of messages and their size impact the CPU utilization for both memory demanding and computation intensive web services (*Sort*). They determined that CPU utilization is always greater while hosted in the cloud compared to on-premise, for the same load and maximum allocated resources (4 CPUs). In this paper, we confirmed the same correlation of CPU utilization and the input parameters, and not only when maximum resources are allocated in particular virtual machine instance, but also when allocated with 1 or 2 CPUs. Velkoski et al. [11] analyzed the CPU utilization for the same (maximum) total amount of cloud resources, but orchestrated in different number of virtual machine instances with different size. They determined that allocating all resources into one "huge" virtual machine instance provides greater CPU utilization compared to the case where the same amount of resources are allocated to many "small" virtual machine instances for huge (the same) server load (number of concurrent messages) regardless of their size. In this paper, we analyze the CPU utilization of virtual machine instances with different number of CPUs, i.e., scaling the resources from 1 CPU to 2 and 4 CPUs, using the same server load for each virtual machine instance.

The CPU utilization is important factor for overall system performance and cost. Greater CPU utilization produces higher response times for load dependent resources [12]. A CPU bottleneck appears if its utilization goes beyond 80% for a sustained period of time [13]. De Sousa et al. [14] evaluated the CPU utilization of different virtual machine instances on Eucalyptus [15] platform considering different workloads with LINPACK as benchmark which solves dense system of linear equations. In this paper, we load web services hosted in different virtual machine instances on OpenStack cloud.

Many factors impact the CPU utilization and different server loads do not utilize the CPU equally. Even more, not all virtualized CPUs share the whole physical CPU. The small virtual machine instances of Amazon EC2 always get 40-50% of the physical CPU, while the most of medium virtual machine instances get 100% CPU sharing [16]. Hovestadt et al. [17] found that CPU utilization are not displayed accurately inside virtual machines instantiated with XEN, KVM, and in Amazon EC2.

Vilutis et al. [18] propose some of the project executions to be postponed in order to minimize the utilized resources and thus to reduce the overall cost. Balancing the load among more CPUs will also decrease their particular utilization. Jayasinghe et al. [19] analyzed the scalability of n-Tier applications while migrating in the cloud. They determined variations in CPU utilization in different tiers while scaling the resources. In this paper, we provide experimental research to find the behavior of total CPU utilization for the same load, but scaling the resources.

III. THE METHODOLOGY

This section presents the testing methodology used to obtain reliable results in each test case.

A. Technical Details

Client-server web service architecture is used as a testing environment. The server is deployed in OpenStack cloud [20] using KVM (Kernel-based Virtual Machine) hypervisor to instantiate virtual machine instances. The cloud nodes are installed with Ubuntu Server 12.04 operating system. Hardware computing resources consist of Intel(R) Xeon(R) CPU X5647 @ 2.93GHz with 4 cores and 8GB RAM. Virtual machine instances platform is Ubuntu Server 12.04 with Apache Tomcat 6.

The client uses SoapUI [21] to generate different server load. The client and the virtual machine instances are placed in the same LAN segment to minimize network latency [22].

B. Test Cases Definition

The *Concat* web service is memory demanding only service. It accepts two input strings and returns their concatenation. The *Sort* web service accepts two strings and returns their concatenation, alphabetically sorted, which makes it computationally intensive besides the increased memory demands.

Three test cases are defined scaling the number of CPUs per virtual machine instance that hosts the web services, with the following configuration:

- *Test Case 1* - virtual machine instance with 1 CPU (m1.small);
- *Test Case 2* - virtual machine instance with 2 CPUs (m1.medium); and
- *Test Case 3* - virtual machine instance with 4 CPUs (m1.large).

Each test case runs for 60 seconds. The test is repeated if the server replies with an error. Web server is loaded with N messages with parameters size of M bytes each. The range of parameters M and N is selected such that web server in virtual machine instance works in normal mode without replying error messages. Parameter M is measured in KB with the following values 0, 1, \dots , 9 for *Concat* web service and 0, 1, \dots , 6 for *Sort* web service. Both web services are loaded with $N = 12, 100, 500, 750, 1000, 1250, 1500, 1750$ and 2000 requests per second for each M . The values are selected to avoid CPU saturation.

C. Test Data

The CPU utilization is measured for each parameter size M , for different web service loads per second N , in each test case. While testing, we use *top* Ubuntu based utility to capture Tomcat process CPU utilization each 3 seconds, i.e. 20 values per test. An average utilization is calculated of all 20 values for each test case for both *Concat* and *Sort* web services distinctively.

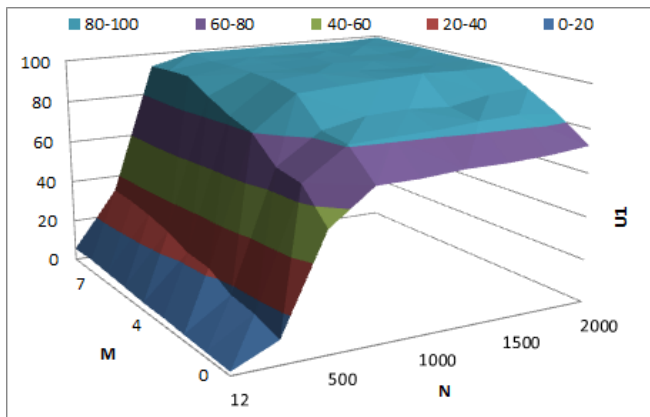


Figure 1. Normalized CPU utilization U_1 for *Concat* web service

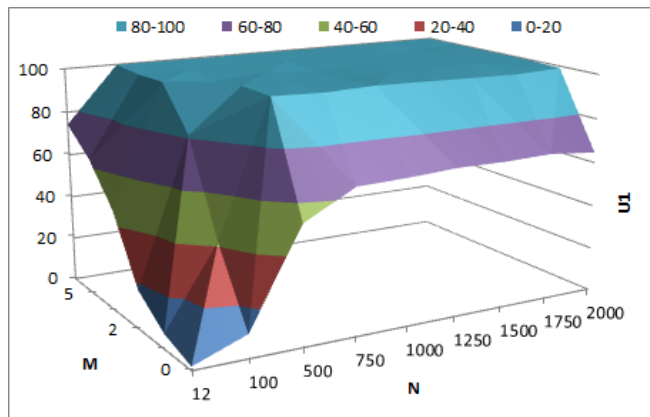


Figure 2. Normalized CPU utilization U_1 for *Sort* web service

D. Analysis of CPU Utilization

We use (1) to normalize CPU usage in range from 0% up to 100%. The nominator is the sum of CPU usage $U_i(n)$ of all n CPU cores, and the denominator n denotes the scaling factor, i.e., the number of CPUs used in particular test case ($n \in \{1, 2, 4\}$).

$$U_n = \frac{\sum_{i=1}^{i=n} U_i(n)}{n} \tag{1}$$

Furthermore, we define *Relative Scaled CPU utilization* $S(n)$ in (2) and calculate it for each test case in order to test both hypotheses, i.e., if $1 < S(n) < n$. U_1 denotes the CPU utilization without CPU scaling.

$$S(n) = \frac{\sum_{i=1}^{i=n} U_i(n)}{U_1} \tag{2}$$

We also define *Relative Multi-core Scaled CPU Utilization* in (3) and calculate it using relative scaled CPU utilization of test cases with scaling factors $n = 2$ and $n = 4$.

$$S_m = S(4)/S(2) \tag{3}$$

IV. EXPERIMENTS AND RESULTS

In this section, we exhibit the CPU utilization results of testing the web services for each test case in order to determine the correlation of CPU utilization with both parameters M and N for both web services hosted on virtual machine with particular number of CPU cores.

A. Test Case 1 - Without Scaling

Concat and *Sort* web services are hosted on 1 virtual machine instance with 1 CPU core in this test case.

1) *Concat Web Service*: Figure 1 depicts the normalized CPU utilization U_1 for *Concat* web service.

The results show that the CPU utilization depends on the number of concurrent messages N with huge increasing factor, and it proportionally increases when the input parameter M increases, but with small increasing factor. The minimum CPU utilization of $U_1 = 1.735\%$ occurs for $N = 12$ and $M = 0$, whereas maximum CPU utilization $U_1 = 99.23\%$ is measured for $N = 2000$ and $M = 9$, as expected.

2) *Sort Web Service*: Figure 2 depicts the normalized CPU utilization U_1 for *Sort* web service.

The results show that the CPU utilization strongly depends on both input parameters N and M . The dependence is expressed with huge increasing factor when changing the parameter M from $0KB$ to $1KB$, and also for $M \leq 2$ and $N \leq 500$. For the rest of the parameters, the increasing factor is small and continuously incremental. The minimum CPU utilization of $U_1 = 1.70\%$ is measured at $M = 0$ and $N = 12$, whereas the maximum CPU utilization $U_1 = 99.85\%$ is measured for $M = 6$ and $N = 2000$.

B. Test Case 2 - Scaling Factor 2

Both web services are hosted on a virtual machine instance with scaling factor 2, i.e., allocated with 2 CPU cores.

1) *Concat Web Service*: Figure 3 depicts the normalized CPU utilization U_2 for *Concat* web service.

The results show a minimum CPU utilization $U_2 = 1\%$ and maximum CPU utilization $U_2 = 85.43\%$ for parameters $N = 12$ and $M = 0$, and $N = 1000$ and $M = 9$, respectively. The dependence is the same as for U_1 , except for a small message size where it performs with decreased CPU utilization.

2) *Sort Web Service*: Figure 4 depicts the normalized CPU utilization U_2 for *Sort* web service.

The normalized results show a minimum CPU utilization $U_2 = 0.88\%$ for $M = 0$ and $N = 12$, and maximum CPU utilization $U_2 = 86.40\%$ for $M = 1$ and $N = 2000$. The dependence is also expressed as for U_1 .

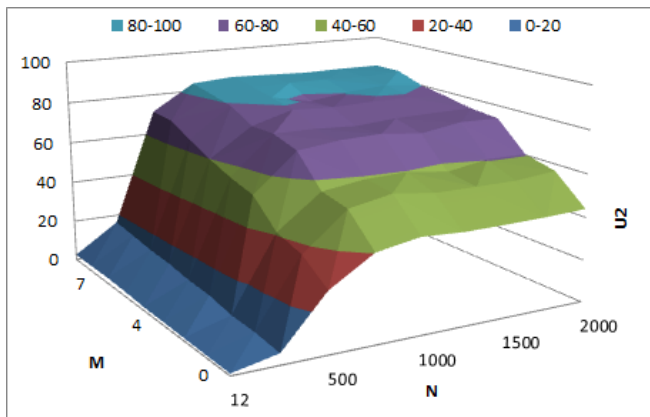


Figure 3. Normalized CPU utilization U_2 for *Concat* web service

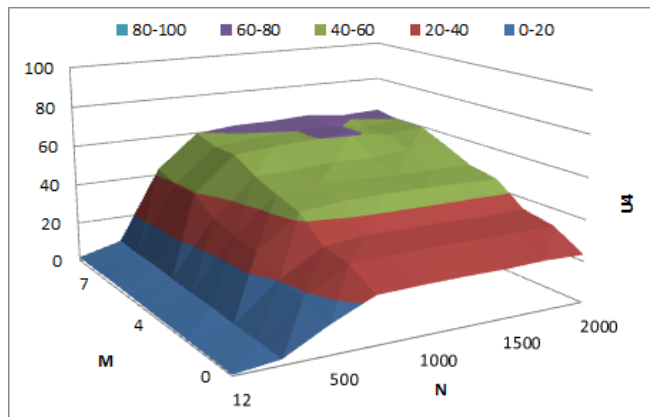


Figure 5. Normalized CPU utilization U_4 for *Concat* web service

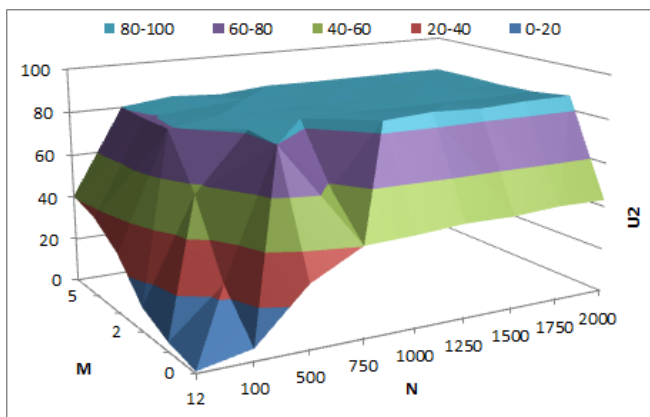


Figure 4. Normalized CPU utilization U_2 for *Sort* web service

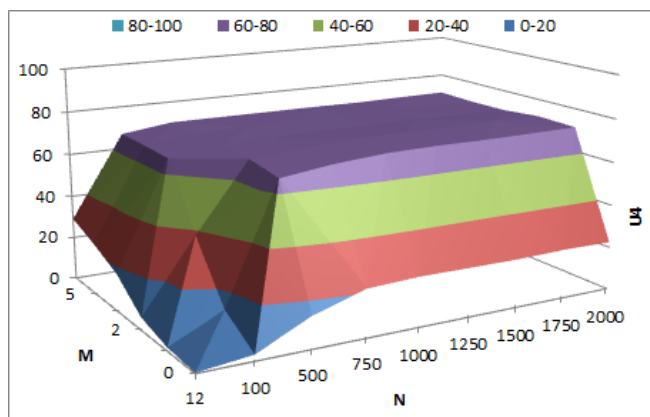


Figure 6. Normalized CPU utilization U_4 for *Sort* web service

C. Test Case 3 - Scaling Factor 4

Both web services are hosted on 1 virtual machine instance with scaling factor 4, i.e., allocated with 4 CPU cores.

1) *Concat* Web Service: Figure 5 depicts the normalized CPU utilization U_4 for *Concat* web service.

Similar increasing factor is observed compared to U_1 and U_2 . The minimum CPU utilization $U_4 = 0.54\%$ is measured for $N = 12$ and $M = 0$, and maximum $U_4 = 63.18\%$ is measured for $N = 1500$ and $M = 9$. For small message size it performs with decreased CPU utilization in comparison to the both U_1 and U_2 .

2) *Sort* Web Service: Figure 6 depicts the normalized CPU utilization U_4 for *Sort* web service.

The results show a minimum CPU utilization $U_4 = 0.33\%$ for $M = 0$ and $N = 12$, and maximum CPU utilization $U_4 = 72.46\%$ for $M = 2$ and $N = 1750$. The dependence and the increasing factor are similar as for U_1 and U_2 .

V. RELATIVE SCALED CPU UTILIZATION

In this section, we analyze the relative scaled CPU utilization while scaling the resources in test cases for both

web services.

A. Relative Scaled CPU Utilization for *Concat* Web Service

This section presents the relative scaled CPU utilization and relative multi-core CPU utilization for *Concat* web service for scaling factors $n = 2$ and $n = 4$.

1) *Scaling Factor 2*: Figure 7 presents the results for relative scaled CPU utilization $S(2)$ for *Concat* web service.

We observe that $S(2) < 2$ for each N and M , i.e., the hypothesis H1 is satisfied. However, very unexpected result is the region for smaller N regardless of M where $S(2) < 1$, i.e., the total CPU utilization with scaling factor 2 is reduced compared to CPU utilization without scaling. We can conclude that there is a region where the hypothesis H2 is not satisfied. Minimum and maximum values for relative scaled CPU utilization are $S(2) = 0.87$ and $S(2) = 1.73$, respectively.

2) *Scaling Factor 4*: Relative scaled CPU utilization $S(4)$ is depicted in Figure 8.

We also observe similar results, i.e., $S(4) < 4$ for each N and M , i.e., the hypothesis H1 is satisfied. The same region

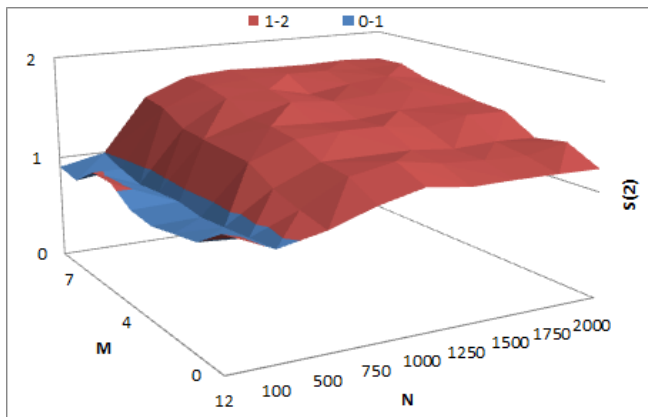


Figure 7. $S(2)$ for *Concat* web service

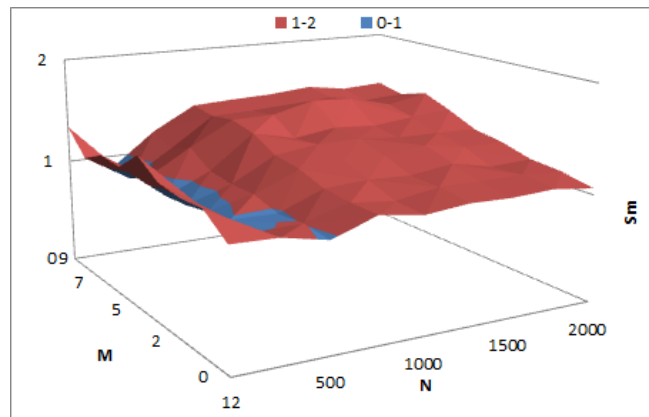


Figure 9. S_m for *Concat* web service

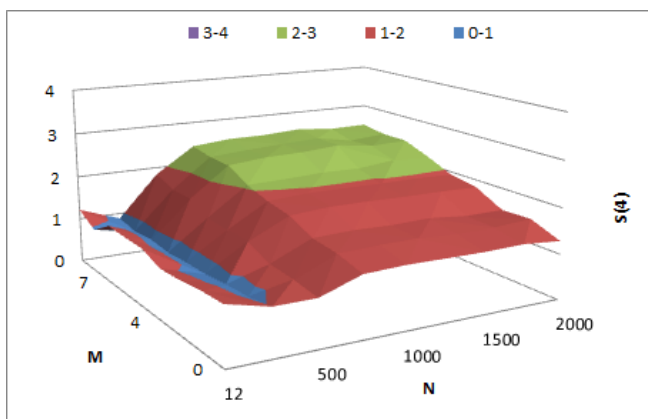


Figure 8. $S(4)$ for *Concat* web service

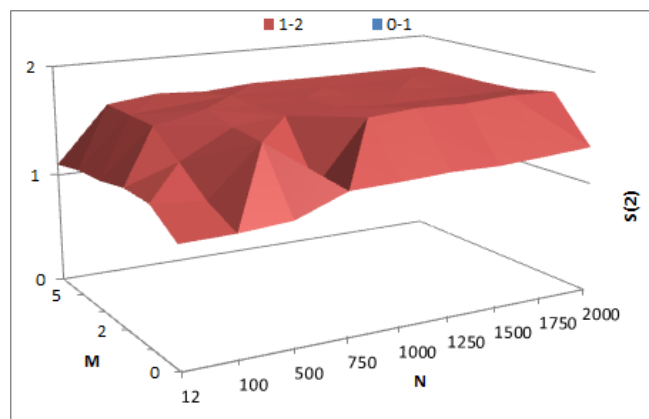


Figure 10. $S(2)$ for *Sort* web service

is observed where $S(4) < 1$ and the total CPU utilization with scaling factor 4 is reduced compared to CPU utilization without scaling. That is, the hypothesis H2 is not satisfied in the same region as scaling with 2 CPUs. Minimum and maximum values for relative scaled CPU utilization are $S(4) = 0.79$ and $S(4) = 2.56$, respectively.

3) *Relative Multi-core Scaled CPU Utilization S_m* : Figure 9 depicts the relative multi-core scaled CPU utilization S_m for *Concat* web service.

The similar conclusions can be derived as $S(2)$ and $S(4)$. We found that $S_m < 2$ for each value of parameters M and N , and also there is the similar region for smaller N where $S_m < 1$. That is, the hypothesis H1 is satisfied always, while the hypothesis H2 is not satisfied for smaller N . Minimum and maximum values for relative multi-core scaled CPU utilization are $S_m = 0.87$ and $S_m = 1.54$, respectively.

B. Relative Scaled CPU Utilization for *Sort* Web Service

This section presents the relative scaled CPU utilization and relative multi-core CPU utilization for *Sort* web service

for scaling factors $n = 2$ and $n = 4$.

1) *Scaling Factor 2*: Figure 10 depicts the relative scaled CPU utilization $S(2)$ for *Sort* web service hosted on a virtual machine with 2 cores.

We observe that the hypothesis H1 is also satisfied for each N and M as for *Concat* web service, i.e., $U_2 < 2$. However, opposite to *Concat* web service, $U_2 > 1$ for *Sort* web service, i.e., the total CPU utilization is always greater while scaling with 2 CPU cores. We can conclude that the hypothesis H2 is also satisfied for each N and M . The relative scaled CPU utilization is the smallest for smaller parameters M and N . Minimum and maximum values for relative scaled CPU utilization are $S(2) = 1.03$ and $S(2) = 1.76$, respectively.

2) *Scaling Factor 4*: The results for relative scaled CPU utilization $S(4)$ are depicted in Figure 11.

We can conclude that $S(4) < 4$ for all values of N and M , i.e., the hypothesis H1 is also satisfied. Even more, $S(4) < 3$ for each M and N . We found a region for the smallest M and N where $S(4) < 1$, i.e., the hypothesis H2 is not satisfied in this region. That is, the

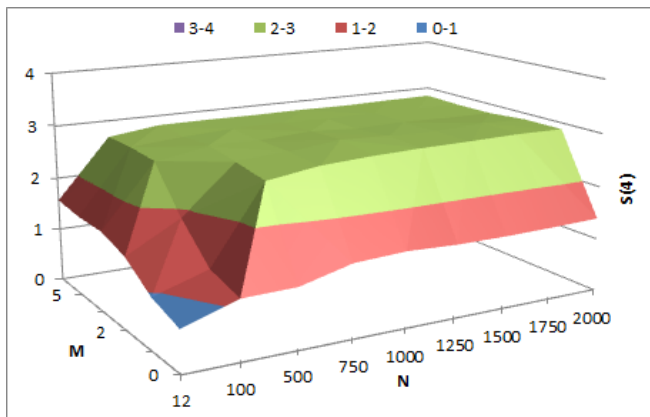


Figure 11. $S(4)$ for *Sort* web service

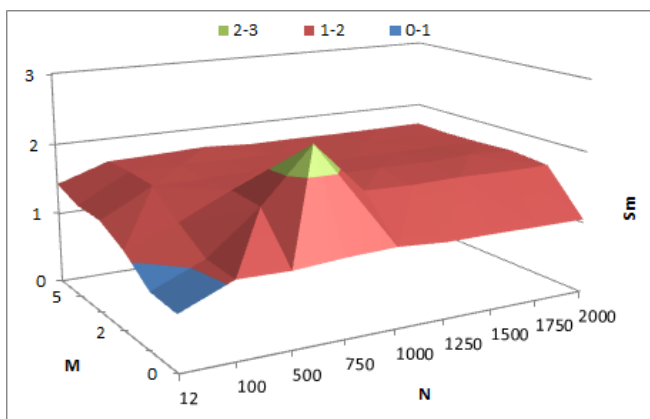


Figure 12. S_m for *Sort* web service

total CPU utilization is smaller while scaling the CPUs with factor 4. The relative scaled CPU utilization increases for greater M or N . Minimum and maximum values for relative scaled CPU utilization are $S(4) = 0.76$ and $S(4) = 2.92$, respectively.

3) *Relative Multi-core Scaled CPU Utilization S_m* : The similar results are observed for relative multi-core CPU utilization for *Sort* web service, as depicted in Figure 12.

There is a region for smaller M and N where $S_m < 1$, i.e., the hypothesis H2 is not satisfied. For all other values for N and M the relative multi-core CPU utilization is in the range $1 < S_m < 2$. That is, both hypotheses H1 and H2 are satisfied. A local extreme exists in point $(M, N) = (1, 750)$ where $S_m = 2.37 > 2$. Minimum and maximum values (excluding local extreme) for relative multi-core scaled CPU utilization are $S_m = 0.74$ and $S_m = 1.73$, respectively.

VI. CONCLUSION AND FUTURE WORK

In this paper, we measured and analyzed the CPU utilization with two web services (*Concat* and *Sort*) hosted on a virtual machine instance on the cloud, while the

number of CPUs was scaled from 1 to 2 and 4 CPU cores. Web services were tested with different load determined with various message parameter size M , and number of concurrent messages N .

We have introduced a relative scaled CPU utilization measure and a relative multi-core scaled CPU utilization for the same server load over scaled resources. The methodology is based on measurement of real CPU utilization and calculation of new relative measures to make better conclusions for scaling.

The results show that normalized CPU utilization depends mostly on the number of concurrent messages for *Concat* web service, while *Sort* web service depends on both input parameters.

Both expected and unexpected results are achieved for relative scaled CPU utilization. It is sublinear for each values of parameters N and M , proving the hypothesis H1. However, contrary to the hypothesis H2, the results show that there is a region where relative scaled CPU utilization is smaller than 1, i.e., the total CPU utilization is even smaller than unscaled test case. This region is determined for smaller N regardless of M for *Concat* web service, while the region for *Sort* web service is determined when both input parameters are small.

CPU utilization has directly impact to the power consumption, both for CPU working and cooling, which is a significant part of cloud total cost. Therefore, reducing the CPU utilization will greatly reduce the overall cost. In this paper, we determine the correlation between CPU utilization (cost) with the number of concurrent messages N and their parameter size M using two different web services *Concat* and *Sort*.

We will analyze the other performance parameters, such as response time for both web services in order to determine the tradeoffs between performance, cost and CPU utilization while scaling the resources on the cloud. Another important analysis will be performed to determine the platform impact (various operating systems and web servers) on CPU utilization in the cloud, using different clouds and hypervisors.

REFERENCES

- [1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, 2010, pp. 7–18.
- [2] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, vol. 53, no. 7, 2010, pp. 1045–1051.
- [3] L. Mei, W. Chan, and T. Tse, "A tale of clouds: Paradigm comparisons and some thoughts on research issues," in *Asia-Pacific Services Computing Conf.*, 2008. APSCC '08. IEEE, 2008, pp. 464–469.

- [4] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," *Cloud Computing*, 2010, pp. 115–131.
- [5] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Services-I, 2009 World Conference on*. IEEE, 2009, pp. 693–700.
- [6] D. Durkee, "Why cloud computing will never be free," *Queue*, vol. 8, no. 4, Apr. 2010, pp. 20:20–20:29.
- [7] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, may 2010, pp. 577–578.
- [8] M. Gusev and S. Ristov, "The optimal resource allocation among virtual machines in cloud computing," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 36–42.
- [9] K. Saleh and R. Boutaba, "Estimating service response time for elastic cloud applications," in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET) (IEEE CloudNet'12)*, Paris, France, Nov 2012, pp. 12–16.
- [10] M. Gusev, G. Velkoski, S. Ristov, and M. Simjanoska, "Web service CPU overutilization in the cloud," in the *6th International Conference on Information Technology*, ser. ICIT 2013, Amman, Jordan, in press.
- [11] G. Velkoski, M. Simjanoska, S. Ristov, and M. Gusev, "CPU utilization in a multitenant cloud," in *EUROCON - International Conference on Computer as a Tool (EUROCON)*, 2013 IEEE, in press.
- [12] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 91–98.
- [13] E. Ciliendo and T. Kunimasa, *Linux Performance and Tuning Guidelines*, 1st ed. ibm.com/redbooks, Jul. 2007.
- [14] E. T. G. de Sousa, P. R. M. Maciel, E. M. Medeiros, D. S. L. de Souza, F. A. A. Lins, and E. A. G. Tavares, "Evaluating eucalyptus virtual machine instance types: A study considering distinct workload demand," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 130–135.
- [15] Eucalyptus Systems. Eucalyptus cloud. [Retrieved: March, 2013]. [Online]. Available: <http://www.eucalyptus.com/>
- [16] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of amazon EC2 data center," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1163–1171.
- [17] M. Hovestadt, O. Kao, A. Kliem, and D. Warneke, "Evaluating adaptive compression to mitigate the effects of shared I/O in clouds," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, 2011 IEEE International Symposium on, 2011, pp. 1042–1051.
- [18] G. Vilutis, L. Daugirdas, R. Kavaliunas, K. Sutiene, and M. Vaidelys, "Model of load balancing and scheduling in cloud computing," in *Information Technology Interfaces (ITI)*, *Proceedings of the ITI 2012 34th International Conference on*, june 2012, pp. 117–122.
- [19] D. Jayasinghe, S. Malkowski, Q. Wang, J. Li, P. Xiong, and C. Pu, "Variations in performance and scalability when migrating n-Tier applications to different clouds," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, ser. CLOUD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 73–80.
- [20] OpenStack Cloud Software, "Openstack cloud," [Retrieved: March, 2013]. [Online]. Available: <http://openstack.org>
- [21] SoapUI, "Functional testing tool for web service testing," [Retrieved: March, 2013]. [Online]. Available: <http://www.soapui.org/>
- [22] M. Juric, I. Rozman, B. Brumen, M. Colnarić, and M. Hericko, "Comparison of performance of web services, WS-security, RMI, and RMI-SSL," *Journal of Systems and Software*, vol. 79, no. 5, 2006, pp. 689–700.

Evaluating Computation Offloading Trade-offs in Mobile Cloud Computing: A Sample Application

Jorge Luzuriaga, Juan Carlos Cano, Carlos Calafate, Pietro Manzoni

Universitat Politècnica de València

Dept. of Computer Engineering

Valencia, SPAIN

Emails: jorlu@posgrado.upv.es, jucano@disca.upv.es, calafate@disca.upv.es, pmanzoni@disca.upv.es

Abstract—Mobile cloud computing is generally referred to as the infrastructure where both the data storage and the data processing happen outside of the mobile device. The nature of the connection between the cloud servers and the mobile host are anyway much less reliable than in classical cloud computing with static devices. A compromise must be found between local versus remote computation so to cope with the reduced performance of the data connection and with the characteristics of the mobile device, basically its power availability limitations. In this paper, we evaluate the tradeoffs of computation offloading using as a case study a facial recognition application for smartphones where recognition is a service in the cloud. We present a specifically designed application for mobile devices developed as a component of the proposed evaluation system. The intensive calculus needed for the image manipulation is compared in terms of speed and accuracy both when we delegate it to cloud computing and when we perform it locally on the mobile device. These two alternatives and the intermediate options are compared to determine the optimal settings to take better advantage of integrating these two technologies.

Keywords—Cloud Computing; Facial Recognition; Mobile devices Applications; Process Outsourcing; Mobile Cloud Computing.

I. INTRODUCTION

The latest advances in mobile communication networks and the increasing penetration of smartphones and other mobile devices, like tablets and portable computers, are transforming the mobile Internet and allowing the users to improve their mobile experience. However, the limited computing and information/energy storage capabilities of mobile devices are delaying their abilities to support increasingly sophisticated applications demanded by users. The emerging cloud computing technology offers a natural solution to extend the limited capabilities of mobile devices. The resulting new paradigm of mobile cloud computing is being adopted by researchers as a powerful new way to extend the capabilities of mobile devices and mobile platforms, which has the potential of a deep impact on the business environment and people's daily life.

The decision of where to place the execution (local or remote mode) should be anyway made based on the quantity of computation and communication that is required by the application. A little amount of communication combined with a large amount of computation should be performed preferably in remote mode, while a large amount of communication combined with a little amount of computation should be performed preferably in local mode.

In this work, we chose face recognition as a sample application to evaluate the tradeoff of offloading computation with the intuitive idea of the required intensive calculus puts in commitment the hardware features of the mobile device. Whereas that, if the same calculus are executed by other systems with better hardware features, these processes are realized with less effort and in much less time.

We analyze the intensive calculus dividing it in sub processes that are distributed between the mobile device and the cloud infrastructure using a cascade of classifiers based on the Adaboost algorithm [19] to detect the presence of faces in an image and the Eigenfaces algorithm [11] to make the training and recognition of these faces.

Finally, we emulate the wireless channel between the mobile device and the cloud server to view how the end-to-end response time can affect at application. And also this emulation allow us to find limitations where we can get advantage with the use of this technique.

The rest of the paper is organized as follows: Section II presents the works related to the topic. Section III describes the proposed system overview, Section IV shows a sample case study: a facial recognition application. In Section V, we present a testbed to evaluate this proposal and in Section VI, we show the results obtained in the tests. The article finalizes with the conclusions in Section VII.

II. RELATED WORK

The computation off-loading from mobile devices to computational cloud infrastructure is based on deciding which methods should be remotely executed, so that benefits can be achieved in terms of both time and use of resources that ends up in saving energy. In the literature, we found a wide set of proposals with frameworks that decide dynamically whether a part of application will be executed locally or remotely [5][8][12][13][14]. Other proposals utilize nearby computers, also known as surrogates. These surrogates are resource-rich computers connected to the Internet and available to use with nearby mobile devices without incurring in WAN delays and jitter. Their objective is similar to proxy servers [15]. More aggressive proposals in which the entire user mobile devices are cloned on a remote server operating in a cloud where the execution of processes would be faster [3][4]. It enables moving an entire operative system and all its applications to selectively execute some processes on the clones, and integrating the results back into the mobile device.

Also, we find in [10] a cost/benefit analysis focused on energy saving for off-load computation to a server, taking into account: the processor speed, the instruction number, the bytes of data exchanged among server and mobile system, the network bandwidth, the energy consumption in different states, reaching the conclusion that energy-saved with off-load computation is greater than the energy spent on communicating with the cloud. So, if the computation is offloaded with relatively low cost, the processing in remote mode may be energy-efficient to the mobile device.

Finally, commercial proposals with applications implemented in client-server model, where the data to make the computation are transferred from a smartphone as a light weight client to the remote heavy weight server hosted in the cloud. These services typically have to interact with very large databases and require maintenance costs. An example of this type of applications is the popular Google goggles that uses pictures to search the web [6].

However, these proposals can only be used when the user is permanently connected to the Internet. Moreover, when we delegate the processing to external entities in many cases these entities follow the pay per use philosophy and their services are offered in monthly plans with a limit number of queries or charge for each query. These services provide a better performance and request priority with technical support in problematic situations. From the point of view of the user, the condition to “pay per use” of any application adds to the cost of the data line, making the remote processing economically speaking more expensive compared with local processing. Finally, it is necessary to consider the security issues in the communication process. When the information is transferred from the mobile device to the server in cloud environments, the data can be sniffed from a person/machine that listens the communication channel, then the data transfers to remote processing is less reliable compared to process the data locally whereas not necessary any transmission by the wireless network.

III. FACIAL RECOGNITION

Face recognition refers to the automatic identification of a person from a digital image. The process involves: (i) face detection, (ii) feature extraction, (iii) creation of the database with known faces, and the (iv) matching with the new face.

We use the Eigenfaces algorithm that applies data dimension reduction with the minimum information lost by PCA Principal Components Analysis [8] to get the coefficients values and is able to make the matching based on the minimum distance.

The Eigenfaces process chooses the factors with high correlation because with the redundant information that exists among them it is possible to select the coefficients that contain the maximum variability and in this form get the dimension reduction of the data. Once selected, the principal coefficients are representing in matrix form.

The process of automatic recognition can be clearly separated into two stages: the training and the recognition stage. The training stage is required to learn using a classifier [19]. In our case, the learning consists in transforming the features

of human faces into the form of coefficients and to store them in a matrix. This matrix represents a database of facial features of known faces. In the recognition stage, the classifier is used again to classify the data of the test image and to get values of correlation coefficients that represent the face found in the image; this stage realizes the features extraction. Finally, the values obtained are compared with the matrix values of the training stage. The minimal difference among these comparisons is the result of recognition process.

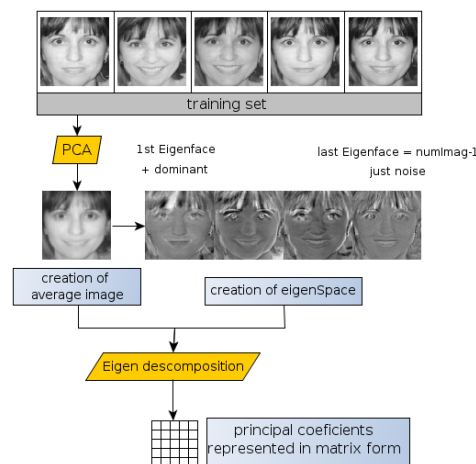


Fig. 1: The general process of training and recognition.

In the upper part of Figure 1, we have a training set of 5 images of the same person. To work with a standard input, these images have previously passed various sub processes: (i) compression, (ii) grayscale, (iii) crop, (iv) resize, and (v) equalization. The next step is to apply the process called Principal Components Analysis (PCA) where the principal features of each image are extracted and represented as another set of images with ghostly aspect. These images contain only relevant information to recognition process. The complete set of these images is called Eigenspace. Also, the PCA creates a common image that represents all images in the Eigenspace. In the Eigenspace, the first image is the most dominant and contains the representative features of all faces and the last images are the weakest that contain common features that can be found in any face. According to the specified threshold the weakest images cannot be taken into account because they are considered as noise. The last step is to get the numeric coefficient of each image in the Eigenspace through Eigen decomposing and represent in matrix form. So, each column of the matrix represents one image in the Eigenspace. A complete explanation of Eigenfaces process can be found in [11].

IV. EVALUATION APPROACH

When we delegate the processing to external entities we benefits in terms of hardware resources use and inclusive in terms of energy saving, but the latency in communication with the external entity is a big overhead especially when large data are involved. To reduce this problem a preprocessing of the information is required so that the accuracy results in posterior processes are not affected.

The cloud-based platform hosts the application in a centralized form and provides a software delivery model known as Software as a Service or SaaS in which the customers or users can access remotely using a thin client over the internet [9]. The main advantages of Software as a Service is the possibility to offer better services in a form totally scalable with the demand.

The first step is to evaluate the baseline performance of the application when run locally in the mobile device. Then, we identify the application dependencies among the executed processes identifying the parts of the code that can be executed in the cloud. Finally, we estimate the response time of the application in a cloud-based platform before of its deployment.

We will therefore implement and deploy applications according to different local versus remote mix to compare their execution performance. The different ways, modes or scenarios describe the place where the intensive calculus are realized. We define a first scenario where the mobile device has the capability to process images and to make recognition locally, called local mode. A second scenario where the recognition process is offloaded to a cloud computing infrastructure, called remote mode. And a third and last scenario where the recognition is performed in a distributed environment, mixing features of the two previous scenarios, called mix mode.

To know how the end-to-end response time can be affected, we emulate the communication channel in different conditions, to this end we used the Linux kernel tools of routing, filter and classification of packets to guarantee performances, bandwidths and lower latencies respectively by the utilities collection grouped in iproute2. Between the principal tools and utilities we used arpd, cstat, ifcfg, ip and tc. The last two utilities are known as LARTC of Linux Advanced Routing and Traffic Control [7] to manage the traffic traversing a network interface.

To add scenarios with delay and packet loss, we made use of NetEm that is a network emulator. This network emulator permits to convert the local area network in a slow and heavy network as can be an extended area network. NetEm is perfect to evaluate the behavior of protocols, applications and final systems, which have to be used on distributed environments. Originally NetEm behaves as a FIFO queue without delays or packet loss. To modify its discipline and its parameters, we can do it by the tc command in a Linux shell. To modify the parameters, we consider that the network delay is a variably value that depends of the amount of traffic that fluid by the same network. Generally, the delay is represented as a normal distribution with a medium value more/less the standard deviation value. The next parameter that we specify to modify the network behavior is the packet loss, here NetEm deletes randomly as packets as needed to fit to parameter.

V. EXPERIMENTS

As a basic tool for our evaluation, we developed a mobile client application for Android devices. A snapshot of this application is shown in Figure 2, where the green box indicates the face detected. In the bottom part of the screen, the characters indicate the name of the recognized person. And if appear a number next to the name, it indicates the age estimation of the person as an extra detail only available in the remote mode.



Fig. 2: Snapshot of the face recognition mobile application.

This application first captures pictures of people faces in local mode without using any communication with the external server, as if the cloud server was unavailable, the application makes the image processing necessary in the learning stage and the application is ready to receive any face image of people to make the recognition stage. In the case in which the network and the cloud server are available, the application automatically sent the data to remote processing. And the last function of the application is the visualization of the results as a front-end terminal.

The evaluation of the application has been performed on Samsung Galaxy Ace Smartphone running Android OS version 2.2 connected to internet by a WiFi and 3G networks. The testbed consists of execution of the application with the objective of getting the average execution time of each sub process with the same workload used in different scenarios, as shown in Figure 3. To get normalized values each test was repeated 10 times. The face recognition process is made with different values of people from 5 up to 20. And to each person different face images, from 1 up to 5, getting training sets with number of images multiples of 5.

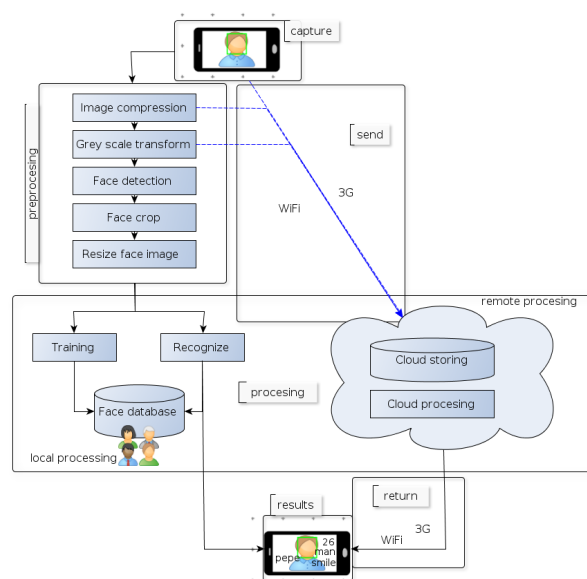


Fig. 3: Proposed system architecture.

Regarding network emulation, we first estimated the real values of latencies and packet lost experienced in a communication with a server in a cloud infrastructure. Then we used these values as guidelines to modify the network conditions when we make a POST request from the mobile device to servers in cloud infrastructure. The network conditions such as network delay, bandwidth rate, and packet loss were modified through the NetEm parameters. These tests was realized with virtual machines: we had a virtual machine executing a web server as a server on a cloud infrastructure. And the request were realized from another virtual machine, representing a mobile phone. Here, we varied the delay parameter using the values 2, 10, 50, 100 and 500 milliseconds and with each one of these values we used a different value for the packet loss parameter from 0% (emulating a perfect channel) passing for 1, 5, 10 until 20% (as a noisy channel). Each test was repeated for 10 occasions, too.

VI. RESULTS

In all scenarios it was necessary to process the captured image with an average cost in time of 480 ms. Then, the average times of each of the other pre-processes are shown in the table 1 and are graphically represented in the Figure 4. The processes that consume more time are: convert color images to grayscale (595 ms) and the detection of faces in an image (667 ms).

TABLE I: Time consumed for each sub-process to pre-processing images

Pre-process	Average Time in milliseconds
compression	180
gray scale	595
face detection	667
crop face	12
re-dimension	30
equalize	10

Now, please remember that processing is made of two stages: training and recognizing. In the training stage the overall system time is related directly to the number of images in the training set (while more images for each person more accurate are the recognition results). This creates a first trade-off between the required time to make the training and the

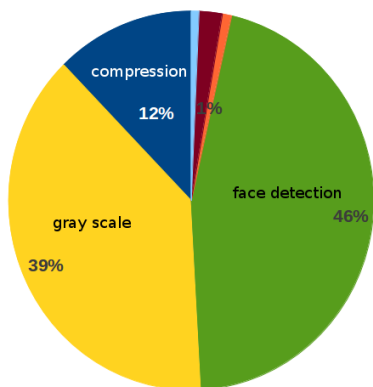


Fig. 4: Proportion of time consumed in pre-processing an image.

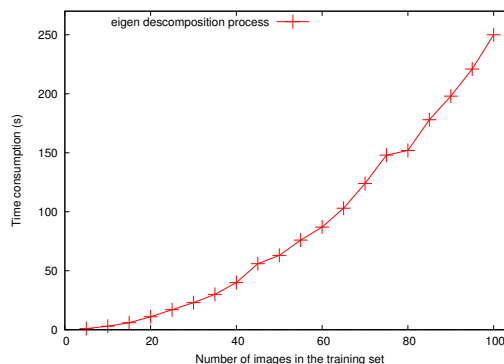


Fig. 5: Time required in the learning stage of local mode according to the number of images in the training set.

number of samples to train. As we can see in the Figure 5, a training set of 5 images needs 1,5 seconds to train while the case of a training set of 100 images needs 250 seconds. These values have an quadratic growth.

Considering the time required for the application to do the training stage can affect the user’s patience. A reasonable size of training set can be of 50 images that requires 63 seconds of waiting. The recognition stage need less time when compared with the training stage, to any person whose images have been previously trained, the facial recognition is realized in a range of 1,5 to 2,5 seconds according the number of images per person, as we see in Figure 6a. Here, we confirm that the fact of using Eigenfaces converts the recognition process in a quick process and permits to operate with wide sets of faces in very short times [11].

The recognition accuracy rate when we use only one image per person is unreliable because it doesn’t arrive to 50%. With 3 images per person is over 60%, but continues being unreliable. We reach close to 80% accuracy, when we use 5 images per person. As we can see in Figure 6b.

In remote processing, the images were sent in first place via a WiFi network and then via a 3G network. When we send images product of a strong pre-processing, the results are obtained in 939 ms with WiFi and in 3908 ms with 3G. When we send images with a lighter pre-processing we obtain the results in 2045 ms with WiFi and in 9790 ms with 3G. As we can see in Figure 7a.

In remote mode, the accuracy rate (Figure 7b) with images whose size is in the range of 8 kB to 102 kB is over the 80%. With images of 160 kB the accuracy is 91%. Namely better results with images without compression or in general without apply the pre-processing steps. But, if we avoid the pre-processing steps, the communication is affected by performance loss. This scale the problem size with higher latency and occupancy of bandwidth. To overcome these limitations, the scenarios with mix mode, we consider that the pre-processes of gray scale and face detection with 595 ms and 667 ms respectively are very expensive in terms of time consumption, then we decide not to use them. Simply the images captured

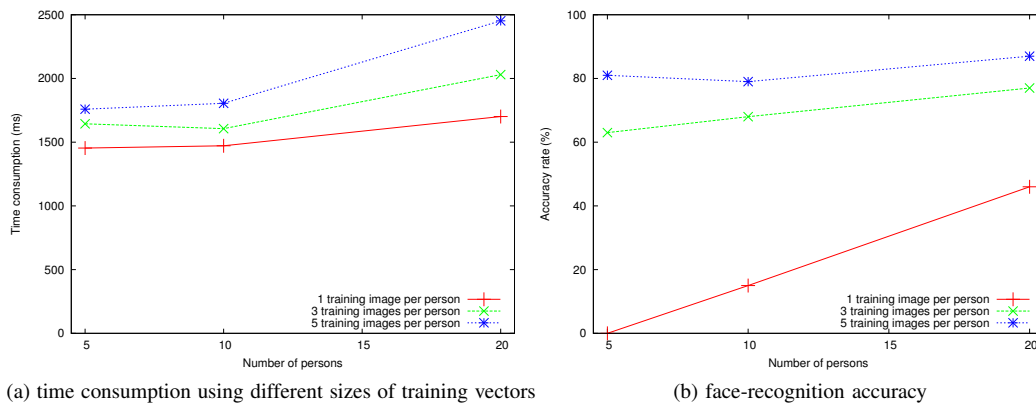


Fig. 6: Results obtained in local mode

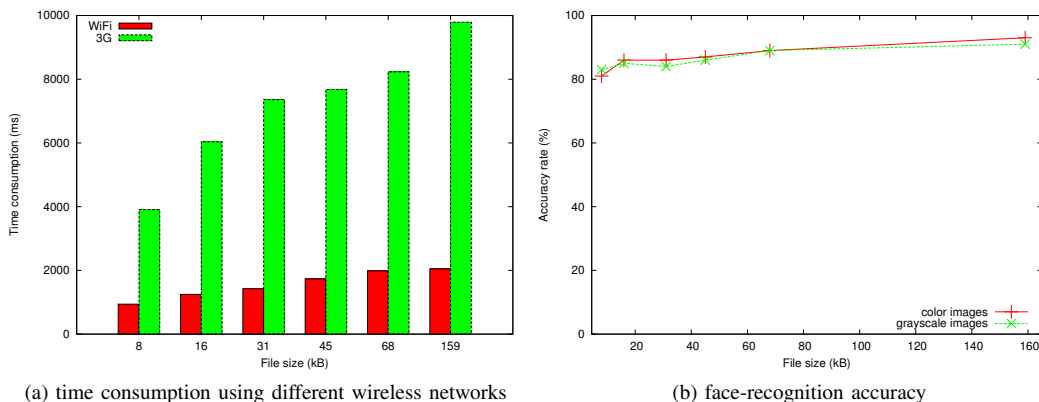


Fig. 7: Results obtained in remote mode

are compressed with JPG standard at 85%, to get a reduced version of these images with a size of near 16 kB. Then these images are sent to remote processing in the cloud infrastructure preferably using the WiFi network, to obtain results in 1712 ms with an 86% accuracy. It is the best combination and it can be considered as the optimum mode that we recommend to use.

If these preprocessed data are sent via WiFi network the recognition results are timeless that when are realized in local mode inclusive including the latency and round-trip delay time communicating with remote mode. The latter method requires network connectivity from the mobile device to Cloud environments. In cloud side due hardware potential and the complex of the algorithms, this scenario can provide more accurate results.

In the emulation of network conditions, with a 3G channel, the time required to make the request and get the response with an ideal channel (0% of packet loss) is 6 seconds with the minimum delay (2 ms), and with the maximum delay (500 ms) the response is obtained in 22 seconds. In Figure 8a, is displayed linear growth of the time necessary to receive a reply, under the differing amounts of packet loss for some link latencies.

Finally, we modify the latency values in WiFi channel from 2 ms up to 500 ms, the emulation deliver values from 800 ms to 1800 ms respectively, as we can seen in the Figure 8b.

VII. CONCLUSIONS

The growth of complex applications to mobile devices with support of cloud computing infrastructure demands better understanding of the effects of latency and packet loss. The communication client-server in wireless environments might suffer more latency and are more prone to packet loss. This communication is also affected by the Internet latency.

For this reason in this paper, we presented an application designed to allow the isolation of each process involved in a recognition of a face, integrated in a testbed that allowed the control of network conditions, such as latency and packet loss.

From the obtained results, we consider that offloading computation from mobile devices to cloud computing infrastructure can be done safely only if we have a guaranteed availability of a stable channel. In fact, with a broadband access of a WiFi network, we have low aggregate latencies close to 1 second. And if we use the 3G network, we have aggregate latencies near to 4 seconds. Both options with a packet loss level under

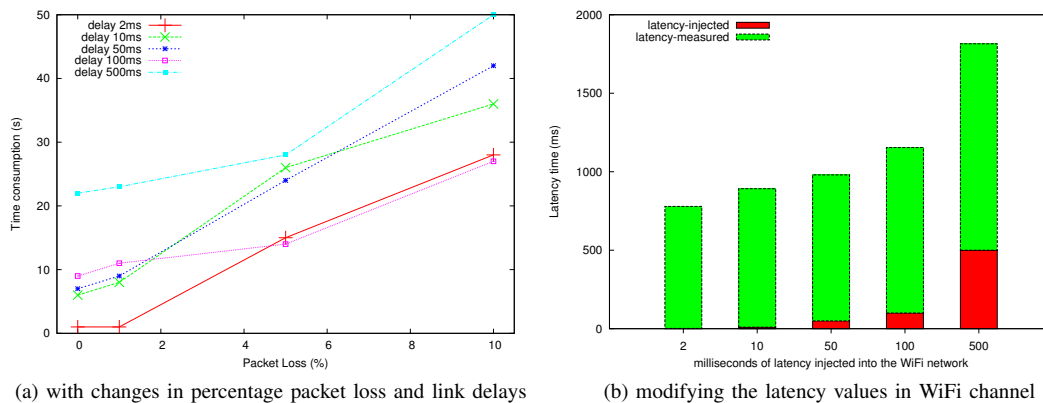


Fig. 8: The times obtained in the channel emulation using NetEm

3%. These are suitable for a good performance. When the packet loss level is over 3%, latencies are highly affected and this can be annoying to the user’s experience. For these cases is preferable to use the local calculus with order to keep low latencies.

With the continuous evolution of mobiles devices and the communication networks, it is possible to design, develop and use applications that combine the two operational modes in better efforts. For example, using these operational modes in applications where we will get the results in less of one second in autonomous mode or we will automatically use the remote mode sending queries to remote servers and get results in 2 seconds in normal cases or in 5 seconds in the worst case. With recognition training vectors previously charged to both options. If these results are not correct or are unreliable is possible to aggregate new registers manually, to future queries in a crowd sourcing style. As the people’s identity is a delicate theme. We can use this architectural proposal and the image processing in other aims, following with the visual content that can be found in an image, that requires recognition and identification.

ACKNOWLEDGMENTS

This work was partially supported by the *Ministerio de Ciencia e Innovación*, Spain, under Grant TIN2011-27543-C03-01.

REFERENCES

[1] D.-Y. Chen and J.-T. Tsai, “Resource-limited intelligent photo management on mobile platforms,” in *Machine Learning and Cybernetics (ICMLC)*, 2011 International Conference on, Jul 2011, pp. 627–630.

[2] P. Angin, B. Bhargava, and S. Helal, “A mobile-cloud collaborative traffic lights detector for blind navigation,” in *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, ser. MDM ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 396–401.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “Clonecloud: elastic execution between mobile device and cloud,” in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys ’11. New York, NY, USA: ACM, 2011, pp. 301–314.

[4] B.-G. Chun and P. Maniatis, “Augmented smartphone applications through clone cloud execution,” in *Proceedings of the 12th conference on Hot topics in operating systems*, ser. HotOS’09. Berkeley, CA, USA: USENIX Association, 2009, pp. 8–8.

[5] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “Maui: making smartphones last longer with code offload,” in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys ’10. New York, NY, USA: ACM, 2010, pp. 49–62.

[6] Google, “Google goggles,” URL: <http://www.google.com/mobile/goggles>, [retrieved: 03, 2013].

[7] T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. B. Schroeder, J. Spaans, and P. Larroy. *Linux Advanced Routing & Traffic Control HowTo*. URL: <http://www.lartc.org/>. [retrieved: 03, 2013].

[8] Y. Guo, L. Zhang, J. Kong, J. Sun, T. Feng, and X. Chen, “Jupiter: transparent augmentation of smartphone capabilities through cloud computing,” in *Workshop on Networking, Systems, and Applications on Mobile Handhelds*, ser. MobiHeld ’11. New York, NY, USA: ACM, 2011, pp. 2:1–2:6.

[9] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” in *Wireless Communications and Mobile Computing*. Wiley Online Library, 2011.

[10] K. Kumar and Y.-H. Lu, “Cloud computing for mobile users: Can offloading computation save energy?” in *Computer*, vol. 43. IEEE Computer Society, 2010, pp. 51–56.

[11] L. Lorente Giménez, “Representación de caras mediante eigenfaces,” in *Buran*, vol. núm. 11, 1998, pp. 13–20.

[12] E. Marinelli, “Hyrax: Cloud computing on mobile devices using mapreduce,” Master’s thesis, Carnegie Mellon University, 2009.

[13] J. S. Rellermeier, O. Riva, and G. Alonso, “Alfredo: an architecture for flexible interaction with electronic devices,” in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, ser. Middleware ’08. New York, NY, USA: Springer-Verlag New York, Inc., 2008, pp. 22–41.

[14] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, “The smartphone and the cloud: Power to the user,” in *MobiCloud*, vol. 28, October 2010.

[15] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” in *Pervasive Computing IEEE*, vol. 8, No. 4, 2009.

[16] Y. Taigman and L. Wolf, “Leveraging billions of faces to overcome performance barriers in unconstrained face recognition,” *CoRR*, vol. abs/1108.1122, 2011.

[17] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, “Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing,” in *Mobile Networks and Applications*, vol. 16, Jun. 2011, pp. 270–284.

[18] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face recognition: A literature survey,” in *ACM Computing Surveys (CSUR)*, vol. 35, Dec. 2003, pp. 399–458.

[19] OpenCV, “Opencv v2.4.3 documentation,” URL: <http://docs.opencv.org/modules/ml/doc/boosting.html>, [retrieved: 03, 2013].

Massively Scalable Platform for Data Farming Supporting Heterogeneous Infrastructure

Dariusz Król, Michał Wrzeszcz, Bartosz Kryza,
 Łukasz Dutka
 AGH University of Science and Technology
 Academic Computer Centre Cyfronet AGH
 Krakow, Poland
 {dkrol, wrzeszcz, bkryza, dutka}@agh.edu.pl

Jacek Kitowski
 AGH University of Science and Technology
 Department of Computer Science and Academic Computer
 Centre Cyfronet AGH
 Krakow, Poland
 kito@agh.edu.pl

Abstract — Data farming is a scientific methodology, which heavily depends on technical advances in high throughput computing to generate large amounts of data with computer simulation to investigate studied phenomena. Unfortunately, the availability of versatile data farming systems is very limited and none of existing tool enables integration with novel Cloud solutions. This paper presents a flexible platform for conducting large-scale data farming experiments on heterogenous computational infrastructure including: clusters, Grids and Clouds. Another important feature of the presented platform is the support of interactive data farming experiments, which includes an online analysis of partial experiment results and experiment extending capabilities.

Keywords - scalability; data farming; software platform; high throughput computing.

I. INTRODUCTION

In many disciplines of modern science, scientific discoveries are results of collecting and analyzing large amounts of data. In particular, an increasing popularity of conducting experiments, both physical and virtual, to understand studied phenomena leads to big data generation. A physical experiment often is too expensive to conduct it multiple times, e.g., when requires expensive equipment such as airplane engines or battleships, thus computer simulations are performed instead. Technological advances in recent years have led to significant improvements in the computer simulation field, e.g., reduction of the required time to run a computer simulation and refinement of simulation models in regard to its complexity. One can now simulate complicated phenomena in minutes or hours instead of days or months, with an improvement of results quality and simulation complexity.

Based on this technological progress, new forms of scientific methodologies have emerged, which are based on data-intensive computation and analysis. One such a methodology is called “The Fourth Paradigm” [1], in which new scientific findings are discovered by analyzing big amount of data coming from various scientific experiments. A complementary approach, which is gaining more and more popularity in recent years, is Data Farming [2], whose main objective is to develop a better understanding of landscape of possibilities as well as outliers that may be discovered

through simulation. This is especially important when concerning a decision-making process regarding complicated nature of scenarios involving security forces. The origin of the Data Farming methodology is in USA Marine Corps, where it was proposed to enhance military strategies. Though today, it is used in other disciplines of science [3-4]. The basic idea behind Data Farming is to grow significant amount of data by performing large number of simulations of a studied phenomena, each with a slightly different input values. Simulation results are described by a vector of parameters, called Measures of Effectiveness (MoE), which is used to evaluate each simulation. A result vector is treated as a single point of possible output landscape. After gathering a number of such points, a scientist can perform analysis of existing trends or anomalies, based on which, new insights into phenomena can be obtained.

A crucial requirement for conducting data farming experiments effectively is usage of high performance and throughput computer infrastructure. It is necessary to run a large number of simulations simultaneously and gathering output results. In addition, it is often required to integrate many heterogeneous computational infrastructures, when an experiment requires more computational power than a single computer centre can provide. Moreover, as new types of computational infrastructures are emerging, e.g., public Clouds, integration with existing infrastructures, e.g., Grid environments, becomes a major issue. Thus, a holistic platform, which will virtualize computational and storage resources, is required to conduct data farming experiment in an efficient way. In particular, it should automate all cumbersome technical aspects of infrastructure configuration and simulation running. Besides fulfilling functional requirements, such a platform should be scalable and adaptable to a changing state of knowledge about the studied phenomena, in order to be used in both small and large data farming experiments.

The rest of the paper is organized as follows. In Section II, we present existing tools, which can be utilized for conducting data farming experiments. Section III describes our platform, called Scalarm, its main design principles and objectives. Then, in Section IV, an experimental evaluation of the presented platform is depicted. We conclude this paper in Section V.

II. STATE OF THE ART

Although, Data Farming is becoming a popular scientific methodology lately, the software supporting this methodology is rather limited. One of the very few examples of such tools is OldMcData [5], which supports only two parts of a data farming experiment. It can prepare input a set of input vectors based on possible range of parameter values and selected design of experiment methods. Afterwards, it can schedule simulations to run on available computational resources with the Condor software [6], which can be configured to work with distributed resources and is able to move simulations' output from distributed computational resources to a designated location. However, no method for data analysis is provided, which means that external tools have to be used. Moreover, running simulations is a batch-like process, i.e., a whole package of inputs is submitted to a scheduler at once. The user can proceed to data analysis after the whole experiment is finished. There is no information about any partial results and the user cannot modify the parameter space of an experiment once submitted. Condor supports heterogeneous infrastructure integration, but it lacks the scaling feature in regards to application managers, which means the infrastructure delegated to perform the experiment has to be set before starting simulations and cannot be changed during the runtime.

Although, data farming oriented tools are rather limited in number, there are several tools, which can support different phases of the data farming process independently. One of the most important phases of the process is simulation execution with high throughput computational infrastructure. There are several tools available for this task as this is a generic problem in many computational disciplines. Distributed Infrastructure with Remote Agent Control (DIRAC) [7] is a platform supporting computations with heterogeneous resources including local clusters, Grids and Clouds. It was originally developed to provide a complete solution for using the distributed computing resources of the LHCb experiment at CERN for data production and analysis. DIRAC provides an additional abstraction layer between users and various compute resources to allow optimized, transparent and reliable usage. It exploits the concepts of Workload Management System with Pilot Jobs, which increase computations efficiency and reliability. DIRAC utilizes an agent-based architecture, where agents are deployed on the worker nodes, building a dynamic overlay network of readily available resources. These agents, being actually a representation of available computing resources, intend to reserve computational power to run actual tasks, which are distributed using a custom scheduling method. By using the Pilot jobs and Workload Management System concepts, DIRAC implements redundancy at the computational task level, i.e., DIRAC guarantees that tasks will be run, and in case of any failure it will be rescheduled. In addition, these concepts allow aggregating in a single system computing resources of a different nature, such as computational grids, clouds and clusters, transparently for the users. DIRAC provides the data management functionality, however it is

related to data distribution in a reliable manner among computational resources. It does not provide functionality required to analyse job results. Also, it does not have design of experiment methods built in for sampling input parameter value space, based on which computational jobs should be generated and scheduled. Thus, it can be only used as a part of a complete data farming platform, rather than being a complete solution for its own.

Falcon, which stands for a "Fast and Light-weight task execution framework", is a framework for rapid execution of many tasks on compute clusters [8]. Falcon focuses on efficient task dispatching, and delivers dispatching performance better than other systems, i.e., upto 440 tasks/sec. Furthermore, Falcon is highly scalable in terms of workers, which can be utilized to perform tasks, i.e., to over 54,000. Thus, applications end-to-end run time can be reduced in some cases up to 90% relative to versions that execute tasks via separate scheduler submissions. To achieve such performance and high scalability, Falcon utilizes a concept of multi-level scheduling to separate resource acquisition from task dispatch. Moreover, a streamlined dispatcher is used, which improves performance but eliminates support for features such as multiple queues, priorities, accounting, etc. Falcon consists of a dispatcher, a provisioner, and multiple executors. The dispatcher accepts tasks from clients and schedules subsequent tasks to next available executors. The provisioner is responsible for creating and destroying executors on available computational resources. Executors run tasks received from the dispatcher. Each new executor registers with the dispatcher. Components communicate via Web Services (WS) messages, except for notifications are performed via a custom TCP-based protocol. Although, Falcon provides high throughput and executors' scalability, it lacks dispatchers' scalability, i.e., performance of Falcon is constrained by capabilities of the server, which runs the dispatcher component. Moreover, whole Falcon functionality is limited only to dispatching, hence no functionality related to parameter space generation or results analysis is provided.

Since data farming is still a relatively uncharted territory none of existing tools provides functionality required for flexible running of various data farming experiments with different types of parameters and even simulation implementation technologies.

III. SCALARM PLATFORM

Due to lack of versatile software for conducting data farming experiments, we developed a new system from scratch, called Scalarm [9], which stands for Massively Scalable Platform for Data Farming. Scalarm intends to fulfill the following requirements:

- support all phases of a data farming experiment, starting from a design of experiment phase, through simulation execution and progress monitoring, to statistical analysis of results,

- support different sizes of experiments from dozens to millions of simulations through massive scalability,
- support for heterogenous computational infrastructure including private clusters, Grids and Clouds.

A. Provided functionality

Scalarm functionality focuses on conducting experiments, which follows the Data Farming methodology. In addition, Scalarm introduces an exploratory approach to experiment conducting. In a batch-like experiment execution, the user submits an experiment as a single package, waits for all simulations to compute, and then analyze obtained results. Based on the result analysis, new experiments are conducted to investigate interesting cases in more details. This loop can be repeated several times. On the other hand, the exploratory approach enables users to expand the parameter space of running experiments, based on an on-line analysis of already computed simulations, e.g., with regression trees and MoE histograms. Hence, the user can specify only small parameter space at first, and expand it on-line later on, which is a more natural way of conducting such experiments.

Supported use cases can be divided into three groups based on their expected results: experiment management, analysis and platform management. The first group includes activities related to preparation of new data farming experiments, their further monitoring and management, e.g., adding computational resources to execute simulations

included in a concrete experiment. The second group, i.e., analysis, contains all actions, which intend to visualise and discover knowledge from simulations' results in form of various charts and graphs. Hence, they can be utilized to discover meaningful insight into studied phenomena. The last group, i.e., platform management, includes use cases, which are important for a multi-tenant environment to operate, but they do not support the data farming process directly, e.g., login.

B. Architecture of the platform

Selecting an appropriate architecture style for virtual platforms, which intend to be deployed at a large scale, is the basic problem of modern software engineering. At a high-level of abstraction, Scalarm follows the “master-worker” design pattern, i.e., one part of the platform is responsible for scheduling the actual work to the other part of the platform.

Scalarm’s architecture utilizes a service-oriented approach with an additional modification, which addresses the scalability requirement. To cope with the requirement, we do not operate on the level of components and services, which represent single instances only. Instead, we extended the meaning of an application’s modularization unit to embrace the scalability feature. Thus, each Scalarm service can consist internally of a number of component’s instances, which provides exposed functionality, and a load balancer, which constitutes a single entry point to the service. An overview of the architecture is depicted in Fig. 1.

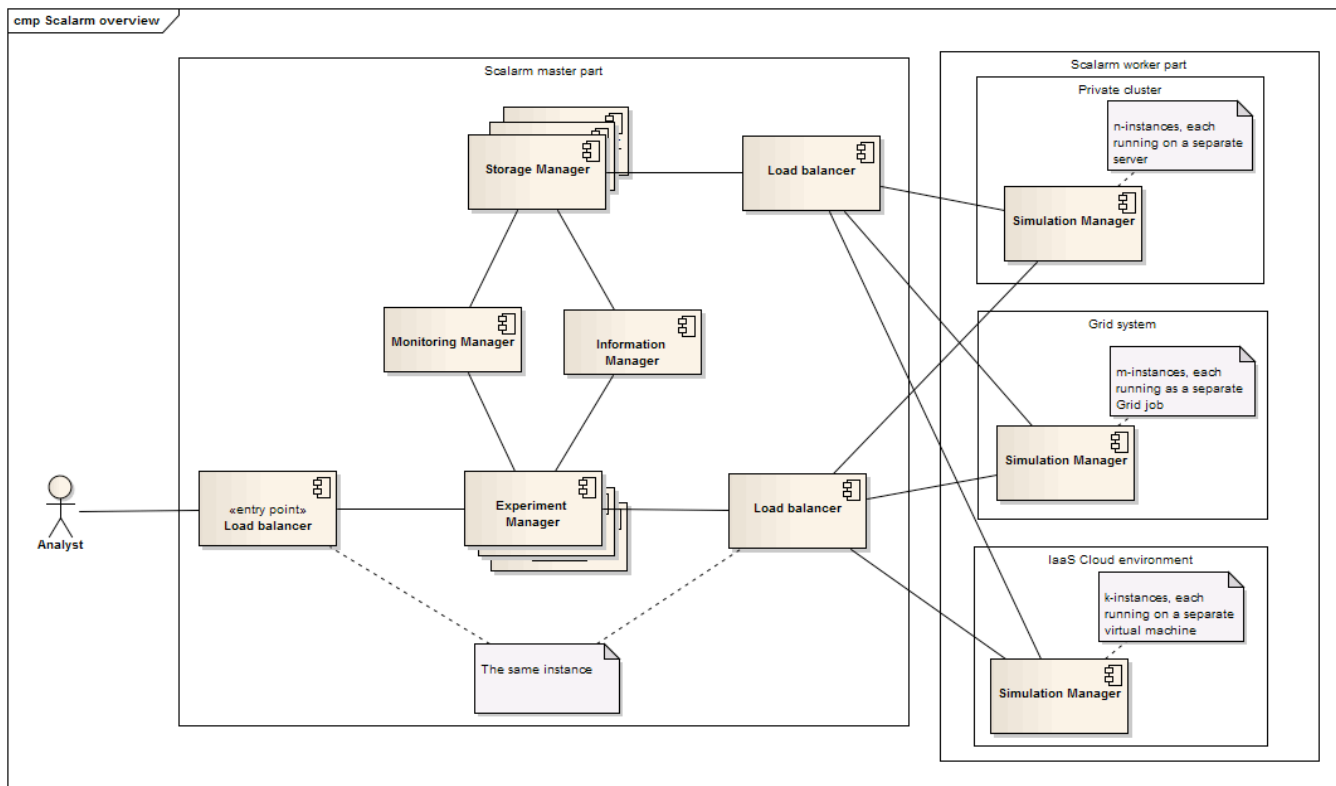


Figure 1. A component diagram of Scalarm.

The Scalarm platform includes the following components:

- Experiment Manager, which handles all interaction between the platform and actual users via a Graphical User Interface. On the one hand, it constitutes a gateway to the platform for analysts, i.e., provides a coherent view of information about all running and completed data farming experiments, and enables analysts to create new experiments or to conduct statistical analysis on existing ones. On the other hand, Experiment Manager is responsible for scheduling simulations to Simulation Managers.
- Storage Manager is an equivalent of the persistence layer concept but in a form of a separate service. Other components, mainly Experiment and Simulation Managers use this service to store different types of data: structural information about each executed simulation and experiment, and actual results of simulations, which may be either binary or text data. By utilizing a built-in load balancer, Storage Manager can be treated as a virtually centralized but physically distributed single point of data storage, which facilitates the client side while preserving performance and scalability.
- Simulation Manager is an intelligent wrapper for actual simulations, which can be deployed on various computational infrastructures, e.g., private cluster, Grids or Clouds. It can be treated as an implementation of the Pilot job concept, i.e., a special application that intends to acquire computational resources to run actual applications. However, while the Pilot job concept was created for Grid environments only, Simulation manager is infrastructure independent. The wrapper is responsible for preparing whole environment for a simulation, i.e., download necessary code dependencies and input parameter values. After a simulation is finished, Simulation Manager uploads results to the "master" part, i.e., log files and other binary outputs are sent to Storage Manager, while MoE values are sent to Experiment Manager along with information about simulation completion. As it can operate in a highly dynamic and unreliable environment, Simulation Manager supports fault tolerance for Experiment and Storage Managers failures as well as network connectivity issues. Moreover, to maximize resource utilization, Simulation Manager starts multiple simulations in parallel based on actual computational resource capabilities, i.e., additional simulations are started if it will not significantly decrease performance of already started simulations.
- Information Manager is an implementation of the Service locator pattern, known from SOA-based systems. It is a "well-known" place for

each component in the system, which stores information about other components' locations.

- Monitoring Manager constitutes a distributed monitoring system for the Scalarm platform. It contains two separated elements: sensors, which periodically sent monitoring data and a service, which stores this information. Sensors are built directly into each Experiment, Storage and Simulation Managers. It collects information about workload of Scalarm components, using operating system metrics, e.g., CPU and RAM memory utilization, as well as component specific metrics, e.g., response time of various requests.

C. Supported applications

Scalarm was originally evaluated in a multi-agent simulation area, with a goal of supporting a training process of security forces. A sample simulation scenario involved controlling the access of civilians to a military base camp during elections in a mission abroad. In this scenario, civilians were waiting in front of a camp entrance to an operation base with an intention to start a skirmish. From the security forces point of view, the goal of this scenario was to prevent the escalation of aggression by effective negotiations. However, civilians may act differently, depending on input parameter values, hence actions performed by security forces should be adjusted to a concrete behaviour. A goal of a data farming experiment, which used this simulation scenario, was to find out how to minimize the number of injured civilians in such a scenario, regardless their behaviour.

Scalarm facilitated the experiment at the following phases:

- A design of experiment phase, whose result is a specification of the input parameter space. Scalarm provides a set of views, where an analyst specifies types of parametrization for each input parameter and design of experiment methods, which should be used.
- Simulation execution on heterogeneous computational infrastructure. Scalarm supports different types of computational infrastructures, i.e., common computational clusters available via SSH, Grid environments accessible via the gLite middleware [10], and public clouds supporting Amazon EC2 API.
- Statistical analysis of results. Scalarm provides a set of built-in graphs, which can be created based on completed simulation results: histograms, regression trees and bivariate graphs.

For more details about conducted data farming experiments regarding security forces, please refer to [11]. Though, Scalarm was evaluated with a particular type of simulations, it can be used in any other science discipline, where the Data Farming methodology can be utilized, e.g., materials science or life-science.

IV. EXPERIMENTAL EVALUATION

To evaluate Scalarm, we conducted both functional tests of supporting different computational infrastructures and performance tests to measure the platform's scalability.

Functional tests concerned simulating a scenario described in Section III. Two standard HP ProLiant worker nodes (described in a following section) were used to run one instance of Experiment and Storage Managers. To run Simulation Managers, we used:

- 9 worker nodes from a private cluster,
- 50 Grid jobs scheduled to a Grid infrastructure,
- 50 High-CPU Extra Large instances from Amazon EC2.

In the experiment design phase, 14 from 92 of simulation input parameters (describing initial emotional state and other attributes of simulated entities) were set to the "Range" parametrization with 2^k method applied, which generated 16 386 different cases to simulate. The utilized set of resources enabled us to execute more than 620 simulations simultaneously with more than 140 simulations complete in each minute.

An output of each simulation included: a text file with less than 7 MB of simulation logs, and 44 different MoEs describing aggregated emotional states of different entity groups and statistic regarding the simulated scenario. Compressed logs were sent to Storage Managers, which had a disk array connected with 6 TB of total capacity.

After several minutes of computations, an analysis of gathered results was conducted using histograms and regression trees. Based on this analysis, the experiment was extended with additional parameter values. A whole test was recorded and can be found online at [12].

The second set of tests concerned the platform's scalability. We intended to evaluate scalability of the master part, which includes Experiment and Storage Managers, since running many independent workers is trivial. We used production infrastructure, however an empty simulation was actually performed to minimize the number of Simulation Managers required to saturate platform's throughput, which was measured with completed simulations registered by Experiment Managers in a period of time.

A. Testing scenarios

Our testing scenarios focused on evaluating how Scalarm handles experiments of various sizes with different amount of computational resources. The main measured parameter was the total execution time of each experiment. Scalarm has three main components, namely Experiment, Storage and Simulation Managers, which can be scaled. In presented tests, the number of Simulation Managers was experimentally selected to saturate platform's throughput. Hence, only numbers of Experiment and Storage Managers were used as parameters of performed tests.

Regarding experiment sizes, i.e., the number of simulations within an experiment, we used the following set of values to present full capabilities of the platform: 100 000, 200 000, 500 000, 1 000 000, 2 000 000, 5 000 000.

Concerning computational resources, the parameter depicted the number of servers dedicated to run Experiment and Storage Managers. Our tests included the following values of this parameter: 1, 2, 4 and 8. However, each component run on a separate set of servers, which means that in each test, the total number of servers was doubled.

B. Testing environment

In case of performance tests, we used a computing cluster to run Experiment and Storage Managers to minimize the network latency. Simulation Managers were scheduled to a part of PL-Grid infrastructure located within the same site.

To run each component, we used standard HP ProLiant worker nodes, connected with each other through a 10 GbE network switch, while connection between a worker node and switch was 1 GbE link. Each worker node has the following parameters:

- 2x Intel Xeon CPU L5420 @ 2.50GHz
- 16 GB RAM
- 120 GB hard drive (5400 RPM)

C. Evaluation results

Aggregated test results are depicted in Fig. 2. Each line on the chart denotes a separate configuration of Scalarm used in tests, i.e., numbers of servers running Experiment and Storage Managers represented as a pair (<experiment managers count>, <storage managers count>). For each configuration, we measured total execution time in seconds for experiments of different sizes.

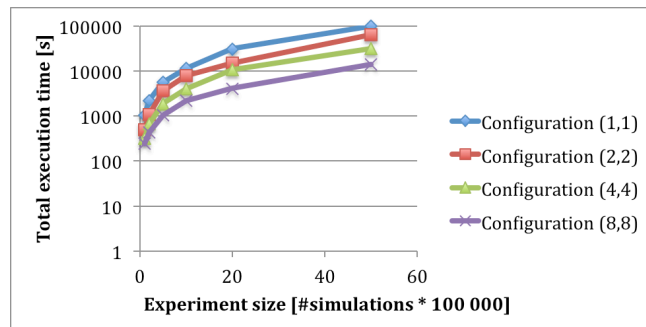


Figure 2. Experiments' execution time for different Scalarm configurations.

There are a few things worth noticing. First of all, the more resources Scalarm has, the better performance it provides. The performance gain varied depending on actual experiment size. Let's compare configurations (1,1) and (2,2). Execution time decreases by 53% for experiment size 100 000, but only by, 31% for experiment size 1 000 000.

The second notice concerns the execution time of experiments with an increasing size using the same configuration. Regardless the configuration, the execution time of subsequent experiments with an increasing number of simulations rises more than linearly. It is caused by an increasing effort of simulation information management. Each simulation is represented in Scalarm by a row in a non-relational database. Performance of such databases depends on the IO subsystem, especially when concerning millions of rows. Hence, after exceeding some thresholds of a database

size (about a million of rows on a single server), database operations tend to take more than expected.

Based on obtained results, we calculated speedup (1) and efficiency (2) metric using classical formulas.

$$\text{Speedup}(N) = T(1) / T(N) \quad (1)$$

$$\text{Efficiency}(N) = \text{Speedup}(N) / N \quad (2)$$

Efficiency of Scalarm (depicted in Fig. 3) is greater than 0.7 in most cases, which is a good result, especially when concerning a wide range of tested configurations. Furthermore, for some experiment sizes and the configuration consisted of 2 servers for Experiment and Storage Managers respectively, efficiency is greater than 1, which could be have been caused by data sharding between instances of a database on seperated servers, which enabled having all data in memory instead of using local disk.

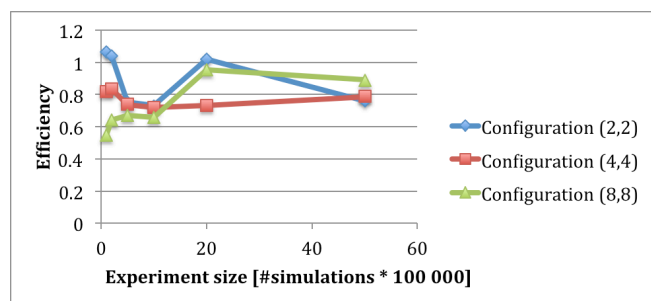


Figure 3. Scalarm efficiency for configurations including more than 1 server per component.

An average throughput for the Configuration (1, 1) was about 4776 simulations per minute. We estimated the number of Simulation Managers, which would be required to saturate Scalarm when running actual simulations by comparing to the throughput of running actual simulations and the throughput with an empty simulation. In the case of our simulation, we should have more than 120 000 of Simulation Managers running simultaneously. This was the main reason why the scalability evaluation was performed with an empty simulation.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a versatile system for running large scale data farming experiments involving processing of a parameter space where custom design of experiment methods and interactive fine tuning of processed parameter space are required. The system is currently being evaluated for military mission planning support in order to improve behavior models for agent-based simulation component and to allow drawing conclusions regarding selected Measures of Effectiveness for higher echelons.

The future work will include application of the platform in a metallurgy scenario [13], with focus on distributed semantic-based Virtual Organization collaborations [14].

ACKNOWLEDGMENT

The authors are very grateful to Łukasz Flis, Marek Magryś and Patryk Lasoń from Cyfronet for help in preparing the testing environment based on the PL-Grid infrastructure. The research is partially supported by the POIG.02.03.00-00-096/10 “PL-Grid PLUS” project. D. Król thanks to the National Science Centre grant no. 2012/05/N/ST6/03461 for support. AGH-UST grant no. 11.11.120.865 is also acknowledged.

REFERENCES

- [1] T. Hey, S. Tansley, and K. Tolle, “The Fourth Paradigm: Data-Intensive Scientific Discovery”, Eds., Redmond, VA: Microsoft Research, 2009, ISBN 978-0-9825442-0-4.
- [2] A. Brandstein and G. Horne, “Data Farming: A Meta-Technique for Research in the 21st Century”, in *Maneuver Warfare Science 1998*, Marine Corps Marine Corps Combat Development Command Publication, Quantico, Virginia, 1998.
- [3] D. Moses, “Data farming helps hospital keep nurses at bedside”, *HealthCareITNews*, [online: <http://www.healthcareitnews.com/news/data-farming-helps-hospital-keep-nurses-bedside> as of January 14, 2013]
- [4] T. Beach, et al., “Application of Design of Experiments & Data Farming Techniques for Planning Tests in a Joint Mission Environment”, *International Data Farming Workshop 15*, November 2007.
- [5] S. Upton, “Users Guide: OldMcData, the Data Farmer”, Version 1.1, United States Marine Corps Project Albert. Quantico, Virginia, 2010.
- [6] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, “Condor: a distributed job scheduler”, *Beowulf Cluster Computing with Windows*, MIT Press Cambridge, MA, USA, 2002, pp. 307 – 350, ISBN:0-262-69275-9.
- [7] J. Saborido, F. Gomez-Folgar, J. L. Cacheiro, and R. G. Diaz, “DIRAC Integration with Cloud Stack”, *Proc. IEEE Third International Conference on Cloud Computing Technology and Science*, 2011, pp. 537-541. ISBN: 978-0-7695-4622-3.
- [8] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, “Falkon: a Fast and Light-weight task executiON framework”, *Proc. ACM/IEEE conference on Supercomputing*, 2007, pp. 1-12, ISBN: 978-1-59593-764-3.
- [9] B. Kryza, D. Król, M. Wrzeszcz, Ł. Dutka, and J. Kitowski, „Interactive Cloud Data Farming Environment for Military Mission Planning Support”, *Computer Science Journal*, vol 13(4), 2012, pp. 89-100.
- [10] gLite – Lightweight Middleware for Grid Computing website, [online: <http://glite.cern.ch/> as of January 14, 2013]
- [11] S. Dlugolinsky, et al., “Using parallelization for simulation of human behaviour”. *7th International Workshop on Grid Computing for Complex Problems*, Bratislava, 2011, pp. 258-265, ISBN 978-80-970145-5-1.
- [12] Scalarm website – overview section, [online: <http://www.scalarm.com/overview.html> as of January 14, 2013]
- [13] J. Kitowski and B. Kryza, “Dynamic virtual organization management framework supporting distributed industrial collaboration”, *Computer Methods in Materials Science*, vol. 11(4), 2011, pp. 514-523.
- [14] A. Mylka, A. Mylka, B. Kryza, and J. Kitowski, “Integration of Heterogenous Data Sources in an Ontological Knowledge Base”, *Computing and Informatics*, vol. 31(1), 2012, pp. 189–223.

Fuzzy Controlled QoS for Scalable Cloud Computing Services

Stefan Frey, Claudia Lüthje, Vitali Huwwa, Christoph Reich

Furtwangen University

Cloud Research Lab

Furtwangen, Germany

{stefan.frey,claudia.luethje,vitali.huwwa,christoph.reich}@hs-furtwangen.de

Abstract—An important characteristic of cloud infrastructures is scalability on demand. A scalability service monitors performance load metrics and decides to scale up or down, by provision or revoke of cloud resources. This could guarantee Quality of Service (QoS) and enforce Service Level Objectives (SLOs). The approach of this paper shows that with additional imprecise information (e.g. expected daytime performance) the up and down scale mechanism of such an infrastructure can be improved and SLA violation can be avoided.

Keywords—Cloud Computing; Scaling Service; Fuzzy Logic; SLA; QoS

I. INTRODUCTION

Cloud computing offers customers resources on demand on a self-service basis and gives them access to a large pool of computational power and storage. Customers do not have to manage and maintain their own IT assets and get charged by cloud providers based upon the amount of resources used or reserved. The fly in the ointment is the minimal guarantees of Quality of Service (QoS) for the user's applications. It is common that big cloud providers like Amazon offer only rudimentary service guarantees, like for example a guarantee for 99,95% availability of their EC2 cloud service. In most cases providers do not give any performance guarantees at all. Cloud computing services, like the auto scaling service of Amazon [1], scale the capacity of virtual machines (VM) up or down automatically according to e.g. CPU utilization. Such a service controls the number of VMs to maintain the performance of a service that experiences hourly, daily, or weekly variability in usage. The architecture of such a setup can be seen in Figure 1 inside the blue dashed box. This obviously has the potential to guarantee Key Performance Indicators (KPIs) indirectly but KPIs such as e.g. request response time which are typical Service Level Objectives (SLOs) in a Service Level Agreement (SLA) are not controlled directly.

Therefore, SLA violations can happen especially due to peak demands, caused by all kind of reasons (e.g. product launches, political statements, service advertisement, weather changes, etc.), and the up scaling delay of the infrastructure (e.g. VM start time, LB reconfiguration, infrastructure limits, and economical limits to prevent extraordinary costs, etc.). Other reasons for not matching the SLA guarantees could be limitations, like the maximum number of VMs or non-ideal load balancing algorithms, which are not considered in the approach of this paper. Decent scaling is very important,

because the if scale down happens to early SLA violations occur and if its set to late the customer will pay for resources that are not utilized.

To minimize the number of SLA violations and to guarantee the QoS, an behaviour, load and performance prediction model is needed. If one could predict the usage of an service, looking ahead further than the infrastructure delay time, one could guarantee the QoS for that specific service.

The rest of the paper is organized as follows. In Section II the related research efforts are discussed. Then a detailed description of the problem of QoS in cloud computing can be found in Section III. In Section IV, the specific approach using fuzzy logic for controlling the scalable cloud service is introduced. The proof of concept is reported in Section V. Finally a conclusion is drawn and future work is suggested in Section VI.

II. RELATED WORK

Since 2009 many teams are working on the problem to improve the QoS for cloud computing. Armstrong and Djemame tried to transfer the technologies of QoS from grid computing to cloud computing as discussed in the paper "Towards quality of service in the cloud" [2]. The paper of Rochwerger et al. [3] discuss the funded project RESERVOIR, in which pooled resources handle peaks and slopes of resources.

Another interesting approach is the Q-Clouds framework described by Nathuji et al. [4]. This framework for the management of cloud servers enables the possibility to apply and control QoS. The introduced Q-states provide the possibility for users to define certain metric limits of SLOs, based on a cost model. The more the customer is ready to pay, the less likely is a SLA violation. The controller component uses a MIMO (multi-input, multi output) model for the calculation VM resources. So an input vector is defined by the platform controller. Based on that the output vector delivers the predicted QoS values. Unlike to the approach of this paper they basically use infrastructure metrics (e.g. performance, memory, etc.) to control the QoS.

The important next steps in the QoS for cloud computing were developed by Ferretti/Ghini/Panerieri [5]. Their paper presents an architecture, which provides cloud resources dynamically. The developed middleware tries to avoid SLA violations with the same use case as presented in this paper. Therefore they split the problem into three components: The

host platform is an dynamic configuration, that should guarantee the requirement of the SLA. The monitoring component is checking the host platform and its application to display changes in the configuration and violation of the QoS requirements. The third component is responsible for the dispatching and the load balancing. This component tries to keep the required QoS. It is implemented in an intelligent way, to distribute the available resources. It is distinct to load balancing for requests and sessions. Different to this paper is that the load balancer takes over the monitoring of the SLAs instead of a separate module. Further the only metric that is used is the actual request response time. Compared to our approach, there can be considered mean value, derivation value, imprecise information and admin control information.

Many recent works deal with computing resource allocation in clouds focusing specific management objectives, such as energy efficiency [6], fairness [7], economic fitness [8], and service differentiation [9]. All of the above works however deliver placement solutions. They do not consider the problem of controlling a load balanced, scalable Cloud service.

Related research can be found in the area of forecasting the load of electrical power in [10] and [11] where they use social, economic, and weather condition factors. To achieve QoS guarantees this paper uses such additional factors as well.

III. QoS IN CLOUD COMPUTING

A Service Level Agreement (SLA) is a contract between customer and provider, that specifies service performance properties. These properties are called Service Level Objectives (SLOs), which contain metrics called Key Performance Indicators (KPIs) and the specific value to be guaranteed. These performance metrics should be guaranteed over a relatively long time interval and if a metric is violated commonly penalty costs may have to be paid to the customer by the provider.

Compared to traditional data centers it is easier to guarantee QoS in cloud computing data centers, because of the possibility to automate infrastructure administration and added value services such as auto scaling. Today's virtualization technologies allow dynamic provisioning of virtual machines (VM), networks, storage, etc. Therefore a completely automatic, adaptable customer infrastructure is on the horizon to react in real time to load changes.

Especially a scalable infrastructure can easily be provisioned in the cloud service model Infrastructure as a Service (IaaS), that allows automatic up/down scaling according the actual load. This is a big step towards the possibility to guarantee KPIs like the service request response time, that is a widely used Key Performance Indicator and a common Service Level Objective (SLO).

The approach presented in this paper improves the up and down scaling by using additional information, like the expected load at a certain daytime in the future, expected increase at a specific future day because of special events, etc.. With the additional information we try to forecast the load and therefore allow a better pre-acting up or down scale of the infrastructure if needed.

IV. FUZZY CONTROL TO IMPROVE QoS CONTROL

Most approaches consider infrastructure sensor data like bandwidth, request/response time, CPU usage, memory usage, etc. to control the scaling infrastructure as seen in Fig. 1 dashed box. The approach of this paper is to use additional, often imprecise information (e.g. weather) to improve the management to meet QoS requirements stated in SLAs. These imprecise factors (e.g. user wants scaling aggressive/moderate, etc.), political factors (legal changes, political summits, etc.), economic/market factors (product advertising, product launch, etc.), other factors influencing the service usage (e.g. weather, gossip, etc.) can not be modelled precisely.

Fuzzy logic allows to model imprecise information by the user (service administrator) in the form of non-numeric linguistic variables (e.g. age: young/old). These fuzzy inputs are used in the fuzzy control system, that uses expert knowledge to inference a fuzzy output. After defuzzifying this output to a crisp value, then this controls the overall scale system how big the up and down scale factor should be. For example, if a customer wants to have an aggressive scaling control the infrastructure will scaled up with e.g. 3 VMs otherwise with only one VM at a time. The scaling domain expertise is modelled in a knowledge base with fuzzy IF-THEN rules.

In the next Subsection IV-A the architecture of the fuzzy scaling cloud service is described, followed by subsection IV-B, discussing monitoring parameters, which will show the wide variety of information to improve the cloud scaling service. The last subsection IV-C presents the fuzzy control module.

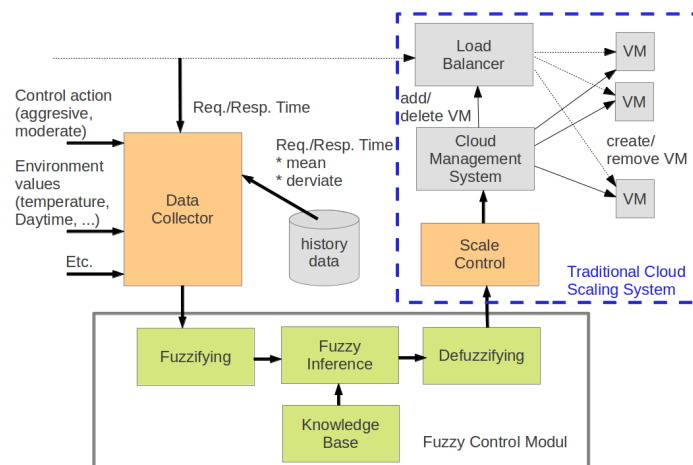


Fig. 1. Fuzzy Controlled Scaling Architecture

A. Fuzzy Controlled Scaling Architecture

Figure 1 shows the architecture for a load balanced service by automatically scaling up/down the infrastructure by starting/stopping VMs. It consists of two new modules compared to the traditional scaling infrastructure (blue box), the *Data Collector* and the *Fuzzy Control Module*.

The *Data Collector* collects all information data, crisp (e.g. cpu usage) and imprecise data (e.g. weather). The data is

categorized in infrastructure data (e.g. req./resp. time), history data (e.g. req./resp. time 5 minutes ago), control action (e.g. aggressive up/down scale), environment data (e.g. daytime), and other information that might influence the load of the service.

All collected data is input data to the *Fuzzy Control Module* where the data is fuzzified, results propagated by the fuzzy inference engine and quantified by defuzzification. The defuzzified value (Number of VM to be started or stopped) is put into the *Scale Control* module. This module generates XML-RPC calls to the *Cloud Management System*.

B. Information Factors for Control

The relevant information to improve the up/down scaling can be categorized into monitoring data: infrastructure, historic infrastructure, time-dependent, and service-dependent sensor data described in the following paragraphs in more detail.

a) Infrastructure Sensor Data: Table I lists factors that can mostly be monitored using sensors placed in various locations in the cloud infrastructure. KPIs, like request response time can easily be measured at the load balancer (LB). Cloud specific parameters, like start time of VMs, can be aquisitioned at the cloud management system. If user service request types should be categorized (typically a imprecise parameter), it is best to ask the user admin of the cloud resource.

TABLE I. INFRASTRUCTURE SENSOR PARAMETER

Parameter	Example	Cloud Source
KPI	req./resp	load balancer
cloud specific indicators	VM start time, bandwidth	cloud management system
request type	long running req.	user
...

The quality of the cloud infrastructure or service implementation can be taken into account as well. The load balancing control might be influenced by the basic robustness of the overall infrastructure. The infrastructure robustness can be modelled by an imprecise parameter e.g. strong, weak.

b) History Infrastructure Sensor Data: Table II lists parameter that have been previously collected in a history data base. The purpose is to calculate values like, mean values, derivation values, etc. These statistical data can be good indicators to improve the LB management.

TABLE II. HISTORY INFRASTRUCTURE SENSOR PARAMETER

Parameter	Example	Source
derivation KPI	derivation req./resp	history DB
mean value KPI	req./resp. mean value	history DB
...

Imprecise history parameters can be of interest as well. Suppose a service depends on the weather condition (e.g. online shop for winter tires), then a sudden change of the weather condition from try to snowy condition makes it more likely, that the load of such a service is higher.

TABLE III. TIME-DEPENDENT SENSOR PARAMETER

Parameter	Example	Source
daytime	end of work	user input
weekday	Saturday	calendar
holiday	Christmas	country holiday cal.
product events	new iPhone	user input
...

c) Time-Dependent Sensor Data: Table III lists parameter that can influence the infrastructure management at a pre-defined time.

The knowledge of the typical weekly usage for an service (see Fig. 2) can be modelled and therefore the decision to scale up or down strongly or weekly depending whether the change is high or not.

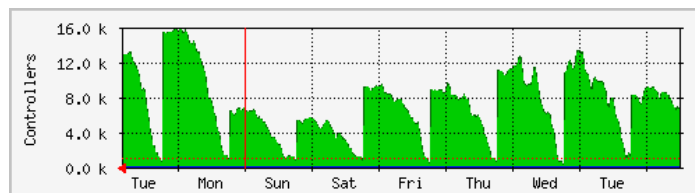


Fig. 2. Example: Weekly Load of the HFU Learning Management Platform

d) Service-Dependent Sensor Data: Table IV lists parameter that influence the control infrastructure depending on the related service. Political parameters, like new legal issues enforcing more logging at the service side. Market events, like product launches, marketing events, new prices, etc. can influence the usage of services. Gossip, modelled as good news or bad news is influencing service usages. Importance of service might need a more aggressive management to make sure, that the SLA violations can be minimized.

TABLE IV. SERVICE-DEPENDENT SENSOR PARAMETER

Parameter	Example	Source
politic	EU summit	news ticker
market price	Facebook share	exchange feed
gossip	new Facebook mobile	news ticker
service importance	control behaviour (moderate/aggressive)	user
...

C. Fuzzy Control Module

The *Fuzzy Control Module* consists of four main fuzzy control processes represented by the four sub-modules respectively (see Fig. 1). The crisp and imprecise input data is converted into fuzzy values for each input fuzzy set with the *Fuzzifying* module. The decision making logic of the *Fuzzy Inference* module determines how the fuzzy logic operations are performed (SUP-MIN inference), and together with the *Knowledge Base* module determine the outputs of each fuzzy IF-THEN rules. Those are combined and converted to crisp values with the *Defuzzification* module. The output crisp value can be calculated by the center of gravity or the weighted average and converted to the number of VM to started or stopped.

It follows a closer look at the 3 processes fuzzification, fuzzy inference and defuzzification.

Fuzzification: Fuzzification is the process of decomposing the input data into fuzzy sets, with trapezoidal shaped membership functions. Figure 4 shows a system of fuzzy sets for an input with trapezoidal membership functions. Any particular input is interpreted from this fuzzy set and a degree of membership is interpreted. If the request-response-time, for example, it set to about 100 request-per-seconds, the fuzzy value *loaddeviation* is set to *low*.

Fuzzy Inference: The fuzzy values gathered from the input data are processed by the inference engine using the expert domain knowledge modelled as fuzzy IF-THEN rules. The following fuzzy rules are examples how to state the domain knowledge in the area of up and down scale control.

```
IF ReqRespTime_rising=high AND
   expected_ReqRespTime_rising=high AND
   product_launch=now AND
   ....
THEN
   up_scale=very high
   ...
```

Defuzzification: After the fuzzy reasoning the resulting linguistic output variable (e.g. scale up = high) needs to be translated into a crisp value (e.g. number of VMs to be started or stopped at time). Defuzzification maps the output from the fuzzy domain back into the crisp domain. The most common defuzzification methods is the Center-of-Area (C-o-A) often referred to as Center-of-Gravity used in this approach and is defined as follows:

$$x^* = \frac{\int \mu_i(x)x dx}{\int \mu_i(x) dx} \tag{1}$$

where x^* is the defuzzified output, $\mu_i(x)$ is the aggregated membership function and x is the output variable. The C-o-A method calculates the area under the scaled membership functions and within the range of the output variable and afterwards calculates the geometric center of this area.

V. PROOF OF CONCEPT BY SIMULATION

In this section we discuss and evaluate the simulation results. The objective of the assessment was to verify whether or not our approach will ensure QoS for a cloud service better than conventional procedures. Hereafter we give a short introduction in our *Simulation Environment* (see section V-A) followed by the main features of our *Simulation Scenarios* (see section V-B), thereafter we discuss the results that we have obtained during several tests.

A. Simulation Environment

For feasibility testing, we created an simulation environment to be capable of validating the general fuzzy controlled scaling architecture proposed in this paper. The simulator therefore consists of four major components. Firstly, a request generator module, which simulates the generation of requests from an application to the cloud service. Here should be stated, that in our simulation requests are generated with an static workload. The second module is the load balancer which receives the generated requests of the request generator and distributes them to the pooled virtual machines. Here,

the time is measured from the generation of the request till the execution inside a virtual machine is completed. This request response time then is checked by the scaler module, which decides either based on the fuzzy or the conventional rules whether to scale the service up, down or wait. The conventional rule set is an simple boundary system, where when the measured average request response hits the upper boundary a virtual machine gets started or when the lower boundary is hit a virtual machine is stopped. In between both boundaries the scaler waits.

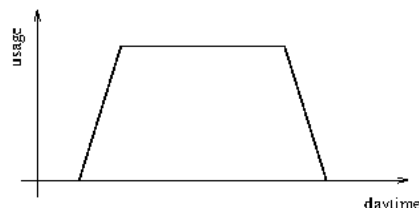


Fig. 3. Expected Load During Daytime

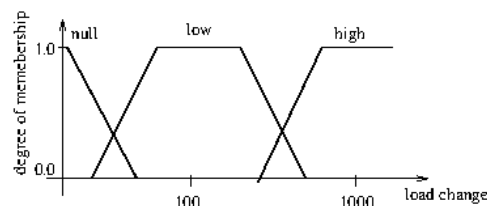


Fig. 4. Input Fuzzy Set for Load Deviation

The fuzzy set uses the same boundaries, but as an additional decision factor, a prediction based on expert knowledge is used.

Figure 3 shows the simplified load of an service during daytime. Based on such knowledge an expert specifies whether the load will be increasing at an *high*, *regular* or *low* rate. In case of an *high* prediction the fuzzy scaler generally scales up faster, which means it starts virtual machines on a lower load and additionally starts up to two virtual machines based on the load. Additionally it will scale down later, keeping a higher pool of available virtual machines. The *regular* prediction equals the conventional rule set, therefore resulting in essentially the same behavior. A *low* prediction, is in principle a reversed *high* prediction, which will change the behavior into generally scale up later and scale down faster. And similar to the *high* it is allowed to stop up to two virtual machines at once.

The simulator is based on a model in which a generated request includes a static processing time of 100ms. The KPI, is measured as the request/response time, based on the average of the last 10 processed requests. Thereby the time is counted from the generation of the request, till arrival of the response after the processing at the load balancer. The QoS limit has been set to 2000ms in this model and the conventional rule set regulates at an average response time of 1500ms by upscaling and at 1000ms by downscaling one virtual machine at a time. To eliminate the influences of the test environment, like processor fluctuations the factor of 10 was used to all above described values.

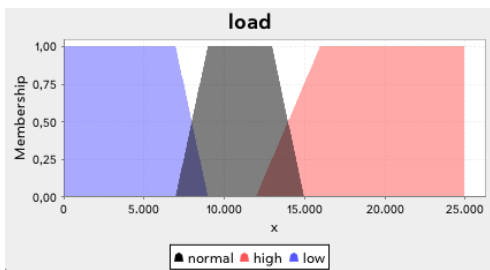


Fig. 5. Responsetime Fuzzy Sets

Figure 5 above shows the corresponding fuzzy set, where a load of below 9,000ms is considered *low* and above 15,000ms is *high*. In between stretches the range of *normal* load. These fuzzy values are combined with the fuzzy prediction values to create the set of fuzzy rules. To determine the suitability of the procedure presented, different scenarios have been created and tested with and without the fuzzy control mechanism. Following the scenario with the specific pre-conditions and characteristics is described and the obtained results are presented.

B. Simulation Scenarios

In the scenario the number of generated requests are increased rapidly and kept on a high level for an minute then to rapidly fall again. Figure 6 below shows the generated load graph for this scenario and the settings are shown in Table V.

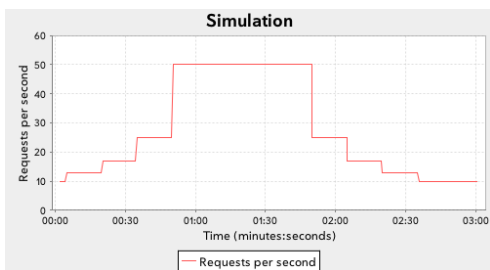


Fig. 6. Generated Load Scenario 1

This scenario simulates a peak load which happens when a service is facing a sudden demand. Such as accessing the canteen online menu just before the lunch break. Peak loads represent a problem in the real world, as countermeasures are most difficult.

TABLE V. SCENARIO 1 VALUES

Parameter	Value
min VMs	2
max VMs	10
runtime	180s

Figure 7 shows the simulation results with the conventional rules. Here it can be seen that the simulation begins with the minimum of 2 virtual machines in the pool. Until about 35s in the simulation the load is low enough for this two machines to cope with. After this point the average response time is rising slowly up until 50s where the load gets increased more. From this point, the response time increases sharply, until the first

boundary limit of 15,000ms is hit and an additional virtual machine gets started.

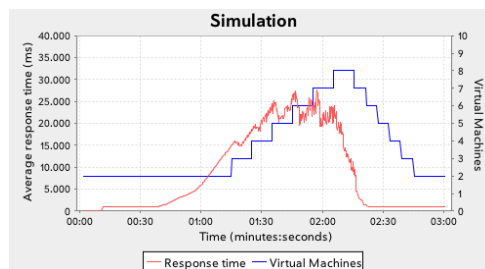


Fig. 7. Scenario 1 Conventional Results

The start of another VM is just not enough to improve the response time significant. Throughout the simulation up to 8 VMs are running simultaneously to manage the load. Comparing these results with the fuzzy controlled results, where the prediction is set to *low*, shown in figure 8, it becomes clear that they are pretty similar. This is because with a *low* prediction, the limits for the up scale are corresponding to those of the conventional rule. Therefore the regulation starts on the same load adding VMs. When switching off VMs, the fuzzy scaler depending on the load cuts off two VMs. This behavior has however no effect on the in this simulation already sinking response time, but it could save resources and money in real life situations. In both cases the QoS limit of 20,000ms is exceeded.

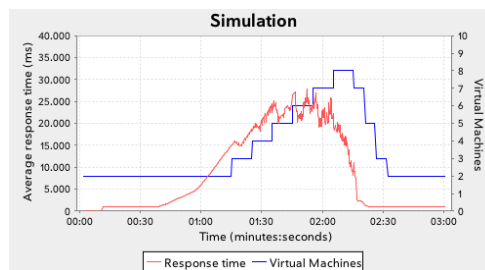


Fig. 8. Scenario 1 Fuzzy Low Prediction Results

The large fluctuations seen at the peak of the load can be explained by the forming the average response time. The individual values between newly started and already longer running VMs vary widely because of the waiting time of the processing packets in the different VMs input queues.

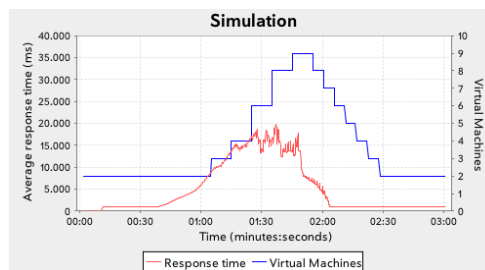


Fig. 9. Scenario 1 Fuzzy High Prediction Results

Figure 9 shows the results for the simulation with the fuzzy scaler and an *high* prediction. Compared to the other two tests

we can clearly see that the QoS limit of 20,000ms is respected this time. In general, we see that the response time runs in a similar but shallower curve. This can be attributed to the stronger up scale of starting two VMs simultaneously. This marginal difference is sufficient to prevent the response time from increasing over the QoS limit. The earlier intervention, which is already engaged at a normal load, prevents the requests from accumulating in the input queues of the VMs. Overall, though, more resources are used than in the other tests.

A comparison between the conventional rules and fuzzy scaler with *regular* prediction is not necessary, because these two sets are the same and thus generate the same results.

Over the running of all tests it has show that in all the scenarios considered, the fuzzy scaler is beneficial. Although this scaler uses in the *high* prediction more resources, for which some could argument it will cost more money, is the benefit in comparison greater, since a service in where less resource are needed but has no decent response times makes no sense to use. The *low* prediction did not improve the archived response time but releases the allocated resources faster than the conventional rules, therefore making it more economical. By the above presented tests it could be shown that by simple means, such as a fuzzy rule set and knowledge in form of an prediction, the response time could be improved or resources could be saved.

VI. CONCLUSION AND FUTURE WORK

The goal of this paper was to show how a common cloud computing scaling service could be enabled to guarantee QoS parameters. Especially the KPI, request-response-time, has been the focus. The extended QoS provisioning architecture with an fuzzy control module has been delineated. A detailed description of possible new information to improve the scaling control system has been discussed. The proof of concept chapter showed that violation of SLAs could be avoided.

Future work is to proof this results within a real test environments and to develop an easy to use user interface. This shall allow users to specify imprecise information input and expert knowledge. Additionally the expansion to other QoS parameters, and further fuzzy input data has to be examined.

ACKNOWLEDGMENT

This research is supported by the German Federal Ministry of Education and Research (BMBF) through the research grant number 03FH046PX2.

REFERENCES

- [1] Amazon auto scaling service. Online. Retrived 01.2013. [Online]. Available: <http://aws.amazon.com/autoscaling/>
- [2] D. Armstrong and K. Djemame, "Towards quality of service in the cloud," 2009, pp. 226 – 240.
- [3] B. Rochwerger, A. Galis, E. Levy, J. Caceres, D. Breitgand, Y. Wolfsthal, I. Llorente, M. Wusthoff, R. Montero, and E. Elmroth, "Reservoir: Management technologies and requirements for next generation service oriented infrastructures," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, 2009, pp. 307 –310.
- [4] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for qos-aware clouds," in *Proceedings of the 5th European conference on Computer systems*, ser. EuroSys '10, New York, NY, USA, 2010, pp. 237–250.
- [5] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "QoS-Aware Clouds," in *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 321–328.
- [6] C.-T. Yang, K.-C. Wang, H.-Y. Cheng, C.-T. Kuo, and W. C. C. Chu, "Green power management with dynamic resource allocation for cloud virtual machines," in *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*, ser. HPCC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 726–733.
- [7] F. Wuhib, R. Stadler, and M. Spreitzer, "A gossip protocol for dynamic resource management in large cloud environments," vol. 9, no. 2, 2012, pp. 213–225.
- [8] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," vol. 11, no. 3. Hingham, MA, USA: Kluwer Academic Publishers, Sep. 2008, pp. 213–227.
- [9] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "DynaQoS: model-free self-tuning fuzzy control of virtualized resources for qos provisioning," in *Proceedings of the Nineteenth International Workshop on Quality of Service*, ser. IWQoS '11. Piscataway, NJ, USA: IEEE Press, 2011, pp. 31:1–31:9.
- [10] Y.-M. Wi, S.-K. Joo, and K.-B. Song, "Holiday load forecasting using fuzzy polynomial regression with weather feature selection and adjustment," vol. 27, no. 2, may 2012, pp. 596 –603.
- [11] K.-B. Song, S.-K. Ha, J.-W. Park, D.-J. Kweon, and K.-H. Kim, "Hybrid load forecasting method with analysis of temperature sensitivities," vol. 21, no. 2, may 2006, pp. 869 – 876.

Scalable Store and Forward Messaging

Ahmed El Rheddane, Noël De Palma, Alain Tchana
LIG/UJF, Grenoble, France

{ahmed.el-rheddane, noel.de_palma, alain.tchana}@imag.fr

Abstract—Since the emergence of the Internet, and particularly with the outburst of cloud computing, the production of reliable and scalable distributed applications is an important area of research. Various middleware technologies were designed for that purpose, among which we find Message-Oriented Middleware (MOM), which provides reliable asynchronous communication through message queuing techniques. MOMs have been standardized using the AMQP protocol, and in the Java world, with the JMS API.

In this paper, we extend a store and forward mechanism to improve the scalability of an end-to-end reliable asynchronous messaging infrastructure while remaining compliant to the standard JMS API. We design a flow control based load balancing policy that, on the one hand, reduces the risk of consumer queues' failures while maintaining a near optimal throughput; and on the other hand, insures the scalability of our load balancing mechanism on the producer's side. We report the evaluation of our solution deployed on a cloud computing infrastructure and implemented within Joram, an open source implementation of the JMS API and the AMQP queuing protocol. This work is now part of the Joram distribution available on the OW2 consortium.

Keywords—JMS; message queues; scalability; load balancing; flow control

I. INTRODUCTION

Today's applications often run on distributed resources. One of the most commonly used ways to simply yet reliably integrate the different components of a distributed software system is through a message-oriented middleware (MOM). MOMs use messages as the only structure to communicate, coordinate and synchronize, thus allowing the components to run asynchronously. MOMs offer two communication paradigms: *one-to-one*, producers send messages to a queue where they are stored till they are consumed by one and only one consumer; and *one-to-many* or *publish-subscribe*, a producer sends a message to a topic that broadcasts it to all the subscribed consumers. Java, with a concern of providing the community with a universal messaging interface has standardized the Java Message Service API (JMS) [1]. This, while making sure that all message-oriented applications would be easily integrated, gives the developers the choice of the implementation beneath depending on their specific needs with regard to reliability and overall performance.

The most intuitive MOM configuration consists in having one server, with the desired queue, generally on the consumer's side, thus rendering the distant communication channel between the producer and the queue vulnerable in

the case of failures. Instead, a more reliable MOM ensures a store and forward mechanism. This mechanism requires a reliable communication model between producers and queues based on the following properties:

- *Asynchrony*: the asynchronous property decouples producers from queues. They do not need to be both ready for execution at the same time. This property enables a deferred access to queues and a loose coupling between producers and consumers.
- *Reliability*: once a message is sent, it is guaranteed to be delivered despite network failures or system crashes.

In this work, we consider the specific case of applications with symmetric consumers, i.e., all the consumers process the same tasks. We also position ourselves in the context of cloud computing, where the consumers might belong to different clouds and their performance varies depending on the load of the cloud, since the virtual machines might share the same physical resources thus affecting each other's performances. Taking this into consideration, we aim to improve the scalability of the store and forward mechanism with clustered queues: we propose a new load balancing policy based on flow control, which dynamically adapts the messages' load on each of the cluster's queues to its consumption rate; this will be highlighted by comparing our scalable store and forward solution to a static load balancing policy such as round-robin. Load balancing is moreover done on the producer's side so as to allow intercloud consumers' deployment. Last but not least, our solution includes a failover mechanism in order to enhance its reliability.

We implemented and evaluated our solution using Joram, for Java Open Reliable Asynchronous Messaging [2], deployed on a cloud computing infrastructure. Joram is a pure Java implementation of the JMS API. It also implements the Advanced Message Queuing Protocol (AMQP) [3].

The rest of this paper is organized as follows: Section II describes our store and forward mechanism and shows how we improve its scalability; Section III formally describes the scalability of queue messaging; in Section IV we detail the proposed load balancing strategy, which we later evaluate in Section V; then we present the related work in Section VI before finally concluding this work in Section VII.

II. STORE AND FORWARD WITH LOAD BALANCING

To provide a store and forward mechanism, a MOM must insure both the asynchrony between producers and the

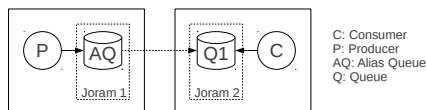


Figure 1. Alias queue's principle

queues deployed on consumers' side, and the communication reliability between them. For that purpose, a solution is to use a special destination called an *alias queue*. An alias queue is a special persistent queue that automatically forwards the messages it is sent to another, generally distant, persistent queue on the consumer's side (see Figure 1). It is set to write-only mode as the "real" destination is meant to be the queue to whom the messages are forwarded. The alias queue would thus be an intermediate destination on the producer's side where messages will be stored, and *visible* (i.e., can be monitored), till they successfully reach their final destination. This persistent pair of queues enforces the asynchronous property. To enforce reliability, the forwarding mechanism involves a distributed transaction between the alias queue and the related queue. This transaction insures, despite network or system failures, that a message is either stored on the persistent alias queue on the producer's side or on the persistent queue on the consumer's side.

The aim of this paper is to improve the scalability of this store and forward mechanism. We propose a new load balancing policy based on flow control described in Section IV. To implement this policy, we extended the alias queue mechanism to support load balancing. This extension is based on a well-known load balancing pattern similar to Web-based system (e.g., JK Apache Tomcat Connector [4]). Each producer is assigned to an alias queue that would distribute the messages to a set of distant clustered queues each corresponding to a set of local consumers (see Figure 2). We also integrated a failover mechanism that allows messages to be re-sent to another queue if their initial destination is unavailable. Note that this pattern is not exactly the same as the one used for Web systems since: (i) load balancing is achieved on the producers' side; and (ii) both the producers' and the consumers' sides can be controlled. Also, this is different from the one-to-many messaging paradigm provided by topics, as one message will be forwarded to one and only one of the cluster's queues. We will see in the following sections how this affects MOM's scalability and what the different strategies of distributing messages between our multiple destinations are.

III. SCALABLE MESSAGING

In this section, we discuss the different factors that affect the performance of a messaging system. First, we will start with the case of a standard queue then generalize our approach to clustered queues using an alias queue as a forwarding mechanism.

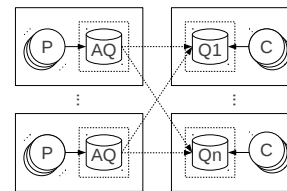


Figure 2. Scalable queuing with enhanced alias queues

A. Standard Queues

Let p be the production rate on the queue and c the consumption rate. l being the length of the queue, i.e., the number of waiting messages, we have:

$$\Delta l = p - c$$

Depending on the result, three cases can be identified:

- $\Delta l > 0$: This means that the queue receives more messages than it is asked to deliver. The number of pending messages grows and we say that the queue is *unstable* and *flooded*.
- $\Delta l < 0$: In this case, the consumption rate is higher than the potential reception rate and receivers are blocked waiting for new messages to come. The queue is still *unstable* and we say that it is *draining*. This means that the queue's resources are underutilized.
- $\Delta l = 0$: Here, the consumption rate matches the reception rate and the queue is *stable*. This is the ideal case that we aim to achieve.

The stability of a queue is thus defined by the equilibrium between the messages' production and consumption.

B. Clustered Queues

In this case, our alias queue, to which the messages are sent, is wired to n queues, on which the messages are received. Let p be the production rate on the alias queue, c_i the consumption rates on each of the consumers' queues, and l_i their respective lengths. The scalability of our distributed system can be discussed on two different levels:

1) *Global Scalability*: Let L be the total number of waiting messages in all the consumers' queues. We have:

$$L = \sum_{i=1}^n l_i \text{ and } \Delta L = p - \sum_{i=1}^n c_i \quad (1)$$

The overall stability of our system is given by: $\Delta L = 0$. This shows that, globally, our system can handle the global production load. However, it fails to guarantee that on each consumer queue, the forwarded load is properly handled. This will be guaranteed by *local scalability*.

2) *Local Scalability*: Depending on how we distribute the messages between the different queues, each would receive a ratio r_i of the total messages produced on the alias queue. Thus, for each $i \in \{1..n\}$ we have:

$$\Delta l_i = r_i \cdot p - c_i \quad (2)$$

Local scalability is then given by:

$$\forall i \in \{1..n\}; \Delta l_i = 0 \tag{3}$$

Note that local scalability implies global scalability as:

$$\forall i \in \{1..n\}; \Delta l_i = 0 \Rightarrow \Delta L = \Delta \sum_{i=1}^n l_i = \sum_{i=1}^n \Delta l_i = 0 \tag{4}$$

In the remaining of this paper, we will suppose that global scalability is verified and try to achieve local scalability by tuning our load balancing strategies.

IV. FLOW CONTROL POLICY

The main question that arises when forwarding messages to different destinations is how to achieve load balancing, i.e., how to distribute the received messages over the clustered destination queues. In this work, we propose a dynamic load balancing strategy based on flow control, i.e., the consumption rates of our consumers. As a reference load balancing strategy, we choose round-robin; we could also have chosen random, which is statistically equivalent, and would ultimately give the same results.

A. Round-Robin

The first implemented strategy is the simplest. It consists in forwarding messages uniformly over our destinations: we would forward the first message to the first queue, the second to the next one etc. Till we are out of destinations, in which case we go back to send to the first queue and so on.

If we take up the forwarding ratios introduced in the previous section, this strategy can be described as follows:

$$\forall i \in \{1..n\}; r_i = \frac{1}{n} \tag{5}$$

n being the number of queues wired to our alias queue.

While this strategy is straightforward to implement, and can even be effective if all the consumer queues have the same consumption rate; it can also result in local instability if our queues have different consumption rates. Besides, it is static, which makes it unable to follow the potential variation of our distributed messaging system. Thus, a more sophisticated adaptive strategy is needed.

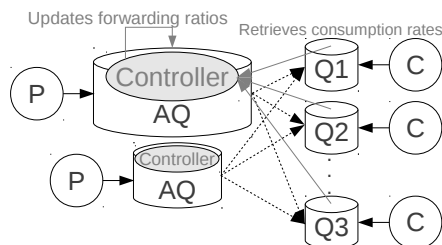


Figure 3. Load balancing controller

Algorithm 1 Flow control's algorithm

```

while TRUE do
  for each consumer queue c do
    rate[c] ← c.monitorConsumptionRate()
    load[c] ← c.monitorLoad()
  end for
  for each consumer queue c do
    weight[c] ← computeConsumerWeight(rate[])
    if load[c] > MAX_LOAD then
      weight[c] ← weight[c]*9/10
    end if
  end for
  p.updateWeights(weight[])
  sleep(period)
end while
    
```

B. Flow Control Principle

Flow control is a dynamic strategy that allows a consumption-aware message distribution. Its mechanism, described by Figure 3, relies on a controller integrated with our alias queues, which has a representation of their interconnections with the consumers' queues. The controller's integration guarantees our solution's scalability with regard to producers since each alias queue has its own load balancer instead of having one centralized load balancing controller. It is also easy to use for an end-user as load balancing is done transparently without any extra configuration.

Our controller periodically monitors the system, retrieving particularly the consumption rates of the consumer queues, i.e., the number of messages each of the cluster's queues has been asked to deliver over the last period. The decision process can then be formally described as follows: let us say that for the k -th period, we retrieved $c_i(k)$ as consumption rates for our queues. In order to make sure that the more a queue consumes messages, the more messages it will be sent, the expression of our $r_i(k+1)$ for the next period is:

$$\forall i \in \{1..n\}; r_i(k+1) = \frac{c_i(k)}{\sum_{i=1}^n c_i(k)} \tag{6}$$

As for the overload that might occur on a queue before its forwarding ratio is regulated, we propose to define a maximum load limit per queue, above which its forwarding ratio will be artificially decreased so as it can handle part of its pending messages.

Naturally, the controller executes its decision by replacing the old $r_i(k)$ with the newly computed $r_i(k+1)$. Technically, Algorithm 1 details the different steps that our controller goes through, where *computeConsumerWeight* implements $r_i(k+1)$'s expression. The weights used in our implementation are directly proportional to our forwarding ratios, they represent the number of messages that will be forwarded to the same queue before changing destinations.

The only question left to be answered is how to determine the period of our control loop. While having a shorter loop increases the reactivity of our system, it also induces a greater overhead as it involves exchanging monitoring messages more frequently. The solution we propose aims at maximizing the reactivity of our system while controlling its induced overhead. We do not fix the period itself, but we fix a tolerated overhead, i.e., the ratio of monitoring messages to the produced throughput: at each iteration, we determine the next period based on last period’s throughput so as to stay within the tolerated overhead.

Our consumption-aware load balancing strategy takes into consideration the differences between our consumer queues in terms of consumption rates, which, a priori, vary with time, and should improve the performance of our system. The second part of the evaluation section verifies this assumption.

V. EVALUATION

Now that our scalable distributed messaging system is properly geared, we have to check its efficiency. To do so, we started by evaluating the proper overhead of the alias queue, and we went on to compare the performances of our two load balancing strategies. Note that overhead always refers to the effect of using an alias queue on performance.

For our evaluation we used virtual machine instances of type m1.small as described by Amazon EC2 [5], i.e., 2GB memory and 1 VCPU, provisioned on a private cloud running racks with two 6 cores Intel(R) Xeon(R) CPU E5645 @ 2.40GHz, 32GB RAM, 1GBps isolated LAN and managed by OpenStack [6]. All our results are computed over campaigns of 1,000,000 messages of 1kB each. Our solution has been implemented and tested using Joram.

A. Alias Queue’s Overhead

In our first set of experiments, we want to evaluate the maximum capacity of Joram with and without using alias queue. Our metric here is the maximum throughput, i.e., maximum number of consumed messages per second. Figure 4 shows the results of these experiments, presented in pairs: either using an alias queue or not.

The first two experiments put both the consumer and producer on the same virtual machine, and use only one Joram server. We can see that the general throughput slightly decreases when using an alias queue as an intermediate message, along with its acknowledgement, is added. This overhead is however less than 4%, as intra-server communication is highly optimized in Joram.

The experiments 3 and 4, add a new Joram server, to evaluate the overhead when messages go through an intermediate server instead of directly reach their final destination; this corresponds to the reliable set-up discussed in the store and forward, subsection of section II, even though both servers are co-located on the same virtual machine. We can see that,

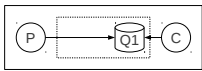

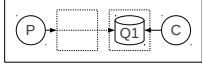
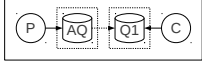
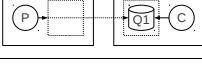
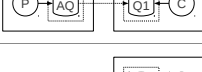
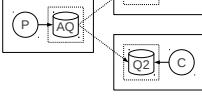
No	Configuration	msg/s
1		2291
2		2211
3		2052
4		2001
5		1944
6		1918
7		3678

Figure 4. Alias queue’s overhead evaluation

in this case the overhead is even smaller (2.5%), as extra messages are needed for the forwarding even without the alias queue.

In the scenario depicted by the experiments 5 and 6, which is the most realistic since the communication is done between two different virtual machines, we can see that the alias queue’s overhead drops to about 1%. Moreover, in this particular case, the virtual machines are co-located; should we consider the latency as well, the alias queue’s overhead can fairly be neglected.

Now that we have established that alias queues’ utilization has almost no overhead, the 7th experiment of Figure 4 shows how this mechanism can be used to enhance the scalability of our messaging system. The resulting throughput, which is roughly two times the previous one (experiment 6), shows that adding consumer queues to the alias queue linearly increases the system’s performance.

In this particular case, the consumers were both identical, as they were both running on maximum speed, on similar virtual machine instances. Thus, the simple round-robin strategy was enough. In the next part, we will see how flow control is sometimes necessary for Joram to work properly.

B. Flow Control Evaluation

To evaluate our dynamic load balancing strategy, we regulate the sending and receiving rates of our clients and calculate the total time needed to receive the 1,000,000 sent messages. We also monitor the system, particularly the queues load during the experiments. Based on the previously

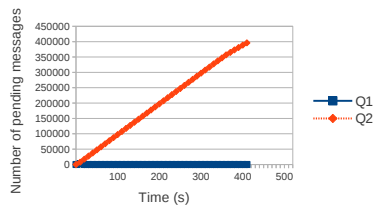


Figure 5. Consumer queues' load evolution

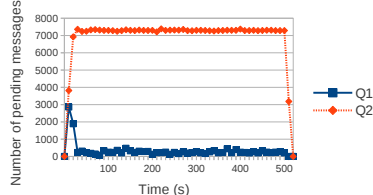


Figure 6. Consumer queues' load evolution in flow control mode

computed maximum throughput (experiment 4, Figure 4), the production rate used for the following experiments is: **2000msgs/s**. The consumption rates of the queues are given as a percentage of the production rate and varies as follows:

1) *One producer and two identical consumers:* The configuration with 1 producer and 2 consumers is similar to the one set for the 7th experiment of Figure 4. Our experiments show that round-robin takes a total time of **500.0s**, whereas our flow control policy results in a total reception time of **500.5s**. This is the ideal case where both consumers receive messages at the same rate (50% of the produced load each), round-robin is here the perfect solution. However, we see that even when our flow control mechanism is activated, it gives us about the same performance. The overhead is due to expected side-effects in the computation of weights as we had to settle for a level of granularity.

2) *One producer and two unbalanced consumers:* In this case, our consumers have significantly different consumption rates (70% and 30%). Round-robin is not at all suited for such a configuration, it expectedly resulted in the time-out of the slowest consumer: it couldn't receive all the forwarded messages in a reasonable time. This is mainly due to the overload on the consumer's queue, as on each round, it keeps 20% of the forwarded messages, which later affects its ability to respond to the consumer's client requests. Figure 5 shows the evolution of the slow consumer's queue load.

We can see that the number of waiting messages on the slowest consumer's queue is growing linearly, and while this queue is flooded, the other is draining. This badly affects the overall performance of the system. Flow control, on the other hand, achieves a total reception time of **510s**, which is not very far from the ideal 500s. The delay is due to the fact that the flow control loop's initial period is 10s, which means that it takes 10s for the first flow control regulation to take place. Figure 6 shows that flow control regulated the forwarded messages to insure a balanced load on both consumers' queues.

3) *Two producers and two variable consumers:* Figure 7 shows the configuration set up for this experiment, which

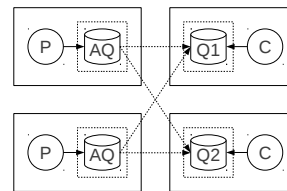


Figure 7. 2 producers, 2 consumers configuration

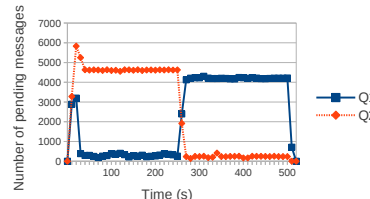


Figure 8. Consumer queues' loads with changing rates

is meant to prove two things: first, that our mechanism can work with more than one producer (i.e., alias queue); more importantly it shows that our flow control effectively adapts to any consumption rates' variation as we start with consumers receiving with 70%-30% rates and invert them on $t = t_0 + 250s$ to 30%-70%. Figure 8 describes the queues' loads during this experiment.

As you can see in Figure 8, the queues' loads are stabilized throughout the experiment, which results in a total reception time of **506s**. This surely concludes the effectiveness of the flow control mechanism.

VI. RELATED WORK

While in our present work, we apply load balancing policies to message-oriented middleware, many previous works have detailed different load balancing strategies, particularly for web-based applications [7], [8], [9], [10]. These policies have been classified as content-blind or content-aware based on whether they take into account requests being forwarded. *Round-robin* and *weighted round-robin* are obviously content-blind. Other content-blind policies are *random*, which dispatches messages randomly between the worker servers; *least connection* and *least loaded*, which forward messages respectively to the server with the least number of connections and the one with the least load, with regard to the server's capacity and current utilization. Content-aware policies aim to achieve better efficiency by taking into account for instance the sessions established between the clients and servers and forward the packets belonging to the same session to the same servers, these are then called *sticky sessions* [11]. Another content-aware policy consists in taking into account the locality of the clients and forward their requests to the nearest servers. While these policies in general aim at optimizing the performance of the system, other studies [12], [13] focus on the energy efficiency of such policies. Our flow control policy is therefore content-blind, it also differs from the previous policies by its integrated store-and-forward mechanism.

Load balancing has also been widely addressed in the context of high performance computing in grids or multi-processor machines [14], [15], [16] where distributed load balancing, which involves exchanging loads between neighbor computing nodes is rather privileged.

Commercial message-oriented middlewares do also integrate load balancing, IBM's WebSphere MQ [17], for instance, provides a basic round-robin policy for its cluster queues that can be enhanced by statically specifying weights for queues in order to manage their priority. Another example is HornetQ [18], which also distributes the loads over its queue clusters on a round-robin basis, it excludes however the queues with no connected consumer. Finally, Oracle's BEA WebLogic [19] JMS implementation also offers load balancing with policies limited to round-robin and random.

In the specific case of Joram, a previous work [20] has addressed scalability differently: producers are statically affected each to a specific consumer queue; these consumer queues are interconnected (clustered queues) in a way that each draining queue will see if the others have extra messages and "steal" them, likewise, once a queue's load reach a certain limit it distributes, if possible, the extra load over the other queues. Whereas this is a corrective policy that handles problems when they occur, which results in extra traffic on our system, as a message is first sent to a queue, then it is potentially forwarded as many times as necessary; our work is based on a predictive policy that tries to forward the messages to the "right" queues in the first place.

VII. CONCLUSION

Message-oriented middlewares have proven to be an effective way to integrate the components of a distributed software system, both guaranteeing asynchrony and end-to-end reliability thanks to their store and forward mechanisms. In this paper, we described and extended the store and forward mechanism of a MOM infrastructure in order to improve its scalability with regard to both the producers and the consumers, while maintaining the JMS API compatibility. Our extension includes the design of a flow control based load balancing policy to insure the local stability of the clustered queues. This has been done with the concern of providing a scalable distributed mechanism that would be totally transparent to the end-user. The evaluation of our solution, carried out on a cloud computing infrastructure, shows the effectiveness of our design compared to a basic load balancing policy. As a future work, we intend to enhance our solution to support the elasticity of message-oriented middleware using the flexibility offered by cloud computing infrastructures. We will thus go beyond the static dimensioning the queues and develop a dynamic provisioning mechanism that would scale automatically the clustered queues based on the total load of our system.

ACKNOWLEDGEMENT

We'd like to thank ANR INFRA for supporting this work.

REFERENCES

- [1] "JMS Concepts," [retrieved: Mar., 2013]. Available: <http://docs.oracle.com/javase/6/tutorial/doc/bncdq.html>
- [2] "Joram home page," [retrieved: Mar., 2013]. Available: <http://joram.ow2.org/>
- [3] "AMQP home page," [retrieved: Mar., 2013]. Available: <http://www.amqp.org/>
- [4] "The Apache Tomcat Connector," [retrieved: Mar., 2013]. Available: <http://tomcat.apache.org/connectors-doc/index.html>
- [5] "Amazon Elastic Compute Cloud home page," [retrieved: Mar., 2013]. Available: <http://aws.amazon.com/ec2/>
- [6] "OpenStack home page," [retrieved: Mar., 2013]. Available: <http://openstack.org/>
- [7] D. M. Dias, W. Kish, R. Mukherjee, and R. Tewari, "A scalable and highly available web server," in Proceedings of the 41st IEEE Computer Conference (COMPCON), 1996, pp. 85–.
- [8] V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic load balancing on web-server systems," in IEEE Internet Computing, vol. 3, May 1999, pp. 28–39.
- [9] K. Gilly, C. Juiz, N. Thomas, and R. Puigjaner, "Adaptive admission control algorithm in a qos-aware web system," in Inf. Sci., vol. 199, Sep. 2012, pp. 58–77.
- [10] B. Yagoubi and Y. Slimani, "Dynamic load balancing strategy for grid computing," in Transactions on Engineering, Computing and Technology, 2006.
- [11] L. Cherkasova and P. Phaal, "Session-based admission control: A mechanism for peak load management of commercial web sites," in IEEE Trans. Comput., vol. 51, 2002.
- [12] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2008, pp. 337–350.
- [13] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in Proceedings of the 2nd Workshop on Power-Aware Computing Systems, 2002, pp. 179–196.
- [14] C. Xu and F. Lau, "Iterative dynamic load balancing in multicompilers," in Journal of Operational Research Society, vol. 45, 1994, pp. 786–796.
- [15] R. Diekmann, B. Monien, and R. Preis, "Load balancing strategies for distributed memory machines," in Multi-Scale Phenomena and Their Simulation, 1997, pp. 255–266.
- [16] Y. Li and Z. Lan, "A survey of load balancing in grid computing," in Proceedings of the First international conference on Computational and Information Science (CIS), 2004, pp. 280–285.
- [17] "WebSphere MQ V6 Fundamentals," [retrieved: Mar., 2013]. Available: <http://www.redbooks.ibm.com/redbooks/pdfs/sg247128.pdf>
- [18] "HornetQ home page," [retrieved: Mar., 2013]. Available: <http://www.jboss.org/hornetq/>
- [19] "Introduction to WebLogic JMS," [retrieved: Mar., 2013]. Available: http://docs.oracle.com/cd/E13222_01/wls/docs81/jms/intro.html
- [20] C. Taton, N. De Palma, J. Philippe, and S. Bouchenak, "Self-optimization of clustered message-oriented middleware," in Proceedings of the 4th International Conference on Autonomic Computing (ICAC), 2007.

Towards a Method for Decision Support in Multi-cloud Environments

Aida Omerovic
SINTEF ICT
Norway

Email: aida.omerovic@sintef.no

Victor Muntés-Mulero and Peter Matthews
CA Technologies
CA Labs Europe

Email: {victor.muntes,peter.matthews}@ca.com

Alexander Gunka
BOC Information Systems
Austria

Email: alexander.gunka@boc-eu.com

Abstract—Providers of cloud services as well as the cloud services themselves differ in the business models, functionality, quality of service, cost, value, etc. which makes the choice of a provider and a service difficult. Beyond that the complexity and lack of transparency with respect to cost and quality render the run-time adaptation and replacement of services almost impossible. This position paper presents main results of our recent efforts towards development of a decision support method (DSM) in multi-clouds. The DSM aims at taking into account risk, quality and cost aspects in order to assist a decision maker in choosing providers and services in a multi-cloud environment. We characterize the needs for the DSM in the multi-cloud context and propose an initial version of the process for the DSM. Based on the method proposed and the needs identified, we elaborate to what degree the current state of the art can be leveraged and what further multi-clouds-specific extensions are needed.

Keywords—multi-cloud; decision support; risk assessment; quality prediction; cost prediction; architectural design; trade-off analysis; cloud service selection; cloud provider selection.

I. INTRODUCTION

The rapidly increasing number of cloud services and cloud service providers opens for new opportunities [1] in designing application and enterprise architectures. It also enables new business models and investments [2] [3] [4], new quality levels [5], as well as new capabilities. The services can be orchestrated and their compositions adapted even more dynamically than earlier. Availability of similar services from several providers opens for replaceability between services, or redundancy of services. As a result, the quality may improve and the risk of vendor lock-in will normally be reduced. However, there are also significant challenges [6] involved in realizing collaborations between clouds. One of the major challenges regarding cloud services and their providers is that they differ in the business models, functionality, quality of service, cost, value, etc. Another challenge is complexity and lack of transparency with respect to cost and quality. This makes the choice of a provider and a service difficult and the run-time adaptation and replacement of services almost impossible. When selecting the cloud services and the cloud providers, systematic support for identifying the candidate services and understanding the implications of choosing the different alternatives, is needed.

Decision support [7] for multi-cloud environments imposes several challenges compared to the traditional model-based decision support. Most notably, the dynamics of multi-cloud require light-weight processes and tools, the decision makers

depend on easy-to-understand representations of the impacts of the decisions, the notion of cost is to a lower degree established in the existing approaches supporting the trade-off analysis of enterprise and software architectures, and a merge of the aspects of risk, cost and quality in a consolidated view imposes a new complexity as well as methodological challenges.

The specific objective of this paper is to establish the necessary baseline for a tool-supported decision support method (DSM) aimed at facilitating selection of cloud services and providers in a multi-cloud environment. In particular, we argue that risk, quality and cost are among the main three factors in such a selection process. To that end, we aim at providing a decision support which analyses the impacts of the possible decision alternatives in a multi-cloud environment with respect to those three factors. We believe that a trade-off analysis between risk, cost and quality based on a consolidated view of the three will provide a useful basis for a decision maker in assessing the possible choices through a cost-benefit analysis.

This position paper presents the main results of the recent efforts towards development of a DSM for multi-cloud environments. We characterize the needs for the DSM in the multi-cloud context and propose an initial version of the process for the DSM. Based on the method proposed, we elaborate on the suitability of both the method proposed and the state of the art for analyzing risks as well as for predicting quality and cost in the multi-cloud context.

The paper is organized as follows. Section 2 summarizes the state of the art regarding risk analysis, quality prediction, and cost analysis. Section 3 characterizes the needs for the DSM in the multi-cloud context. Section 4 proposes an initial process for the DSM. Section 5 discusses to what degree the state of the art can be leveraged within the DSM process proposed. Main conclusions are provided in Section 6.

II. STATE OF THE ART

The ISO 31000 standard for risk management comes with no specific techniques, modeling languages or recommended tools for how to conduct risk assessment in practice. However, most established risk management methods [8] [9] [10] [11] follow the ISO 31000 process, and provide such additional support. Common for these approaches is that they are designed to support risk management and risk documentation from the perspective of an organization and its policies. There is lack of support in the state of the art for extracting the risk picture that is relevant for specific external stakeholders, such

as services consumers, and to present this picture in an intuitive and easily understandable way. There is also lack of an approach which combines cloud modeling and risk modeling. There exist many different approaches to service modeling [12] [13] [14] [15], focusing on expressing relevant elements and aspects of services, such as actors and components, roles, activities, interfaces and contracts. However, none of these have a risk-oriented view where stakeholders are represented as risk owners, and where the assets at stake are made explicit.

In a model-based decision making, the decisions are made based on a number of factors. The major ones include functional and non-functional properties, as well as cost and the added value. A trade-off between such factors is the basis for decision making. This trade-off is particularly complex between the non-functional factors, the variable parts of the architecture, and the cost of the selected solutions. The variability, as well as incomplete information or knowledge, are also sources of risk. Since functional requirements normally are less flexible and specified rather early, and since the added value is strongly related to the functional properties, the factors that are tunable and highly interrelated are risk, quality and cost. Therefore, in a model-based decision making, the decisions are based on a trade-off assessment between risk, quality and cost. The risk assessment, in turn, is based on information that is gathered about assets, entities, actors, etc. that are involved in the service event or action in question.

As a basis for the elicitation of the adequate quality characteristics, we may use the software product quality standard ISO/IEC 9126 [5]. The ISO 9126 defines quality as “the totality of features and characteristics of a software product that bear on its ability to satisfy stated and implied needs”. The ISO 9126 standard provides an established specification of decomposed quality notions with their qualitative and quantitative definitions. The standard defines a quality model for external and internal quality, and for quality in use. External quality is the totality of the characteristics of the software product from an external view when the software is executed. Internal quality is the totality of characteristics from an internal view and is used to specify properties of interim products. The characteristics of the internal and external quality model are functionality, reliability, usability, efficiency, maintainability and portability. These are in turn decomposed into a total of 34 sub-characteristics. Quality in use is the user’s view of the quality of the software product when it is used in a specific environment and a specific context of use. The quality in use characteristics are effectiveness, productivity, safety and satisfaction. There is also a further decomposition of all characteristics into the related metrics.

SMI [16] is a standardization effort from the Cloud Services Measurement Index Consortium (CSMIC) consisting of academic and industry organizations. The Service Measurement Index (SMI) uses a series of characteristics and measures to create a common means to compare different services from different suppliers. The characteristics are categorized as Usability, Performance, Agility, Security and Privacy, Financial, Assurance and Usability. Each of these characteristics has a number of measures that can be used to evaluate the risk in using a service. For example in the accountability category one of the measured attributes is Compliance and another is SLA verification both of which can be used to create a risk measure

for the service and the provider. CSMIC is in negotiation with a number of large standardization organizations to develop a joint working group and specification.

According to Fenton and Neil [17], most prediction models use size and complexity metrics to predict defects. Others are based on testing data, the quality of the development process, or take a multivariate approach. The goal/question/metric paradigm [18] [19] is a significant contribution to quality control and can be used for development of quality models and for the design of a measurement plan [20] [21]. To enable explicit risk and quality assessment, we make use of monitoring and measurement. Risk monitoring is a means to facilitate continuous risk assessment by the monitoring of relevant key indicators or metrics. An indicator can be defined as “something that provides a clue to a matter of larger significance or makes perceptible a trend or phenomenon that is not immediately detectable” [22]. To enable explicit risk and quality assessment, we make use of monitoring and measurement.

PREDIQT [23] is a tool supported method for model-based prediction of impacts of architectural design changes on system quality characteristics (performance, scalability, security, etc.). PREDIQT facilitates specification of quality characteristics and their indicators, aggregation of the indicators into functions for overall quality characteristic levels, and dependency analysis. The main objective of a PREDIQT-based analysis is prediction of system quality by identifying different quality aspects, evaluating each of these, and composing the results into an overall quality evaluation. This is useful, for example, for elicitation of quality requirements, evaluation of the quality characteristics of a system, run-time monitoring of quality relevant indicators, as well as verification of the overall quality characteristic fulfillment levels. PREDIQT makes use of models that capture the system design, the system quality notions, as well as the relations between them. An important aim of PREDIQT is to enable the right balance between practical usability of the models and the soundness of the predictions. The method is compatible with the ISO/IEC 9126 software quality standard, and has been successfully applied in real-life industrial settings [24] [25].

CORAS [8] is a tool-supported and model-driven approach to risk analysis that is based on the ISO 31000 risk management standard. Whereas alternative state-of-the-art approaches such as CRAMM [26] and OCTAVE [27] rely on text and tables, CORAS uses diagrams as an important means for communication, evaluation and assessment. Risk modeling is a technique for risk identification and assessment, and the state-of-the-art offers several tree-based and graph-based notations. Fault tree analysis [28] (FTA), event tree analysis [29] (ETA) and attack trees [30] are examples of the former and provide support for reasoning about the sources and consequences of unwanted incidents, as well as their likelihoods. Cause-consequence analysis [31] (CCA), Bayesian network [32] and Markov analysis [33] are examples of graph-based notations. CCA employs diagrams that combine the features of both fault trees and event trees, whereas the latter two serve as mathematical models for probabilistic and statistical calculations, respectively.

Approaches to quality assessment, risk analysis and security management provide support for decision making so as to

ensure a required quality level while managing risks. However, while identifying and suggesting options and solutions, such as security mechanisms, the methods often lack techniques and tools for analyzing the associated cost and the return of investment in the identified solutions. Franqueira et al. [2] address this problem by proposing a method for handling security investment decisions achieved by so-called Real Option thinking. The method is partly based on Real Option Analysis [3] (ROA), which is a decision support technique in the area of capital investment by means of mathematical models to evaluate financial options. The method is supported by a security trade-off tool called SecInvest, which is implemented as a Bayesian network topology and supports decision makers in evaluating investment options and identifying the most suitable and cost-efficient ones. Other approaches to cost estimation in the setting of security investments are Net Present Value (NPV) [4], Return on Security Investment (ROSI) [34], Architecture Trade-Off Analysis Method (ATAM) [35], the Cost Benefit Analysis Method (CBAM) [7] and the Security Solution Design Trade-Off Analysis [36]. These and similar approaches can be understood as methods and techniques to facilitate so-called security economics.

III. CHARACTERIZATION OF NEEDS

As a part of context establishment, we elicited quality aspects and risks which are specific to a multi-cloud environments. The elicitation was based on a comprehensive model of migration process. The model was used as a baseline and a checklist for understanding and decomposing the risk, quality and cost aspects. The exercise resulted in a high-level overview of main risks, as well as a model of decomposed quality characteristics which are specific to multi-clouds. The three overall characteristics identified are: interoperability, intercloud replaceability and security. In addition, cost of migration between multi-clouds was classified into cost of personnel, cost of time with two coexisting services, cost of compensation for uncertainty, and cost of hardware and other resources. Through these models, a common understanding of the main risk, quality and cost aspects in our context, was established. The initial experiences and results of the quality, cost and risk classification indicate that:

- Before eliciting the quality characteristics and risks of a multi-cloud based architecture, the context has to be thoroughly defined. Moreover, the architecture models of the target need to be established. This provides a common understanding of the scope and objectives, as well as the necessary frames for further modeling and decision making. For example, during the context establishment, a process model for migration was used as the foundation for eliciting the aspects and indicators related to quality, cost and risk.
- The decision support models should, once available, be able to take the proposed alternatives for architecture design (measures and treatments considered) and, based on each alternative, provide the resulting risk picture, predicted levels of fulfillment of the relevant quality characteristics, as well as the estimated costs. Thus, risk, quality characteristics and cost should be treated as separate concerns.

- Ideally, in order to accommodate for a cost-benefit analysis, the method should consider added value (or profit) in addition to cost. Minimizing cost and risks and maximizing quality levels is not necessarily a realistic goal. In fact, the benefits may arise from e.g. process improvement through the new architecture, improved or extended functionality, or similar. Thus the trade-offs between quality, risk and cost may vary significantly depending on the utility function and the risk attitude of the decision maker. In addition, the trade-off (or “selection criteria”) should take into account the need for balancing the cost with the added value beyond achieving the quality and risk relevant objectives.
- The method should be tool supported, and the tool should at least provide a diagram editor as well as an easy-to-understand presentation of the impacts of the decision alternatives on quality, risk and cost. The tool should also offer the interfaces needed for acquisition of the data needed for evaluation of the indicators, as well as the interfaces for the needed trace-link information.

IV. METHOD FOR DECISION SUPPORT FOR MULTI-CLOUD ENVIRONMENTS – A PRELIMINARY SPECIFICATION

The DSM for multi-cloud applications is a model-driven method consisting of three main artifacts: a process, a language and a tool. This section provides the initial specification of the DSM process and the actors involved. The DSM process consists of three overall phases, and each phase is decomposed into a set of sub-phases. The DSM process is undergone while developing, verifying and applying the comprehensive decision support models which include the aspects of architecture, risk, quality and cost. We assume the following four types of actors involved in the DSM process:

- Analyst: the analyst is an expert in the DSM and has the responsibility for leading and facilitating a DSM-based analysis. That is, the analyst coordinates the overall actors, collects the input for developing the decision support models, interacts with the overall actors during the model development and usage, makes sure that the necessary steps have been conducted within the resources allocated, and validates that the models have the needed quality and contents.
- Decision maker: the decision maker defines the scope and the objective of a DSM-based analysis. He/she will provide the instructions as to what parts of the architecture should be encompassed in the models, the expected validity of the models, the scope and kinds of the perspective changes/revisions of the architecture, etc. The decision maker will also be the main user of the decision models once they have been developed. He will therefore specify the decision alternatives in the decision models, and use the resulting impact estimates with respect to risk, cost and quality as an aid in the decision making. This actor is aware of the business model and strategy of the company. Hence, a decision maker may be a business expert as well, capable of making decisions based on his knowledge

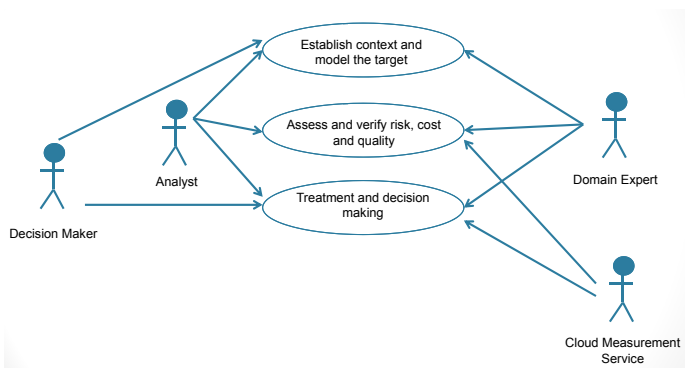


Fig. 1. The top level three phases and the actors involved in the DSM process

of the project budgets, allowable risks and the business processes being supported by the applications. Larger organizations may distinguish between a business expert who builds the requirements specification and a decision maker who selects services based on the specification. For simplicity, these two roles are in our case represented by the decision maker who has all the knowledge sufficient to take decisions.

- Domain expert: normally, a group of domain experts will be involved in a DSM-based analysis in relation to the development, validation and revision of the decision models. The domain experts will contribute by providing the thorough input regarding the current architecture, quality levels, dependencies and processes. The analyst will actively interact with the domain experts during all the three phases of the DSM process.
- Cloud measurement service: this is a (partially) automatized service for retrieval of the empirical data needed for estimating the parameters of the decision models. We assume that the parameters are estimated either based on the feeds from the cloud measurement service or based on expert judgments. A parameter may be estimated or measured either directly, or through estimation of a measurable indicator which then is aggregated and mapped to the decision model through a function. The dynamics of the indicators and the parameters as well as their relevance and uncertainty will be among the factors for determining whether the data acquisition should be automatic (e.g. real-time retrieval based on a monitoring environment) or manual, and how frequent it should be.

Figure 1 shows the overall three phases of the DSM process, as well as the actors involved. In the first phase, the context of the analysis is established. As a part of this, the scope is defined, the relevant risk, cost and quality notions are defined, and the architecture is modeled. In addition, the expected validity as well as perspective business models and architecture alternatives should be anticipated in order to cover the needed scope and level of detail in the target models. During the second phase, the decision models covering the risk, quality and cost aspects are instantiated with respect to target. As a part of this, the dependencies are modeled and the parameters (with the related indicators) are estimated.

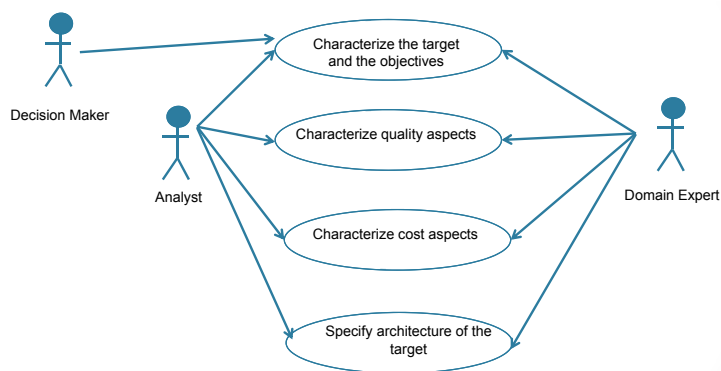


Fig. 2. Establish context and model the target phase decomposed

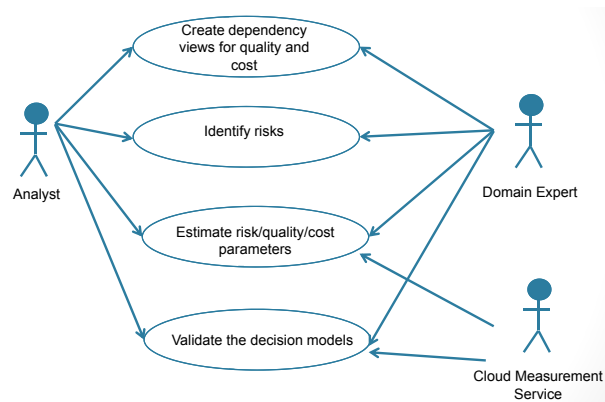


Fig. 3. Assess and verify risk, cost and quality phase decomposed

In addition, the models are validated through various kinds of triangulation, mainly based on the empirical input, logs, domain expert judgments, experience factories, etc. In the last phase, the decision models are applied by first specifying the decision alternatives, applying the alternatives on the models, and finally obtaining the resulting impact of the respective decisions on quality, risk and cost. The result is a consolidated view of the quality, risk and cost picture, provided each decision alternative.

Figure 2 shows the stages of the “establish context and model the target” - phase. First, the target and the objectives are characterized. Based on the initial input, the stakeholders involved deduce a high level characterization of the target architecture, its scope and the objectives of the DSM-based analysis, by formulating the system boundaries, system context (including the usage profile), system lifetime and the extent (nature and rate) of design changes expected. In the second stage, the quality aspects are characterized by specifying which quality characteristics are relevant for the target, and thereafter decomposing them down to indicators. A quantitative and a qualitative definition should be provided for all elements. Thirdly, a corresponding decomposition should be done for the cost aspects. In the last stage, the architecture is modeled with the detail level and within the frames specified during the characterization stage.

Figure 3 shows the stages of the “assess and verify risk, cost and quality” - phase. Firstly, the dependency views for

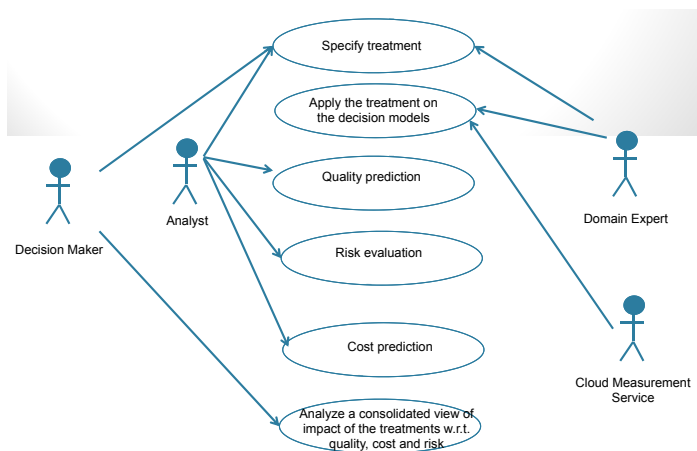


Fig. 4. Treatment and decision making phase decomposed

respectively quality and cost are developed. Secondly, assets and risks are identified in separate decision models (“threat diagrams”). The three types of the decision models (i.e. quality dependency views, cost dependency views and threat diagrams) are then annotated by the parameter values through evaluation of indicators or direct expert judgments on the prior parameters. Finally, triangulation is performed in order to validate the decision models. The models are approved once an acceptable level of uncertainty has been reached.

Figure 4 shows the stages of the “treatment and decision making” - phase. First, the respective decision alternatives are specified separately. Then, each alternative is applied on the decision models. The models and the respective calculus is used to propagate the impacts of each decision alternative on risk, quality and cost. Finally, a consolidated view of the impacts of the decision alternatives is presented to the decision maker.

Figure 5 shows an activity diagram with the entire DSM process, including the feedback loops. The right hand side of the figure indicates the phases presented in Figure 1. The activities are equivalent to the ones presented in relation to Figure 2, Figure 3 and Figure 4.

V. DISCUSSION

This section elaborates to what degree the existing PREDIQT and CORAS methods for for quality prediction and risk analysis, respectively, can serve as a baseline for our DSM in multi-clouds. The objective is to leverage the state of the art decision support, while extending it and adjusting to the special needs of the multi-clouds. Thus, the established methods, languages and tools can be reused with the well known properties and resources, while the efforts can be concentrated on the multi-cloud-specific extensions.

PREDIQT is a method (process, language, and tool support) for model-based prediction of system quality. The PREDIQT method produces and applies a multi-layer model structure, called prediction models, which represent system relevant quality concepts (through “Quality Model”), architectural design (through “Design Model”), and the dependencies between architectural design and quality (through “Dependency

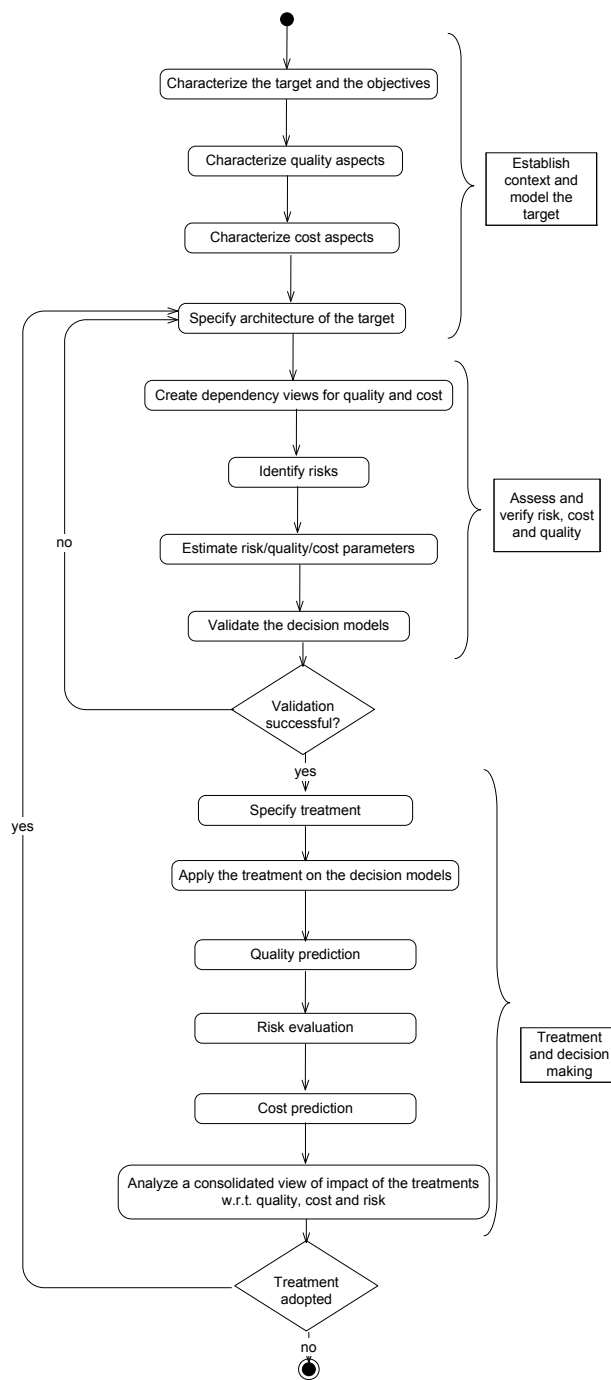


Fig. 5. The DSM process diagram with feedback loops

Views”). The Design Model diagrams are used to specify the architectural design of the target system and the changes whose effects on quality are to be predicted. The Quality Model diagrams are used to formalize the quality notions and define their interpretations. The values and the dependencies modeled through the Dependency Views (DVs) are based on the definitions provided by the Quality Model. The DVs express the interplay between the system architectural design and the quality characteristics. Once a change is specified on the Design Model diagrams, the affected parts of the DVs are

identified, and the effects of the change on the quality values are automatically propagated at the appropriate parts of the DV.

CORAS is a method (process, language, and tool support) for conducting model-based security risk analysis. CORAS provides a customized language for threat and risk modeling, and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis. The Unified Modeling Language (UML) is typically used to model the target of the analysis. For documenting intermediate results, and for presenting the overall conclusions we use special CORAS diagrams which are inspired by UML. The CORAS tool supports documenting, maintaining and reporting analysis results through risk modeling.

The DSM process is based on an attempt to merge the processes of CORAS and PREDIQT for a consolidated analysis of risk, quality and cost. Most of the stages of the DSM process can be found in CORAS and PREDIQT. The actors/stakeholders defined in the DSM are fully compliant with the ones defined by CORAS and PREDIQT. The types of the decision models proposed in the DSM are heavily based on the modeling notations, languages and tools of PREDIQT and CORAS, respectively. The approach to modeling of quality and cost aspects based on the DVs is a part of the PREDIQT method, while a language for risk modeling is provided by CORAS. The respective approaches to modeling in PREDIQT and CORAS are based on graphical modeling languages with defined propagation models. Both modeling approaches are developed with special focus on comprehensibility and expressiveness. In that manner, the models are accommodated for fulfilling real-life needs in terms of covering the representations needed while being rather intuitive so that non-experts should be able to relate to them in an industrial setting. The characterization of quality proposed in DSM is by PREDIQT addressed through the so called Quality Model. Both the Quality Model and the intended quality characterization in DSM are similar to the elicitation we have performed, which is briefly presented in Section 3.

The DSM process is to a high degree a superset of the processes of PREDIQT and CORAS. Moreover, the modeling approaches of PREDIQT and CORAS cover the concerns of quality and risk, as well as partially the concern of cost. Furthermore, the existing tools of CORAS and PREDIQT may be useful in the DSM context. Provided this baseline, we believe that utilization of the CORAS and PREDIQT methods including the processes, the languages and the tools, is worth a further evaluation in the DSM context. In particular, this means that case studies in multi-cloud environments should be performed in order to evaluate the feasibility of DSM, as well as the suitability of the relevant parts of PREDIQT and CORAS in a multi-cloud context.

VI. CONCLUSION AND FUTURE WORK

This position paper aims at establishing the necessary baseline for a DSM. The intended purpose of the DSM is to facilitate the selection of cloud services and providers in a multi-cloud environment. In particular, we argue that risk, quality and cost are among the main factors in such a selection

process. We believe that a trade-off analysis between risk, cost and quality based on a consolidated view of the three will provide a useful basis for a decision maker in assessing the possible choices through a cost-benefit analysis.

Decision support for multi-cloud environments imposes however several challenges compared to the traditional model-based decision support. Most notably, the dynamics of multi-cloud require light-weight processes and tools, the decision makers depend on easy-to-understand representations of the impacts of the decisions, the notion of cost is to a lower degree established in the trade-off analysis of enterprise and software architectures, and a merge of the aspects of risk, cost and quality in a consolidated view imposes a new complexity as well as methodological challenges.

This paper presents the main results of our recent efforts towards the development of a DSM for multi-cloud environments. We characterize the needs for the DSM in the multi-cloud context and propose an initial version of the process for the DSM. Based on the experiences from CORAS and PREDIQT based analyses, and relying on the existing process descriptions and modeling approaches from CORAS and PREDIQT, we propose a comprehensive process for a DSM-based analysis, and present the roles of the actors/stakeholders involved. The DSM process consolidates the steps necessary towards development, verification and application of the decision support models. Based on the method proposed, we elaborate on the suitability of both the method proposed and the state of the art for analyzing risks as well as for predicting quality and cost in the multi-cloud context. We argue that many aspects of CORAS and PREDIQT, including the approaches to modeling (the modeling languages), the processes, and the respective tool support, should be well suited in the DSM context, i.e. in an analysis which merges the aspects of risk, quality and cost. However, in order to evaluate the feasibility of both the proposed DSM in general as well as the CORAS and PREDIQT methods in particular, in the multi-cloud context, realistic case studies should be performed and the proposed method adapted based on the experiences obtained.

Hence, the next steps in the development of decision support for multi-clouds should include case studies, evaluation and development of approaches to modeling (the modeling languages) for a consolidated model-based risk analysis, quality prediction and cost analysis. Moreover, the method should offer an easy-to-understand visualization of the impacts of the decision alternatives on quality, cost and risk. We also aim at refining the method and the tool requirements for DSM, as well as providing a prototype tool which will facilitate a DSM-based analysis.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 318484 (MODAClouds).

REFERENCES

- [1] R. Buyya, "Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009.

- [2] V. N. L. Franqueira, S. H. Houmb, and M. Daneva, "Using Real Option Thinking to Improve Decision Making in Security Investment," in *5th International Symposium on Information Security, LNCS 6426*. Springer, 2010, pp. 619–638.
- [3] M. Amram and N. Kulatilaka, *Real Options: Managing Strategic Investment in an Uncertain World*. Harvard Business School Press, Cambridge, Massachusetts, 1999.
- [4] M. Daneva, "Applying Real Options Thinking to Information Security in Networked Organizations. CTIT Report TR-CTIT-06-11," University of Twente, Tech. Rep., 2006.
- [5] *ISO/IEC 9126 – Software engineering – Product quality – Part 1-4*, International Organization for Standardization/International Electrotechnical Commission, 2001-2004.
- [6] M. Singhal, S. Chandrasekhar, G. Tingjian, R. Sandhu, R. Krishnan, A. Gail-Joon, and E. Bertino, "Collaboration in Multicloud computing Environments: Framework and Security Issues," *Computer*, vol. 46, no. 2, pp. 76–84, 2013.
- [7] R. Kazman, J. Asundi, and M. Klein, "Making Architecture Design Decisions: An Economic Approach. Technical report CMU/SEI-2002-TR-035," Carnegie Mellon, Tech. Rep., 2002.
- [8] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis - The CORAS Approach*. Springer, 2011.
- [9] Siemens, "CRAMM - The Total Information Security Toolkit," March 2004, accessed: January 30, 2013. [Online]. Available: <http://www.cramm.com>
- [10] C. J. Alberts and A. J. Dorofee, "OCTAVE Criteria. Technical Report CMU/SEI-2001-TR-016," CERT, Tech. Rep., 2001.
- [11] T. Peltier, *Information Security Risk Analysis, 3rd edn*. Auerbach Publications, 2010.
- [12] R. Chinnici, J. J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation," June 2007, accessed: January, 2013. [Online]. Available: <http://www.w3.org/TR/wsdl20>
- [13] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema. W3C Recommendation," August 2007, accessed: January, 2013. [Online]. Available: <http://www.w3.org/TR/sawSDL>
- [14] "Service Oriented Architecture Modeling Language (SoaML) Specification, Version 1.0," Object Management Group, Tech. Rep., 2012.
- [15] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "Reference Model for Service Oriented Architecture 1.0.," OASIS, Tech. Rep., 2006.
- [16] Cloud Services Measurement Index Consortium, "CSMIC," accessed: January 2013. [Online]. Available: <http://csmic.org>
- [17] N. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Transactions on Software Engineering*, vol. 25, pp. 675–689, 1999.
- [18] V. R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm, Technical Report TR-92-96," University of Maryland, Tech. Rep., 1992.
- [19] V. Basili, G. Caldiera, and H. Rombach, *The Goal Question Metric Approach*. Encyclopedia of Software Engineering, 1994.
- [20] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., 1998.
- [21] C. Ebert, R. Dumke, M. Bundschuh, A. Schmietendorf, and R. Dumke, *Best Practices in Software Measurement*. Springer Verlag, 2004.
- [22] A. Hammond, A. Adriaanse, E. Rodenburg, D. Bryant, and R. Woodward, "Environmental Indicators: A Systematic Approach to Measuring and Reporting on Environmental Policy Performance in the Context of Sustainable Development," World Resources Institute, Tech. Rep., 1995.
- [23] A. Omerovic, *PREDIQT: A Method for Model-based Prediction of Impacts of Architectural Design Changes on System Quality. PhD thesis*. University of Oslo, 2012.
- [24] A. Omerovic, A. Andresen, H. Grindheim, P. Myrseth, A. Refsdal, K. Stølen, and J. Ølnes, "A Feasibility Study in Model-based Prediction of Changes on System Quality. Technical report A13339," SINTEF ICT, Tech. Rep., 2010.
- [25] A. Omerovic, B. Solhaug, and K. Stølen, "Assessing Practical Usefulness and Performance of the PREDIQT Method: An industrial case study," *Information and Software Technology*, vol. 54, no. 12, pp. 1377–1395, 2012.
- [26] B. Barber and J. Davey, "The Use of the CCTA Risk Analysis and Management Methodology CRAMM in Health Information Systems," in *7th International Congress on Medical Informatics*, 1992, pp. 1589–1593.
- [27] C. J. Alberts and J. Davey, "OCTAVE Criteria Version 2.0. Technical report CMU/SEI-2001-TR-016," Carnegie Mellon University, Tech. Rep., 2004.
- [28] "IEC 61025 Fault Tree Analysis (FTA)," International Electrotechnical Commission, Tech. Rep., 1997.
- [29] "IEC 60300-3-9 Dependability Management - Part 3: Application guide - Section 9: Risk analysis of technological systems - Event Tree Analysis (ETA)," International Electrotechnical Commission, Tech. Rep., 1995.
- [30] B. Schneier, "Attack Trees: Modeling security threats," *Dr. Dobbs's Journal*, vol. 24, no. 12, pp. 21–29, 1999.
- [31] D. S. Nielsen, "The Cause/Consequence Diagram Method as Basis for Quantitative Accident Analysis. Technical report RISO-M-1374," Danish Atomic Energy Commission, Tech. Rep., 1971.
- [32] I. Ben-Gal, *Bayesian Networks*. In F. Ruggeri, R. S. Kenett, F. W. Faltin (eds.): *Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, 2007.
- [33] R. A. Howard, *Dynamic Probabilistic Systems. Volume I: Markov Models*. John Wiley & Sons, 1971.
- [34] W. Sonnenreich, J. Albanese, and B. Stout, "Return on Security Investment (ROSI)-A Practical Quantitative Model," *Journal of Research and Practice in Information Technology*, vol. 38, no. 1, pp. 45–56, 2006.
- [35] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for Architecture Evaluation. Technical report CMU/SEI-2000-TR-004," Carnegie Mellon, Tech. Rep., 2000.
- [36] S. H. Houmb, G. Georg, R. France, J. Bieman, and J. Jürjens, "Cost-benefit Trade-off Analysis Using BBN for Aspect-oriented Risk-driven Development," in *10th International Conference on Engineering of Complex Computer Systems*. IEEE Computer Society, 2005, pp. 195–204.

A DSL For Logistics Clouds

Bill Karakostas

School of Informatics, City University London
Northampton Square
London, UK
billk@soi.city.ac.uk

Takis Katsoulakos

Inlecom Ltd
Knowledge Dock Business Centre, 4-6 University Way
London, UK
takis@inlecom.com

Abstract— Cloud is a new area of specialization in the computing world, and, as such, it has not been explicitly addressed by traditional programming languages and environments. Therefore, there is a need to create Domain Specific Languages (DSLs) for it. This paper presents such a DSL that targets logistics clouds, i.e. networked resources and systems of logistics organisations. The DSL is implemented on top of the functional concurrent language Erlang and its distributed data management system Mnesia. The paper presents features of the DSL that implement commonly occurring use cases in the logistics cloud such as message exchange, document sharing and notifications. We show how program features in this DSL map to the underlying Erlang/OTP runtime.

Keywords- DSL; Logistics Cloud; Erlang/OTP; Mnesia; transport logistics; functional programming

I. INTRODUCTION

Community clouds are implementations of Clouds by a community of organisations such as logistics companies that agree to virtualise and share their computing resources. In contrast to a generic, “horizontal cloud”, components of a logistics cloud are custom tailored to the specific needs of the logistics application area [7].

Effectively, a logistics cloud is a networked data and computing infrastructure that virtualises resources (documents, data, systems and applications) for a logistics business network, to which nodes can dynamically be added and removed. Physical resources in logistics (such as cargo) are, by nature, mobile, and are handled and monitored by multiple IT systems. For cooperative processes, it is therefore important that the information about the state of logistics resources remains independent from location and physical formats of the systems that handle it. Resources and operations on them must therefore be abstracted in an implementation independent form, following the principles of Representation State Transfer (REST) [6]. This allows the participants of the logistics cloud to perform collaborative processes without concern about the physical format and location of data and applications, i.e. to work in a Cloud environment. According to the iCargo project [9], such a Cloud is a ‘parallel universe’ mirroring logistics processes, resources and data, and offering capabilities for co-operative synchronized and real-time management of transport resources (i.e. intelligent planning and controlling transport logistics chains) to optimise efficiency, quality and

environmental performance. The paper presents a Domain Specific Language (DSL) for developing cloud applications for logistics organizations.

The rest of the paper is structured as follows: Section 2 overviews Cloud DSLs and explains the rationale and design objectives for the proposed DSL. Section 3 introduces the main architectural concepts of the logistics cloud, while Section 4 presents the main features of the language. Section 5 highlights the main use cases for the DSL as investigated in the iCargo project. The last section highlights the plans for further research and development.

II. DOMAIN SPECIFIC LANGUAGES AND CLOUDS

A DSL is a programming language or an executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain [3]. DSLs have been used in many domains, particularly due to their expressiveness, runtime efficiency and reliability due to their narrow focus. More recently, DSLs for clouds have been proposed for high performance computing [2] business process management [1] and business applications [8]. Data cloud specific DSLs, such as Pig Latin from the Apache Pig project, are employed for analyzing large data sets [10].

Currently, logistics applications are implemented in general purpose languages (GPL) such as Java and C# and Web languages such as Javascript and HTML. Message exchanges are typically implemented in XML, while system interfaces are specified as Web services. However, logistics organisations and chains have become increasingly distributed and virtualised. Current development technologies fall short in realising the full potential of RESTful architectures and of the Cloud. The aim of our research has been to exploit the potential of concurrent functional languages such as Erlang [5] and distributed data management systems such as Mnesia [5] in developing logistics applications that take advantage of the Cloud’s potential. The use of Erlang to develop RESTful applications has been proposed before by S. Vinoski [13], and the potential of functional languages on the Cloud has also been advocated by J. Epstein *et al* [4]. However, the learning curve for such technologies can be steep. A DSL could help towards easing the adoption of functional concurrent languages, while maintaining their expressiveness and power in developing business applications for the Cloud.

A. Rationale for the Design of the DSL

One of the design goals was to preserve the benefits of Erlang such as the built-in, actor based, concurrency model, while easing the learning curve for the typical logistics application developer. Erlang programming has limitations such as the unconventional syntax, the lack of types, and the general lack of familiarity with functional programming styles amongst developers.

At the same time, the design of the DSL had to address an easier to read and understand syntax (i.e. by avoiding the excessive use of parentheses and brackets) and support for types. To avoid designing yet another GPL, however, only predefined types, derived from a Common Reference Model (Framework) for logistics domain were allowed. The Common Framework used was developed in EU projects such as e-Freight and iCargo [9] and provided the basis for the main domain concepts of the logistics DSL.

III. MAIN CONCEPTS

Erlang is a functional programming language used to build massively scalable soft real-time systems [5]. A distributed Erlang system consists of a number of nodes (Erlang runtime systems) communicating with each other. A node is an executing Erlang runtime system which has been given a name. Each such runtime system is called a node. The distribution mechanism is implemented using TCP/IP sockets. Mnesia is a multiuser distributed data management system written in Erlang, which is also the intended target language. In our prototype implementation, the execution of a program written in the DSL results in several spawned Erlang processes. These processes communicate with other processes across the logistics cloud, and manipulate Mnesia tables holding information about logistics resources. In a logistics cloud, the physical implementation and address of resources is virtualised. Resources are identified using logical Uniform resource identifiers (URIs) constructed from domain names of their owners and literals such as internal identifiers. Our approach assumes that logistics cloud participants have unique URIs (i.e. domain names in the Domain Name System) and all other resources acquire their unique identifiers relatively to the URIs of their owners. This avoids the need to assume (and agree upon) resource identifiers that are globally unique across the whole logistics Cloud.

We implement RESTful (PUT and GET) operations in our approach, but with functional semantics to maintain consistency with the Erlang underpinnings.

The code written in the DSL is translated with the use of a pre-processor (similar to Erlang's pre-processor) to Erlang modules that can be loaded and executed by the Erlang emulator. A program in the DSL is therefore an Erlang module containing function definitions that can be compiled and executed by the Erlang emulator. A typical execution spawns several Erlang processes. These can run on different nodes of the logistics cloud. As with standard Erlang, inter-process communication is via message exchanges.

IV. GENERAL SYNTACTIC CONVENTIONS

To reduce the learning curve, the DSL has a minimal set of constructs and relies on predefined domain types that are manipulated in a RESTful way to create and access resources. To distinguish between the DSL and regular Erlang language constructs, the former must begin with an underscore and consist of all capitals letters. Tokens that are not recognised by the pre-processor as reserved must be valid Erlang terms.

Reserved keywords fall under the categories of:

- Logistic Roles e.g. `_CONSIGNER`, `_FREIGHTFORWARDER`, `_CONSGINEE`
- Resource Types: Business documents, e.g.: `_TEP` (transport execution plan exchanged between logistics partners), administrative forms, etc. notification types such as `_DISPATCH_NOTICE`
- Resource read and modify operations using `_NEW` and `_GET` commands.
- Control Flows such as `_FOREACH` for iteratively applying a function to the members of a list
- Some Erlang data types such as lists constructors ('[]') and operators such as ! (for sending messages to processes) are also explicitly supported by the DSL.

Logistics roles are implemented as Mnesia transactional queries, while business document types are implemented as Mnesia tables and document instances as document records. This is further explained in the following section.

V. USE CASES

Below we show some typical use cases for this DSL, highlighting the syntax of the commands, the effect of the operations and an explanation of their underlying Erlang/Mnesia semantics.

A. Defining users and roles in the Logistic Cloud

Each organisations participating in the logistics cloud implements a (distributed) Erlang network node, for example the following set of nodes that correspond to 4 participating logistics organizations is defined in a logistics cloud:

```
consigner1@org1.com, consignee1@org2.com
freightforwarder1@org3.com, carrier1@org4.com
```

The above logistic cloud participants, agree, for example, to share data between them. A participant, such as freightforwarder1, may know all other participants (due to its coordinating role), and can therefore, initiate the sharing of the Mnesia database by executing the following command locally:

```
mnesia:create_schema([consigner1@org1.com,
consignee1@org2.com, freightforwarder1@org3.com,
carrier1@org4.com])
```

More participants can be added to the logistics cloud at any point, dynamically, by following this approach. Mnesia tables are automatically created for each supported resource

type on every participant node, subject to sharing declarations (explained below). A Mnesia table is a collection (more precisely, a bag) of records. Records (instances of resources) are created by participants as explained below.

B. Sharing resources amongst participants

The general syntax for explicitly sharing resources (tables) with other cloud participants is:

```
<Resource type> _SHARE_WITH <list of participants> _AS <Qualifications>
```

This results in changes to the corresponding table replication properties in the underlying Mnesia database, so that the table can be shared as read-only, read-write, and so on.

C. Creating new resources

The general syntax for creating records (instances of resources) is:

```
_NEW <resource type> _WITH <key-value list>
```

For example, to create a new arrival notice, the following command is used:

```
_NEW _ARRIVAL_NOTICE _WITH {ref="12345", status="OK"}
```

The result of the operation is to add a new arrival notice record to the Mnesia table (bag) *arrival_notice*.

The internal record definition in Mnesia is *record(arrival_notice, {ref:: string(), status:: string()})*.

Note that *_NEW* does not have to specify the location of the target database, as the record is added to the local table of the node where the command is executed and replicated according to the policies defined for that table.

D. Querying Resources

The general syntax for retrieving resource records is:

```
_GET <Resource Type> _WITH <Qualifications>
```

This returns a list of instances (records) of type 'Resource Type' that match 'Qualifications'. If the Qualifications part is omitted, all records (up to a maximum system imposed limit) are returned. This list of records can then be accessed using the *_FOREACH* operator.

Qualifications are logical expressions that specify range and other logical conditions on the properties of resources being queried.

For example, to retrieve all consignments for consigner with id *consigner1@org1.com* that have status 'dispatched', the following query can be used:

```
_GET _CONSIGNMENT _WITH {consigner = "consigner1@org1.com", status = "dispatched"}
```

Internally, the pre-processor converts queries like the above to Erlang 'list comprehension' style of queries that are then executed as Mnesia transactions.

E. Messaging

Messaging has been inspired by REST messaging approaches such as RESTMS [11].

The general syntax for messaging is

```
<Recipient List> !_MESSAGE_TYPE _WITH {key value list} _AS <Message Format>
```

Where 'Recipient List' can be the result of a query that returns the identifiers of recipients. The following code for example, sends a message (formatted as XML) containing a dispatch notice, to the owner (consigner) and the recipient (consignee) of a consignment:

```
[consigner1@org1.com, consignee1@org2.com] !_NEW _DISPATCH_NOTICE _WITH {ref= "12345", status= "dispatched"} _AS_XML
```

Additional parameters can be specified, for example, regarding the exact time the message is to be sent, how to handle errors such as no replies (timeout conditions) and so on. Internally, this is converted into message sending operations to the message listening processes of the recipient nodes. Such processes are automatically spawned when the nodes join the logistics Cloud. Messages can also be sent to recipients outside the logistics cloud by using call-back methods.

F. Event Notifications

Logistics cloud participants can publish and subscribe to events in the logistics cloud. This is often a more flexible approach than direct messaging as it decouples the senders and consumers of event notifications.

A Consigner *consigner1@org1.com*, for example, can subscribe to notifications when dispatch notices are created. The general syntax for subscriptions is:

```
_SUBSCRIBE_TO <Resource Type> _WITH <Conditions>
```

Internally, this is implemented by an Erlang process on node *consigner1@org1.com* that subscribes to update events on table *dispatch_notice*, using the command *mnesia:subscribe({table, dispatch_notice, simple})*.

If the monitoring process receives a message notification such as *{write, NewRecord, ActivityId}*, it will check that the conditions are satisfied, and if they are, the process will notify the callback process on the *consigner1@org1.com* node.

VI. CONCLUSIONS AND FUTURE WORK

Functional concurrent languages have a great potential for building the next generation of Cloud applications, due to scalability, side effect free code and ease of transformation to multiple representation formats (XML, JSON,...) of Cloud resources. Our approach is at the early stages of developing an easy to use Cloud development environment for logistics applications. We are currently investigating security features (authentication, authorization at organization and user role level) for the proposed DSL, and also support for transactional rollback and error handling both at the pre-processing stage and at runtime. We also plan to explore alternative target Cloud environments that support functional programming languages, such as Scala. After we complete the development of the pre-processor, we plan to develop a full blown transport logistic Cloud based collaborative application within the iCargo project. This application will demonstrate an implementation of the Common Framework in the DSL and the use of associated interfaces to facilitate the connection of logistics companies to the iCargo ecosystem.

ACKNOWLEDGMENT

The iCargo project "iCargo - Intelligent Cargo in Efficient and Sustainable Global Logistics Operations" is co-funded by the EU FP7 Program.

REFERENCES

- [1] B. Karlis, C. Grasmanis, A. Kalnins, S. Kozlovics, L. Lace, R. Liepins, E. Rencis, A. Sprogis, and A. Zarins. "Domain Specific Languages for Business Process Management: a Case Study". Proc. the 9th OOPSLA Workshop on Domain-Specific Modeling. 25-26 October 2009.
- [2] C. Bunch, N. Chohan, and K. Shams. "Neptune: A Domain Specific Language for Deploying HPC Software on Cloud Platforms" UCSB Technical Report #2011-02 .
- [3] A. van Deursen, P. Klint, and J. Visser. "Domain-specific languages: an annotated bibliography". SIGPLAN Not., 35(6):26--36, 2000.
- [4] J. Epstein, AP Black, and S. Peyton-Jones. "Towards Haskell in the cloud." Proc. The 4th ACM symposium on Haskell, ser. Haskell. ACM, New York, NY, USA, pp. 118–129.
- [5] Ericsson AB. "Erlang/OTP System Documentation 5.8.3" March 14 2011.
- [6] R. Fielding. "Architectural Styles and the Design of Network-based Software Architectures". Ph.D. Thesis. University of California, Irvine, 2000.
- [7] B. Holtkamp, S. Steinbuss, H. Gsell, T. Loeffeler, and U. Springer. "Towards a Logistics Cloud" Proc. 2010 Sixth International Conference on Semantics, Knowledge and Grids Beijing, China November 2010.
- [8] M. Kumar. "Domain Specific Language Based Approach for Developing Complex Cloud Computing Applications." Masters thesis. Wright State University 2011.
- [9] J. Pedersen. "Frameworks and Applications for Logistics". Proc, 2nd European Conference on ICT for Transport Logistics (ECITL), Venice 2009,
- [10] Pig Latin Basics. Available from <http://pig.apache.org/docs/r0.10.0/basic.html> [retrieved March 2013]
- [11] RESTMS. Available from <http://www.restms.org/> [retrieved March 2013]
- [12] D. Stieger, M. Farwick, B. Agreiter, and W. Messner. "DSLs to fully generate Business Applications". Available from www.jetbrains.com/mps/docs/MPSShowcase.pdf [retrieved March 2013]
- [13] S. Vinoski. "RESTful Services with Erlang and Yaws". InfoQ, March 31, 2008.

Deploying a Multipoint Control Unit in the Cloud: Opportunities and Challenges

Álvaro Alonso, Pedro Rodríguez, Joaquín Salvachúa, Javier Cerviño
 Departamento de Ingeniería de Sistemas Telemáticos
 Universidad Politécnica de Madrid
 Madrid, Spain
 {aalonsog, prodriguez, jsalvachua, jcervino}@dit.upm.es

Abstract—A Multipoint Control Unit (MCU) is a software component that manages different aspects of multimedia systems: mixing, forwarding, recording or transcoding media streams. This paper shows how Cloud infrastructures offer new opportunities to MCUs in a range of scenarios, scaling to a variable number of users. However, this deployment also implies some important challenges that need to be solved, considering the MCU functionalities and the common scenarios in which it will be used. These challenges are related to up and down scalability, geographic distribution of the users and the MCU system profiling. We provide an overview of the most effective solutions to face them and the characterization of a previously developed MCU in two videoconference scenarios. A Cloud-based MCU provides important advantages to take into account and the challenges we detected are already solved in similar environments making its deployment a promising research area.

Keywords—Cloud Computing; MCU; multimedia.

I. INTRODUCTION

Multimedia systems have gained a relevant role within software applications and services in the Internet over the recent years. Thus, we daily use multimedia applications, like video streaming or video recording. Some of these applications have strong real time requirements, such as videoconference or multiplayer online games.

In this type of applications we need to interconnect two or more users that will exchange some resources like video, audio or data. Moreover, this exchange can be made in real time. Frequently, and specially when there are more than two users, is necessary an intermediate device that manages the communication between the users and the exchange of the resources. The name of this component is Multipoint Control Unit (MCU) and its function is to coordinate the distribution of audio, video, and data streams amongst the multiple participants in a multimedia session [1].

Due to MCU's characteristics, it is possible to convert a mesh topology of connection in a star topology. This way the MCU acts as a central device forwarding the multimedia streams among the participants in the session. However, it can make some additional task that frequently reduces the compute requirements of the devices in the final user. Also it adds interesting features due to the fact of all data is going to go through the MCU, allowing several operations that can provide advanced services often requested in multiconferencing and collaborative multimedia applications:

- *Broadcasting*: this is the basic operation of the MCU, by which it sends a stream from a publisher to multiple subscribers. These subscribers receive this

stream once and the publisher only sends it once to the MCU, saving bandwidth in its network interface at the expense of the MCU, which has usually better network performance.

- *Transcoding*: the use of a more advanced MCU able to mix and transcode media streams can pave the way to solving the heterogeneity of devices and access networks. By transcoding streams into different bit-rates and sizes, the communication can be adapted to diverse network conditions and screen sizes optimizing the use of network and CPU in the clients at the expense of the MCU. This is also useful in a gateway scenario where media streams have to be translated.
- *Composing*: by generating a single video or audio stream from the available inputs, the MCU can reduce the amount of CPU overhead and control needed to participate in a multiuser multimedia system when needed.
- *Recording*: the MCU is receiving all the streams present in the session and, as stated in the previous point, is able to generate a composed stream by combining them. If a recording of the session is required, the MCU can store that stream for future reproductions.

All these features normally require a high computation level in the device where the MCU is running. The computer usually needs high level of memory and CPU power. However, these capabilities may change dynamically with the variations in the number of users or the different scenarios of the applications. The requirements of this type of devices blend very well with the Cloud Computing model because, according to the NIST definition [2], it provides characteristics like *On-demand self-service*, *Broad network access*, *Resource pooling*, *Rapid elasticity* and *Measured service*.

In the next section we analyse the opportunities and advantages that, according to these characteristics, the deployment of an MCU in the cloud offers. However, it implies important challenges that we describe in Section III, presenting also the most effective solutions to them. Finally, Section IV describes the conclusions as well as the future lines of work.

II. OPPORTUNITIES

In this section we will review the main advantages of running an MCU on Cloud Computing systems. An MCU component may require different computing characteristics depending on the number of participants, session conditions

(recording, forwarding, transcoding or composing), and the physical location of the participants.

Furthermore, these conditions of operability may vary dynamically during the session. Thus, we will demonstrate the benefits of deploying the MCU in a cloud scenario, where the session can be adapted easily and dynamically to variations on this type of conditions according to the particular requirements in each moment.

The cloud model defined by NIST and its essential characteristics illustrates these advantages and help us to better understand them:

- *On-demand self-service*: Users can provision computing capabilities (CPU, network, storage, etc.) as needed.
- *Broad network access*: Those capabilities are available over the network in different locations and are served through standard mechanisms.
- *Resource pooling*: The cloud follows a multi-tenant model, assigning resources to different users.
- *Rapid elasticity*: Capabilities can be provisioned and released automatically to scale to user demand.
- *Measured service*: Resources are automatically controlled, monitored and reported by metering systems.

Below we explain how these features provide new opportunities to MCU-based communications in these scenarios.

A. Scale to user demand

Multimedia systems offer their users the possibility of joining a conference before and during the session. They could also leave the session while it is running. Depending on the type of session this variability could be high.

A high number of users usually means more bandwidth, memory and CPU consumption. In other words, an MCU would demand more capabilities from its computing infrastructure.

In a traditional environment the provider should previously provision its own physical machines to tackle with the high peaks of demand. However, this solution implies more idle resources when the user demand is low.

In a cloud environment the multimedia provider could dynamically provision and release virtual machines on demand. This is usually done by turning on and off those virtual machines depending on the resources needed, according to the participants in the session.

This could also be achieved by dynamically increasing the performance of virtual machines. We could, for example, increase the CPU and memory capacity of a running virtual machine. We could also improve the network performance of these machines by changing their size. For example, Amazon EC2 [3] offers different network performance depending on the size of its virtual machines.

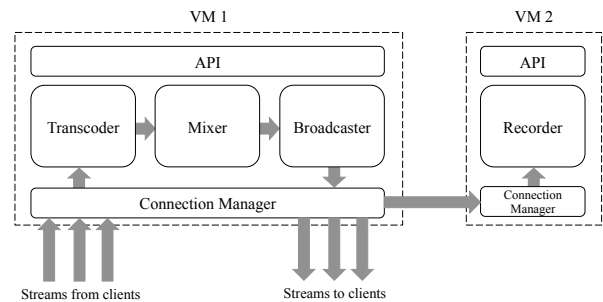


Fig. 1: Example of two MCUs performing different tasks.

B. Scale to scenario requirements

MCU operation also depends on the type of session it runs, and it could perform a variety of tasks: forwarding, recording, mixing (composing) and transcoding. Each of these features requires different computation capacities.

A basic MCU device only forwards streams from one participant to others and requires low levels of computation. However, the required level of memory and CPU increases considerably if the MCU performs the other advanced tasks. These additional features may change during a session depending on different factors: number of users, size of available and generated videos, codecs, etc.

For example, a high number of users usually forces the MCU to compose a single video from the others. Besides, in scenarios where clients connect from different type of devices, the MCU will transcode video and audio to adapt to their different CPU and bandwidth requirements. Finally, the MCU could record the entire session or part of it, including all individual videos, a subset of them, or a composed video.

Virtualized environments of cloud systems help the MCU to adapt to the varying requirements of such features. As in the previous case, we could turn on a new machine when more CPU is needed and later turn it off when this need decreases. Moreover, we could vary the capabilities of a specific virtual machine on the fly, by increasing or decreasing its memory, CPU, number of cores, etc. This would allow our MCU to adapt faster to variations on the scenario requirements.

Another workaround offered by the cloud is to configure different types of virtual machines depending on the features that they will perform. In the example in Figure 1 a machine responsible for broadcasting flows will consume a lot of CPU and memory. On the other hand, a machine responsible for recording a videoconference session consumes low memory and CPU if it receives a single flow with the whole composition of the session.

Summarizing, a cloud-hosted MCU component could easily and dynamically manage the configuration of different types of machines, adapting it to all scenarios. Thus, in a cloud environment we could provide an adaptive multimedia service, which efficiently uses the available resources, reducing costs while improving overall performance in every scenario.

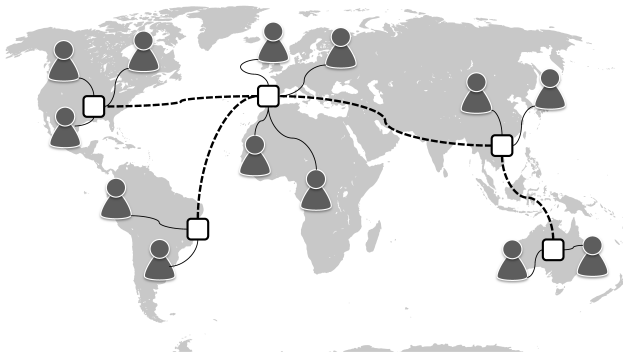


Fig. 2: Example of MCUs interconnection in a global multimedia session.

C. Geographic flexibility

Another critical factor in real time applications that directly affects user experience is the latency of packets that travel between peers. This latency usually depends on the geographical distance between them, and we should reduce it to achieve the lowest possible latency. In multimedia scenarios all streams are sent or received from the MCU and the geographic location becomes crucial for decreasing latency.

Thanks to a cloud-based system we can run MCU devices in different geographic locations, connecting each one with the users that are using the service in each region. For example, at the time of writing this paper, Amazon EC2 provides data centers in North Virginia, Oregon, North California, Ireland, Singapore, Tokyo, Sydney and São Paulo, while Rackspace [4] owns data centers at Texas, Illinois, Vancouver, Hong Kong, London and Slough, UK.

These cloud providers also allow to interconnect MCUs in different regions, so we could even offer sessions around the world, by connecting users to the closest MCUs and interconnecting all the MCUs in the same session. An example of this scenario is shown in Figure 2.

III. CHALLENGES

As seen in the previous section, the deployment of a software-based MCU in the cloud can bring several key advantages to multimedia service providers. The increased efficiency in terms of hardware requirements together with the flexibility in terms of geography and the possibility to adapt the solution to different scenarios encourages the move to the cloud. However, to make the most of the cloud and make the most of its advantages its important to design the system accordingly and take into account the target of the deployment.

Furthermore, we will propose strategies to scale up and down in the cloud that differ from more general approaches such as the one seen in [5].

This section analyses the challenges posed by the optimal adaptation of an MCU to the cloud.

A. Characterizing the system

Characterizing the MCU's performance is the first step towards its efficient deployment in the cloud. Depending on

the task (recording, transcoding...) to be performed by a given MCU, the hardware and bandwidth requirements vary significantly. By measuring the performance in a known environment, we can approximate the tier of the instance or the amount of CPU, RAM and bandwidth that is going to be needed when deploying in the cloud. For instance, transcoding needs considerably more CPU power than just forwarding packets. In order to optimise the deployment, we have to quantify this type of characteristics.

Once a complete characterization of the system is made, it is interesting to find correlations between the pure technical resources and the more high-level, application based ones. For example, in a web environment this would mean assessing the increase of CPU usage for each concurrent request of a given type. In the videoconferencing world, the number concurrent users is the typical unit that shows the capabilities when it comes to capacity of a system. Furthermore, we can group these users in different conferences that coexist in the same MCU. We will call this conferences 'rooms'. The number of users in each room is usually limited to a fixed number in videoconferencing systems.

Finding a correlation between the hardware resources needed and the number of users and rooms in a system can simplify the work we are going to do in the next subsections, scaling the system up and down. Knowing the incidence of each new user combined with continuously monitoring the resources consumed by a deployed instance we can react effectively to changes in demand. Of course, this implies measuring the incidence of each user in the cloud instances.

However, there is still a further challenge imposed by the deployment of a known system in the cloud. There is plenty of literature [6][7] on the possible interferences between different virtual machines running on the same host as well as possible ways to characterize the problem [8]. For the purpose of this paper we will assume that the deviation caused by these interferences will not be big enough to invalidate the per-user estimations.

As an example of this type of characterization we will show and comment a real case of MCU deployed in Amazon EC2. We have used an MCU designed by us for WebRTC [9] compatible systems [10][11]. For the characterization we have designed two scenarios that are the most common in videoconference systems: a real time video streaming and a multiuser videoconference. On both systems we have monitored the CPU and memory usage and the bandwidth (incoming and outgoing) consumption in the MCU computer during the experiment.

In the first scenario, live streaming, one of the clients is publishing its media stream (audio and video) in the session and subscriber clients are gradually added to view the published stream. As we can see in Figure 3 the CPU usage in the MCU increases linearly with the increase of the number of users subscribed to the streaming. This occurs because WebRTC standard uses SRTP [12] for the packet transmission implying that the MCU has to unprotect and protect each packet in order to make the retransmission from the publisher to each subscriber. We can observe that the inbound bandwidth consumption is constant during all the session and the outbound increases linearly because of for each new client connected is necessary to make a retransmission

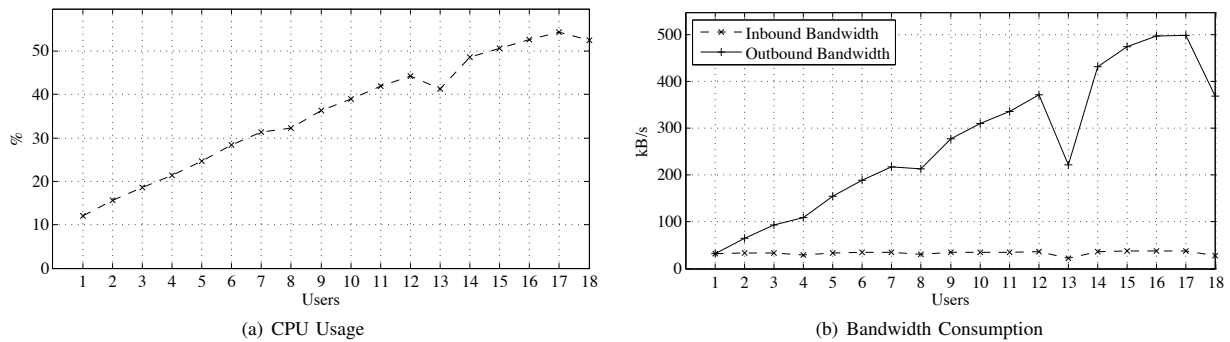


Fig. 3: Use of resources in the MCU when the number of users increases in a live streaming

more. About the memory used by the MCU it increases also linearly but with a minimum variation during the experiment (just a ten of MBytes). Finally note that in sometimes during the test a descent is registered in the results (when user number 8 and 13 connects). We can notice that this anomaly is associated with a decrease of the inbound traffic so we can deduce that it is doubt to a small bug of performance in the publishing client computer, which is also running in a Amazon EC2 virtual machine.

In the second scenario, multi user videoconference, each user that connects to the service publishes its media stream and subscribes to all the streams published previously. We have established a limit of 6 users because this is usually the maximum number of users in a standard videoconference room. In Figure 3 we can observe that in this case the inbound bandwidth consumption increases linearly with the increase of users in the room doubt to the fact of each new user publish its own stream to the room. However, the outbound bandwidth and the CPU usage increase exponentially because of for each new user the MCU has to forward the new stream to all the rest of users. Therefore the number of outgoing flows increases following the equation $N = n(n-1)$, where n is the number of users in the room. The memory usage also varies exponentially but like in the first case the variation is irrelevant.

B. Scaling up

When the currently provisioned resources reach their limit, we should able to take advantage of the cloud to keep providing service as seamlessly as possible for the users. In order to do that there are two main types of scalability: horizontal and vertical.

To scale horizontally means to add new servers to the existing pool of resources while scaling vertically is to upgrade the already running servers on the fly.

When it comes to an MCU, both methods have its uses. If the new resources are required to make some additional task in the session like, for example, recording a videoconference talk, the horizontal scalability may be a better solution. However, if the resources are needed because of an increase of the number of users in a determined session the easiest solution may be to add more capabilities to the same computer already managing that session. However, it has to be kept in mind that vertical

scalability is not present in all cloud platforms and not all operative systems allow for it.

With the exception of some very specific cases that we will explain later, the fact of have all the participants in a session in the same MCU implies facilities in the forwarding and composing of the media streams. As discussed in the next subsection this is especially critical if a scale down is necessary during a session.

So an important challenge in the case of scaling up is the decision of which type of scalability is better to choose when an increase of the resources in the MCU module is needed.

Both types of scaling involve a latency caused by, either starting a new machine or modifying the existing one. In order to have a satisfactory user experience, this has to be taken into account so no interruptions take place in the communications. This problem can be avoided by anticipating the rise in demand whenever it is possible and react accordingly.

A first approach is to use algorithms to, based on a monitoring of the system, calculate when it is in a limit situation and this way anticipate the necessity for resources starting new machines or adding more capabilities to the existing. The MCU must monitor at all times the state of the system by the analysis of the different factors studied above. If we have been able to establish the limits of the system and the correlation with the number of users and rooms it should be quite straightforward to react whenever the deployed system is reaching its peak.

The result would be a set of thresholds that would define when to add more processing power to the system. This is similar to setting elasticity rules that define the system scalability, an example of this type of approach can be seen in [13].

We can go one step further by use predictive models to anticipate the changes in the requirements of the MCU based on the analysis of previous data. With these type of models we can analyse behaviour patterns of the system to predict the activity that will be in a determined moment. These patterns may be obtained in two main ways. The most effective way is to obtain it from the previous behaviour of the own system. However, if it is not possible we can use the patterns from the behaviour of similar systems.

A good starting point is [14] where this problem is ad-

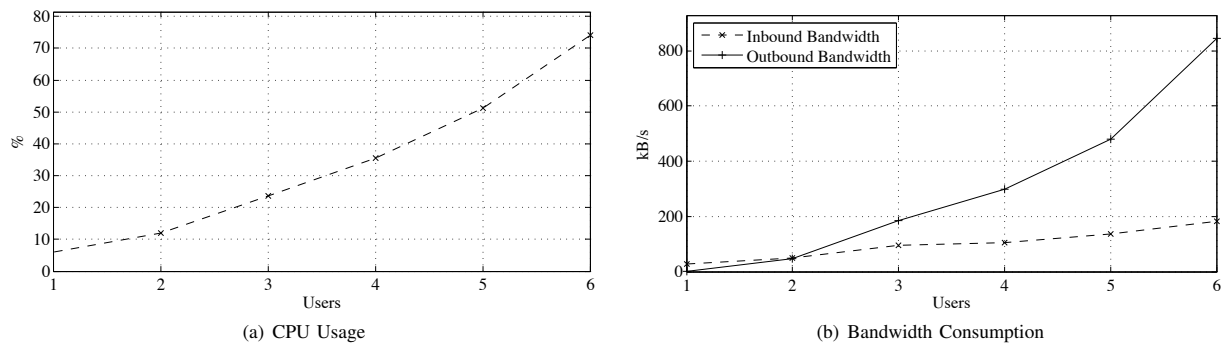


Fig. 4: Use of resources in the MCU when the number of users increases in a videoconference

dressed by an algorithm that predicts resource usage by using pattern matching.

C. Scaling down

In the same way that during a session may be necessary to increase the available resources, it may also occur that at any given time, the demand peak has ended and we have provisioned more resources than needed. As discussed in previous subsection, there are different ways to increase the computation level of an MCU module. In scale down case we can make the reverse operations to reduce the resources dedicated to the MCU.

Therefore the scaling down presents a similar challenge than scaling up. When we detect that the demand has gone down and we have allocated more resources than necessary we must select the closer to optimal way to reduce them. We can reduce the capabilities of the running computers or shut down one or more of them.

But moreover in this case the second option presents additional difficulties. It must be taken into account that the computer that we are going to turn off most probably is performing tasks that we must distribute between the rest of computers. Such redistribution is not a trivial issue.

A client participating in a session is sending and receiving several multimedia streams to and from an MCU. If this computer is going to be shut down, it is necessary to forward the traffic and it should be done in a transparent way to the user. Moreover, to optimise the use of the resources, a full redistribution of the clients and the rooms on the system may be in order.

A possible solution to this problem, shown in Figure 5, is to include a proxy between the clients and the MCU module. When a machine is going to shut down the proxy begins to duplicate the streams between the old and new MCU. When all the streams are prepared the proxy changes the sending and receiving to the client from the old MCU to the new and in this moment the old MCU can be turned off.

D. Geographic distribution

With the flexibility provided by geographically distributed cloud providers comes the challenge of optimally placing the

MCU instances in order to get the best service as possible. While the decision might be trivial when all users are located in the same continent or cloud provider's zone, deciding how to react when users are located in distant places can greatly determine the quality of the session.

When making the decision must be taking into account the number of users in each geographical region but also the quality of their connections. To characterize the links between different regions is interesting to do measures of bandwidth, jitter, packet loss or Round Trip Time (RTT). By testing the connection of each user to the different regions of the cloud we can decide where it will perform better.

As seen in [15] the network connections between Amazon instances in different regions perform better than the average internet connection. We should also take this into account when designing and deploying the system.

IV. CONCLUSIONS AND FUTURE WORK

In this paper we have analysed the main opportunities that the deployment of an MCU component offers in a cloud-based infrastructure. As we have been seen, this alternative provides interesting advantages to multimedia and real time systems with high or variable number of users because these systems usually work in scenarios in which flexibility and scalability are required. However, the deployment is not an easy task and its performance presents also important challenges that need to be solved.

We have also discussed some possible alternatives to face these challenges. The first step is to characterize the system in order to establish relationships between the number of users and the task that the MCU will realize with the technical requirements of the computers. We have presented an example of these measures in two videoconference scenarios and an overview of the existing solutions to the challenges that the scalability (up and down) presents and the geographic distribution of the MCUs.

The conclusion of our work is that the Cloud provides important advantages and that the challenges we detected are already solved in similar environments, so the deployment of MCUs in the Cloud is a promising research area. Our future work is to further analyse these solutions in multimedia scenarios and apply them to real services.

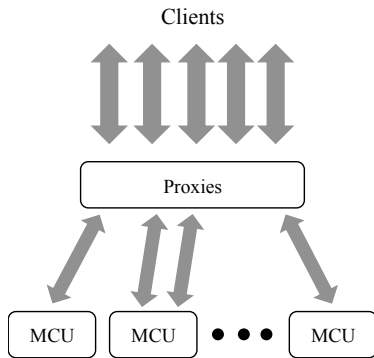


Fig. 5: Proxies forwarding traffic to MCUs.

Finally, we will apply prediction models and algorithms to our open source webRTC project, named Lynckia [16], in order to research the best way to achieve a scalable and flexible real time communication provider. We will also study the performance of the media proxy that will manage the forwarding of media streams when the systems need to scale down.

REFERENCES

[1] M. Willebeek-LeMair, D. Kandlur, and Z.-Y. Shae, "On multipoint control units for videoconferencing," in *Local Computer Networks, 1994. Proceedings., 19th Conference, 1994*, pp. 356 – 364.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf> [retrieved: March, 2013], 2009.

[3] Amazon AWS. <http://aws.amazon.com> [retrieved: March, 2013].

[4] Rackspace. <http://www.rackspace.com> [retrieved: March, 2013].

[5] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," *SIGCOMM Comput. Commun. Rev.* vol 41, num 1, January 2011, pp. 45 –52.

[6] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling virtual machine performance: challenges and approaches," *SIGMETRICS Perform. Eval. Rev.* January, 2010, vol. 37, no. 3, pp. 55 – 60.

[7] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium*, April 2007.

[8] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Zomaya, and B. B. Zhou, "Profiling applications for virtual machine placement in clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, July 2011, pp. 660 –667.

[9] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan, "WebRTC 1.0: Real-time communication between browsers," W3C, Working Draft WD, August 2012, <http://www.w3.org/TR/webrtc/> [retrieved: March, 2013].

[10] S. Loreto and S. Romano, "Real-time communications in the web: Issues, achievements, and ongoing standardization efforts, september-october, 2012," *Internet Computing, IEEE*, vol. 16, no. 5, pp. 68 –73.

[11] P. Rodríguez, J. Cervino, I. Trajkovska, and J. Salvachúa, "Advanced videoconferencing services based on webrtc," *IADIS International Conferences Web Based Communities and Social Media 2012 and Collaborative Technologies 2012*, pp. 180–184.

[12] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (srtp)," *Internet Engineering Task Force*, March 2004, updated by RFC 5506.

[13] F. Galán, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L. M. Vaquero, "Service specification in cloud environments based on extensions to open standards," in *Proceedings of the Fourth International ICST Conference on COMMunication System softWARE and middlewARE, ser. COMSWARE '09*, New York, NY, USA, 2009, pp. 19:1–19:12.

[14] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference*, December 2010, pp. 456 – 463.

[15] J. Cervino, P. Rodriguez, I. Trajkovska, A. Mozo, and J. Salvachua, "Testing a cloud provider network for hybrid p2p and cloud streaming architectures," in *Cloud Computing (CLOUD), 2011 IEEE International Conference*, July 2011, pp. 356 –363.

[16] Lynckia. Open Source WebRTC Communications Platform. <http://www.lyncxia.com> [retrieved: March, 2013].

An Approach to Assure QoS of Machine Translation System on Cloud

Pawan Kumar
Expert Software Consultants Ltd
New Delhi, India
hawahawai@gmail.com

B. D. Chaudhary
Motilal Nehru National Institute of Technology
Allahabad, India
bdc@mnnit.ac.in

Rashid Ahmad
LTRC, International Institute of Information Technology
Hyderabad, India
rashid.ahmed@research.iiit.ac.in

Mukul K Sinha
Expert Software Consultants Ltd
New Delhi, India
mukulksinha@gmail.com

Abstract—Transfer based Machine Translation (MT) System is a large complex functional application. When these MT systems are deployed with increasing translation load the Quality of Service (QoS) degrades (namely, *job completion time* increases, *system throughput* decreases, and system performance does not scale with increase in provision of resources). To improve QoS of the MT system MapReduce framework for distributed processing was explored. MT application, which has very large code size (order of 100 MB) of computation, transferring it across the data nodes of the cluster would be totally antithetical to the basic goal of throughput enhancement. To utilize the benefit of parallelism provided by Hadoop, a very large complex MT application has adopted a distinct approach to overcome this difficulty with no time penalty. This paper presents an engineering approach to delude MapReduce framework for parallelization of machine translation tasks on a large cluster of machines to assure QoS of MT system. This paper reports the initial results of the experiments done in our laboratory by running MT System under cluster of virtual machines in private cloud. Further this paper asserts that, with the availability of *elastic* computing resources in cloud environment, the *job completion time* for any translation, irrespective of its size, can be assured to be within a *fixed time limit*.

Keywords—Quality of Service; Machine Translation; Virtual Appliance; Natural Language Processing.

I. INTRODUCTION

Sampark is a machine translation (MT) system that applies transfer based approach to translate text documents among nine pairs of Indian languages [1]. Sampark system was deployed and released for public use at Sampark website for interactive as well as batch usage in 2008 [27]. The overview of this MT system comparing its transfer based approach (comprising three steps, viz., *analyze*, *transfer*, and *generate*) of machine translation to that of statistical based approach, followed by Google and Microsoft has been briefly reported in [2]. As the system was not designed a priori for scaling, its performance, with the increase in number of translation job requests, degrades sharply. Provisioning of additional computing resources, and

employing load balancer, did not improve the overall system performance incrementally. With increase in number of jobs there is either degradation, or absence of improvement in the Quality of Service (QoS) of the system, mainly in three dimensions, viz.

- Job *completion time* (solution time) increases fast
- System *throughput* decreases (number of sentences translated per unit time) and
- System performance (with provision of additional computing resources) does not scale linearly.

An MT system is a very complex application with a large code size of the order of 100 MB. It is a *functional* application where one sentence in the source language is translated into one sentence in the target language. To explain further, all the modules of a MT system produce same result given same input text, output does not depend on any *hidden information* or *state* as the program execution proceeds or between different executions of the program. An MT system treats its input text data as a list of sentences. Translation of each sentence is done independently, and has no effect either from its preceding or from following sentences. Further, it is also a compute intensive application as it takes quite a long time to translate a sentence. On an Intel(R) Core(TM)2 Quad CPU Q8300 @ 2.50GHz, L2 Cache 2048 KB, translating a sentence (average sentence size 10 words) takes approximately 3 seconds. As the compute cost is the product of number of compute resources and its utilization time to execute a job, the compute cost to translate a single sentence is 3 seconds.

An MT system like Sampark that translates a text document from a source language to a target language may have jobs that have large variance in their input data size (*workload*). On one end there may be a job to translate a single sentence, to other translating a newspaper of 30 pages, or yet another job translating a book of 500 pages. In spite of provisioning of additional computing resources, the completion time of a large job cannot be reduced. A large job does not get advantage of available and unused computing resources as a load balancer assigns each job, irrespective of size of its workload, to a distinct computing resource. This limitation caused due to the specific nature of MT

application forced us to explore the applicability of MapReduce [3] parallelization framework to reduce the completion time of large machine translation jobs.

The Map Reduce framework is suitable for functional applications as it is able to split a large job into multiple job partitions, and each job partition can run on different computing nodes in parallel. This approach of parallel execution of job partitions not only reduces a job's completion time, it also facilitates the better utilization of available computing resources. The MapReduce programming model has been designed for applications that expect provisioning of on-demand service model for computing resources. The Cloud computing platform comprising large clusters of machines provides, on-demand, availability of computing resources of desired size and number that can be scaled up/down incrementally [4].

This paper presents an engineering approach, utilizing Hadoop [5], the open source implementation of Map Reduce framework, to partition each large MT job into multiple job partitions, to run them, in parallel, on a given cluster of virtual machines provided by private Eucalyptus Cloud [6] set up in our laboratory. This parallel execution of the job partitions reduces the job completion time, and also enhances utilization of the given compute resources.

In the cloud computing environment, in addition to the reduction in job completion time, there is need to enhance the system throughput, as well. Then only it ensures the best utilization of computing resources, resulting in increase in the overall system performance, giving us the cost benefit of cloud computing environment.

The set of three experiments that we conducted show that:

- a) for a large job of any size, the job completion time can be reduced with increase in computing resources,
- b) there is an optimum job partition size (described in Section V, Experiment Two) that ensures nearly the best system throughput (i.e., number of sentences translated per unit time), and
- c) the optimum job partition size also ensures best utilization of available computing resources, resulting in completion of each job with least computation cost, assuring, in turn, very high overall system performance.

In this way, our approach assures all the three dimensions of the QoS of the MT system. MT system is an example of class of Natural Language Processing (NLP) applications that are functional in nature. This engineering approach to assure QoS can be applied to other similar applications like, text-to-speech, speech-recognition, and text-summarization, etc.

In Section II, an overview of the Map Reduce Framework is given, including its strengths and limitations while the Section III lists related works, its adaptation for various types of applications, and also for various types of platforms. In Section IV, our approach to employ Map Reduce techniques is discussed that assures the QoS for the Sampark MT system, and Section V gives the details of our experimental results. And finally, Section VI presents our conclusion.

II. HADOOP MAP-REDUCE FRAMEWORK: OVERVIEW, STRENGTHS AND LIMITATIONS

A. MapReduce: An Overview

MapReduce has become the most used parallelization framework in the data centers comprising of commodity computers [7]. MapReduce is mostly suited for functional applications, and its two functions that is *map* and *reduce* are inspired from LISP, the *functional programming* language [8].

The Hadoop Framework, the open source variant of Map Reduce, is composed of Hadoop MapReduce, and Hadoop Distributed File System (HDFS). HDFS is used to store both input data to the *map* step and the output data from the *reduce* step. A Hadoop installation is comprised of a *cluster of nodes*, consisting of a *master node*, called the *JobTracker*, and several *worker nodes*. The *JobTracker* is responsible for accepting the jobs from the clients, and splitting each job into multiple job partitions, and assigning those job partitions to be executed by different worker nodes. Each worker node runs a *TaskTracker* that executes currently assigned task to it, and on its completion, informs the same to the *JobTracker*. By communicating with each *TaskTracker*, the *JobTracker* keeps track of all the running job partitions, and also schedules of new job partitions to worker nodes that are free.

In Hadoop, the input data of a job gets distributed on the worker nodes of the cluster while it is being loaded. The Hadoop Distributed File System (HDFS) splits the input data into chunks, and each chunk is loaded on different nodes of the cluster, well before the application gets initiated.

When the JobTracker assigns a job partition to a worker node it sends the program code to that node. *It is presumed that the time spent in transferring the program code to the worker node is relatively very small in comparison to the execution time of the job partition.*

B. Strengths

The main advantage of MapReduce programming model is its simplicity. The user has to specify his algorithm as a pair of *map* and *reduce* tasks that conform to the programming model. *A functional application whose input data can be represented as a list can always be modeled in MapReduce framework.* The rest of the details, like, workload partitioning, distributed execution, network communication, coordination, and fault tolerance, etc., are all handled by the MapReduce framework itself.

This model of Map-Reduce is very efficient primarily for batch jobs, and also for those functional applications that have relatively smaller code sizes and operates on extremely large input data sizes.

C. Limitations

The intrinsic limitation of MapReduce is its *one-way scalability of its design*, i.e., to scale up to process very large data sets [9]. Again, it handles large data sets that are at rest, but is unable to handle large data in motion that can come as stream [10].

In the present implementation of MapReduce in Hadoop, the program code gets transmitted across the worker nodes of the cluster. And hence, for an application that has very large code size transferring it across the worker nodes would completely drain its job completion time enhancement due to parallel processing of its job partitions. Thus, the main limitation of Hadoop MapReduce is that it is completely unsuitable for jobs with large code size.

To utilize the benefit of parallelism provided by Hadoop, a functional application with large code size is required to evolve a distinct approach to overcome this difficulty with no transfer time penalty.

III. RELATED WORK

The MapReduce framework that was originally proposed by Google is being utilized by it to process more than 10 petabytes of data per day [3]. After the release of Hadoop implementation of the MapReduce framework more than hundred organizations, including large companies and academia are using it for various types of applications. This has also resulted intense research and development activities in various directions [11]. Some researchers have developed of many distinct MapReduce algorithms for processing of different types of massive data [12, 13], some have simulated well known parallel processing algorithm in MapReduce framework [14], while some others are involved in developing schemes for implementing MapReduce framework in distinct types of physical platforms [15, 16], and in optimizing the scheduling problem in its context [17].

The quality of output of Statistical Machine Translation (SMT) Systems increases with the increase in amount of their training data [18, 19]. Good SMT systems usually train their translation engines on 5-10 million sentences pair corpora, and to train engine on such massive volume of data, even on good processing platforms, takes couple of days to even a week. And hence, many efforts are being pursued to use MapReduce framework to execute such training module over large corpora on a large distributed systems, bringing down the training time within couple of hours [20]. Hadoop MapReduce framework has been used to study throughput improvement of SMT system [18, 19, 20, 21]. Open source toolkits capable of training phrase based SMT models on Hadoop cluster [22] and grammar based SMT on Hadoop cluster [23] have been reported.

IV. TO ASSURE QoS OF SAMPARK MT SYSTEM: AN ENGINEERING APPROACH

First, we have tried to abstract those distinguishing features of our application, viz., the transfer based MT system Sampark, that makes it an attractive application for MapReduce framework, and they are:

- A transfer based MT system is a functional application, and hence, MapReduce framework would be applicable to it,
- Any text document file that is required to be translated, i.e., data input to the MT system, can always be abstracted as a list of paragraphs, or a set of sentences of any required size, and hence, it can

be easily parallelized and executed on large cluster of machines [24],

- The incremental scaling up of computing resources on-demand is integral part of any MapReduce framework, whether it is a cluster of multi-core physical machines, or large set of virtual machines in the cloud [4]. And hence, we would be able to assure all the three dimensions of QoS (discussed in the Section I: Introduction) of MT system.

A. Hurdle: To Run Application with Large Code Size on Hadoop

The Hadoop uses strategy of *moving computation to the data site*, instead of moving the data to the computation site. This strategy allows Hadoop to achieve *high data locality* which, in turn, results in high performance.

As discussed earlier, the Sampark MT system is a very large and complex application with large code size of approx. 220 MB. This code comprises of around 100,000+ lines of code (in various programming languages), including the lexical resources, the rule base, and the machine learned data, each is of very large size, required by its various modules to perform their functionality. Transferring such a large code to each worker node would create large communication load draining completely the advantages achieved by parallel processing of job partitions.

B. Solution: Sampark MT System as a Virtual Appliance

To circumvent the above problem of transferring large code size to each worker node, the Sampark MT system is packaged as Virtual Appliance [25]. An MT virtual appliance is a full application stack containing the Just enough Operating System (JeOS), the Sampark MT system, the Hadoop system, their required dependencies, and the configuration and data files required to run the MT system. Everything is pre-integrated, pre-installed, and pre-configured to run on a virtual machine.

Whenever a new VM is provisioned from cloud, an image of the Sampark virtual appliance is actually instantiated on the new VM. For a dedicated application environment, this engineering approach completely avoids the need of transferring the MT computation code to worker nodes at run time. This technique facilitates new nodes to be added on demand.

C. Implementation: To run Sampark MT System with Large Code Size under Hadoop

To circumvent this problem for running MT System on a Hadoop, we have taken following three steps:

- We have developed a program, called *mtclient* that runs on the Hadoop *master node*. Traditional implementation of MapReduce expects data to be partitioned well before the MapReduce job is executed. This *mtclient* partitions the workload and submits the job for translation to the Hadoop master
- *mtmap* is another program that is invoked by Hadoop master for each of the workload partition. The code of *mtmap* is transported to each worker node for execution of the map tasks.

- *mtmap* in turn calls *mtmain*, which is part of the Sampark virtual appliance. *mtmain* is the main translation system that takes list of sentences as input and produces a list of sentences as translated output.

Once all the *map* tasks are over, Hadoop master calls *mtreduce* to collate the output translation. In this way, we have deluded Hadoop to run a large machine translation job as set of parallel map tasks in a dedicated application scenario.

V. THE EXPERIMENTAL SETUP, SET OF EXPERIMENTS, AND THER RESULTS

The experiment has been done on Hindi to Punjabi Sampark MT system to measure the various QoS dimensions of the system. The Sampark MT system (program codes along with lexical resources, rule bases, and machine learned data) is packaged as a virtual appliance [25]. The Sampark MT virtual appliance that we used for performing our experiments was based on CentOS-5.7 as host OS, with Xen as virtualization layer, along with Hadoop 0.20.2 as middleware for work load partitioning.

All the experiments are performed on similar virtual machines in the Eucalyptus cloud. Each of the virtual machines in the cloud are 2 CPUs, 1GB RAM with CentOS-5.3 (64-bit) as guest OS. For our experiments we had allocated 10 worker nodes in the cloud. On each worker node, the Sampark MT virtual appliance was pre-installed as a part of the setup.

We conducted three different types of experiments with different number of compute resources, and different data sets as it was required by the experiments (for experiment one 1500 sentences, for experiment two 3000 sentences, and for experiment three the data set varies from 200 to 25600 sentences). As the virtual compute resources are homogeneous in nature, and to make the data sets homogeneous in nature, we have replicated a set of 10 sentences (with average size of 8.5 words) repeatedly, to get the required size of experimental data sets.

A. Experiment One: To Investigate the Relation of Job Completion Time with respect to the Amount of Compute Resources

In this set of experiments, each experiment was done, for a given number of virtual nodes in the cloud, and with the fixed job size of 1500 sentences with increasing number of job partitions (also called *task*). The job partition sizes used in experiments are 5, 10, 15, 20, 25, 50, 75, 100, and 150 sentences each.

The same experiment was repeated with increasing the number of virtual nodes in the cloud, viz., node clusters of 2, 4, 8, and 10.

The same experiment was earlier performed on a standalone system with same virtual machine configuration in the cloud but without Hadoop.

When we have small job partition size, for a given job the number of job partitions would be large. And hence, for a given number of virtual nodes, to run all job partitions (to complete the job), it would take multiple cycles of run. *In*

comparison to a job partition (task) execution time, the inter-cycle run overhead would be negligible.

Table I shows the job completion time with increasing number of virtual nodes, and with increasing size of job partition. From this set of experiments we conclude:

- For a given job, the job completion time reduces with the increase in computing resources,
- The reduction in job completion time is linear in the beginning, but starts saturating beyond a certain point

TABLE I. SHOWING JOB COMPLETION TIME IN SECONDS FOR 1500 SENTENCES

Partition Size (Sentences per Task)	Job Completion Time (in Seconds)				
	10 Nodes	8 Nodes	4 Nodes	2 Nodes	1 Nodes*
5	258	302	583	1150	2704
10	173	215	402	798	1979
15	139	176	312	631	1704
20	137	167	285	566	1803
25	130	171	305	528	1487
50	119	134	284	433	1275
75	152	104	174	363	1193
100	151	119	211	362	1412
150	152	194	397	324	1385

* This experiment was done on a single virtual machine without Hadoop

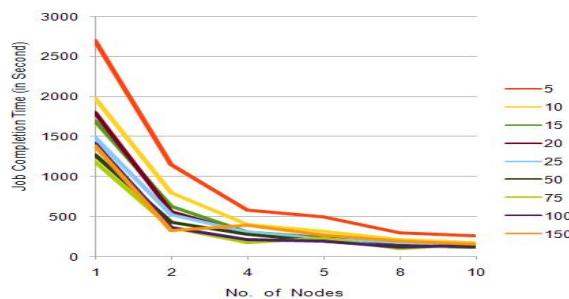


Figure 1. Job Completion time vs. No. of Nodes

B. Experiment Two: To Investigate the Relation of Job Partition Size with respect to Throughput.

In this set of experiments, we increased the size of data set to 3000 sentences, mainly to reduce the influence of inter-cycle run overhead on the throughput. Larger is the job completion time lesser would be the influence of inter-cycle run overhead. Each set of experiment the job partition sizes used were 5, 10, 15, 20, 25, 50, 75, 100, and 150 sentences each. This variation in job partition size is the same as in Experiment One.

Again, to focus our attention on throughput we have conducted only two sets of experiments on two compute resource configurations, viz., 5 and 10 virtual nodes.

Again, to focus our attention on throughput we have conducted only two sets of experiments on two compute resource configurations, viz., 5 and 10 virtual nodes.

Table II enumerates the results of the two sets of experiments. The result shows that, for a given job the best

throughput is achieved at a particular job partition size, irrespective of number of compute resources utilized. By increasing job partition size, the improvement in throughput is not very significant. *As we have reached the rim of the best throughput, we call this job partition size as the optimum job partition size.*

TABLE II. SHOWS COMPUTATION COST VS PARTITION SIZE FOR 3000 SENTENCES

No of Tasks	Partition Size (Sentences per Task)	10 Nodes	5 Nodes
600	5	35	37
300	10	49	53
200	15	59	64
150	20	63	69
120	25	68	72
60	50	86	90
40	75	93	97
30	100	82	104
20	150	101	107

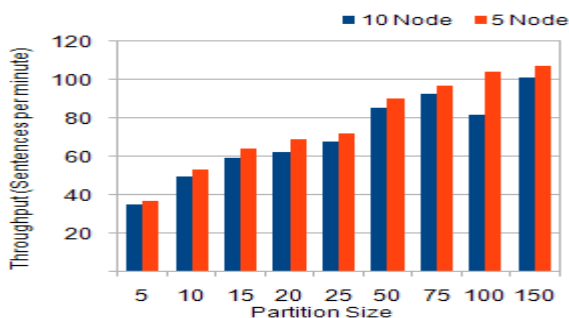


Figure 2. Throughput vs partition size of task in term of number of sentences

C. Experiment Three: To Investigate the Relation between Job Partition Size and Throughput.

In this case, we have conducted 3 sets of experiments, each with the same compute resource configuration of 5 virtual nodes.

As we are varying the job partition size to observe that where the throughput is the maximum, in each set of experiment we have maintained a fixed number of job partitions (tasks). To keep fixed number of partitions while varying the job partition size, we have to increase the job size i.e., number of sentences. The 3 sets of experiments have 40 tasks, 60 tasks and 80 tasks respectively. Figure 3 shows throughput verses partition size of task.

Table III enumerates the results of the three sets of experiments done. These results show that for a given job the best throughput is achieved at a particular job partition size. It also shows that by changing the job size (i.e., the number of sentences) hardly changes the optimum job partition size. Increasing the partition size beyond the optimum job partition size does not enhance the throughput significantly. We see that, in this range, if the partition size is doubled, the throughput increases by less than 5%.

TABLE III. SHOWS TIME TO TRANSLATE A GIVEN TASK FOR VARIOUS PARTITION SIZES ON A 5 NODE CLUSTER FOR 25600 SENTENCES

No of Tasks	Partiti on Size	Total Sentences	Total Compute Time in seconds	Throughput per minute
80	10	800	1055	45
80	20	1600	1475	65
80	40	3200	2340	82
80	50	4000	2620	92
80	80	6400	3750	102
80	100	8000	4455	108
80	160	12800	6665	115
80	200	16000	8240	117
80	320	25600	12920	119
60	10	600	800	45
60	20	1200	1025	70
60	40	2400	1715	84
60	50	3000	2005	90
60	80	4800	2830	102
60	100	6000	3320	108
60	160	9600	5055	114
60	200	12000	6140	117
60	320	19200	9990	115
40	10	400	570	42
40	20	800	855	56
40	40	1600	1165	82
40	50	2000	1355	89
40	80	3200	2115	91
40	100	4000	2275	105
40	160	6400	3435	112
40	200	8000	4175	115
40	320	12800	6430	119

TABLE IV. SHOWS THROUGHPUT VARIATIONS FOR VARIOUS PARTITION SIZES

Partition Size (Sentences per Task)	80 Task	60 Task	40 Task
10	45	45	42
20	65	70	56
40	82	84	82
50	92	90	89
80	102	102	91
100	108	108	105
160	115	114	112
200	117	117	115
320	119	115	119

VI. CONCLUSION AND FUTURE TASKS VISUALIZED

This paper presents the engineering approach that we have developed to run a functional application like MT system with a large code size as a dedicated application in MapReduce Framework, to get enhanced QoS utilizing its list homomorphism characteristics [24] for parallel

execution. This approach to assure QoS can be applied to a large group of NLP applications.

We have also developed a scheme to delude Hadoop MapReduce framework to load the MT system with large code size (by packaging MT as a virtual appliance), a priori on all worker nodes, to overcome the transfer cost at run time.

Contribution of our work is threefold:

- Completion time for any large job can be reduced with increase in computing resources,
- There exists an optimum size of job partition for which the best system throughput is achieved,
- The minimum completion time along with the best system throughput would incur the minimum compute cost in the cloud environment.

In this way, our approach assures all the three dimensions of the QoS of the MT system. In future we plan to extend this approach to other NLP applications that exhibit list homomorphism and can be partitioned for distributed computing.

ACKNOWLEDGMENT

We would like to thank Prof. Rajeev Sangal for his help to setup the experiments in their laboratory.

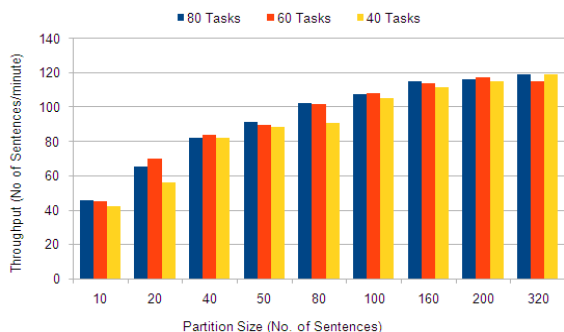


Figure 3. Throughput vs. partition size of task

REFERENCES

[1] R. Sangal, "Project proposal to develop Indian language to Indian language Machine Translation System", IIIT Hyderabad, TDIL Group, Dept. of IT, Govt. of India, (2006).

[2] G. Anthes, "Automated Translation of Indian Languages," CACM, vol. 53 (1), Jan. 2010, pp. 24-26, doi: 10.1145/1629175.1629184.

[3] J. Dean and S. Ghemawat, "MapReduce: A Flexible Data Processing Tool," In Proc. of OSDI'04: Sixth Symp. on Operating System Design and Implementation, pp. 137-149, Dec. 2004.

[4] M. Armbrust, A. Fox, R. Griffith, "Above the Clouds: A Berkeley View on Cloud Computing." Technical Report No. UCB/EECS-2009-28, Univ. of California, Berkeley, Feb. 10, 2009.

[5] The Apache Software Foundation, "Hadoop: MapReduce Framework" <http://hadoop.apache.org> [retrieved: February, 2013]

[6] D. Nurmi, et al., "The Eucalyptus Open Source Cloud Computing System," In Proc. of 9th IEEE/ACM Intl. Symp. on Cluster Computing and Grid", pp. 124-131, 2009.

[7] L. Barroso, J. Dean, and U. Hoelzle, "Web search for a planet: The Google cluster architecture," IEEE Micro, vol. 23, no. 2, pp. 22-28, 2003.

[8] R. S. Bird, "An Introduction to the Theory of Lists," Oxford University Technical Monograph PRG-S6, 1986.

[9] Z. Ma and L. Gu, "The Limitation of MapReduce: A Probing Case and a Lightweight Solution," In Proc. of the 1st Intl. Conf. on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2010). Nov. 21-26, 2010.

[10] V. Kumar, H. Andrade, Bugra Gedik, and K. Lung Wu, "DEDUCE: At the Intersection of MapReduce and Stream Processing," In Proc. of the 13th Intl. Conf. on Extending Database Technology, pp. 657-662

[11] J. Lin and C. Dyer, "Data-Intensive Text Processing with MapReduce," University of Maryland, USA, April 2010.

[12] B. He, W. Fang, Q. Luo, N.K. Govindarajan, and T. Wang "Mars: A MapReduce Frameworks on Graphics Processing," In Proc. of 17th Conf. on Parallel Architecture & Compilation Techniques, pp. 260-268, 2008.

[13] M.de Kruijf and K. Sankarlingam, "MapReduce for the Cell B.E. Architecture", University of Wisconsin, Comp. Sc., Tech. Report: CS-TR-2007-1625, 2007.

[14] H. Karloff, S. Suri, and S. Varrilvitski, "A Model of Computation for MapReduce," In Proc. of 21st Annual ACM-SIAM Symp. on Discrete Algorithm, 2010.

[15] G. Ananthnarayanan et al., "Reining in the Outliers in Map-Reduce Cluster using Manti," In Proc. of USENIX OSDI, 2010.

[16] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating MapReduce for Multi-core & Multi-processor systems," In Proc. of the IEEE 13th Intl. Sym. on High Performance Computer Architecture, pp. 13-24. Phoenix, Arizona, 2007.

[17] H. Chang et al., "Scheduling in MapReduce like System for Fast Completion Time", Bell Labs Alcatel-Lucent, Purdue University, 2011.

[18] M. Banko and E Brill, "Scaling very very Large Corpora for Natural Language Disambiguation," In Proc. of 39th Annual Meeting of Assoc. of Computational Linguistics (ACL 2001), pp. 26-33, Toulouse, France, 2001.

[19] C. Collism-Burch, C. Bannard, and J Schroeder, "Scaling Phrase-based Statistical Translation to Larger Corpora and Larger Phrases," In Proc. 43rd Annual Meeting of Assoc. of Computational Linguistics ACL, pp. 255-262, Ann Arbor, Michigan, USA, 2005.

[20] C. Dyer, A. Cordora, A. Mont, and J. Lin, "Fast, Easy & Cheap: Construction of Statistical Machine Translation Model with MapReduce," In Proc. of 3rd Workshop on Statistical MT at ACL, University of Marytal, Columns, Ohio, 2008.

[21] R.M.Yoo, A. Romano, and C. Kozyrakis, "Phoenix Rebirth: Scalable MapReduce on a Large Scale Shared Memory System," Stanford University, Computer System Laboratory, CA, USA, 2009.

[22] Q. Goa and S. Vogel, "Training Phrase-based Machine Translation Models on the Clouds: Open Source Machine Translation Toolkit Chanki," The Prague Bulletin of Mathematical Linguistics, 93: pp. 37-16, 2010.

[23] V. Ashish and A. Zollnam, "Grammar Based Statistical MT on Hadoop.An end-to-end Toolkit for Large Scale PSCFG based MT", The Prague Bulletin of Mathematical Linguistics (91), pp. 67-78, 2009.

[24] M.Cole, "Parallel Programming with List Homomaphism", Parallel Processing Letters vol. 5, No. 2, pp. 191-203, 1995.

[25] P. Kumar, R. Ahmad, B. D. Chaudhary, R. Sangal, "Machine Translation System as a Virtual Appliance: For Scalable Service Deployment on Cloud," In Proc. of IEEE 7th Intl. Symp. on Service Oriented System Engineering (SOSE 2013), pp. 304-308, 2013.

[26] R. Ahmad, et al., "Enhancing Throughput of a Machine Translation System using MapReduce Framework: An Engineering Approach," In Proc. of 9th International Conference on Natural Language Processing (ICON-2011), pp. 200-206, 2011.

[27] Sampark: Machine translation system among Indian languages, <http://sampark.org.in> [retrieved: January, 2013]

Defining Intercloud Federation Framework for Multi-provider Cloud Services Integration

Marc X. Makkes^{*†}, Canh Ngo^{*}, Yuri Demchenko^{*}, Rudolf Strijkers^{*†}, Robert Meijer^{*†}, Cees de Laat^{*}

^{*} System and Network Engineering Group
University of Amsterdam
Amsterdam, the Netherlands

[†] Information and Communication Technology
TNO
Groningen, The Netherlands

e-mail: {t.c.ngo, y.demchenko, delaat}@uva.nl {marc.makkes, rudolf.strijkers, robert.meijer}@tno.nl

Abstract—This paper presents the on-going research to define the Intercloud Federation Framework (ICFF) which is a part of the general Intercloud Architecture Framework (ICAF) proposed by the authors. ICFF attempts to address the interoperability and integration issues in provisioning on-demand multi-provider multi-domain heterogeneous cloud infrastructure services. The paper describe the major Intercloud federation scenarios that in general involves two type of federations: customer-side federation that includes federation between cloud based services and customer campus or enterprise infrastructure; and provider-side federation that is created by a group of cloud providers to outsource or broker their resources when provisioning services to customers. The proposed ICFF uses cloud resources brokering model as the main operational model in typically non-coordinated Intercloud and multi-cloud environment. The paper analyses federated identity management scenarios and related design patterns that actually creates a basis for operating federations and providing consistent federated access control infrastructure. The paper also refers to successful virtual organisation experience in Grids and attempts to re-use it in ICFF. The presented work attempts to provide an architectural model for developing Intercloud middle-ware and in the way will facilitate cloud interoperability and integration.

Index Terms—Intercloud Federations Framework; Intercloud Architecture; Cloud Computing Reference Architecture; Multi-layer Cloud Services Model.

I. INTRODUCTION

Clouds are increasingly used both by industry and by research community to outsource and/or extend their IT infrastructure. They are also used to offload the computationally intensive tasks and large data volumes, thus make them easily and globally reachable. Cloud Computing [1], [2] technologies are evolving as a common way to provide infrastructure services, resources virtualization and on-demand provisioning. In addition, they bring mobility and hardware independence to the existing distributed computing and networking applications. Despite the growth and improvement in services offered by the cloud mega-providers such as Amazon [3], Microsoft Azure [4], Google Cloud [5], Rackspace [6], an enlarging number of cloud-oriented applications and global services will require provisioning for cloud based infrastructure services involving multi-provider and multi-domain resources. They

also need to inter-connect and integrate with legacy network infrastructures and enterprise services.

Current cloud technologies development demonstrates movement on developing Intercloud models, architectures and integration tools. They support the integration of cloud infrastructures into existing enterprise and campus infrastructures, and provide a common and interoperable environment to move existing infrastructures on the cloud environment [7].

A common approach here is to use different services, resources and identities federation models. However, there is no available well-defined work to provide a common federation model for resources and services integration from multiple providers, which also allows users identities federation between home organizations and cloud service domains.

We refer to our ongoing research to define the general Intercloud Architecture Framework (ICAF) [8]–[10], that intends to address the multi-domain heterogeneous cloud based infrastructure services integration and interoperability including: integration and interoperability with the legacy IT infrastructure services. The ICAF defines the Intercloud Federation Framework (ICFF) as a framework for federating independently managed cloud and non-cloud resources and service domains together with the customer and provider identity services federation.

In this paper we propose a further definition of the ICFF components supporting to create complex projects and group oriented infrastructures on-demand provisioned across multiple providers. The research presented in this paper is based on and attempts to leverage the experience from a number of cooperative projects where the authors actively participated such as EGEE [11], GEANT3 [12] and, GEYSERS [13], that have developed federated models for Virtual Organization (VO), federated Grid resources sharing, federated access to web and network services, and combined network and IT resources provisioning by telecom services providers.

The remainder of the paper is organized as follows. Section II provides analysis of the general use cases and basic scenarios for cloud and inter-cloud federation, including short reference to the VO based federation model in Grids. Section III presents the summary of the Intercloud Architecture framework, and section IV goes into further definitions and details

of the proposed Intercloud federation framework. Section V provides information about our work to build a cloud-based test-bed for modeling and testing the proposed federation models. Section VI gives a short overview of the related works. And finally, Section VII contains conclusions and describes our further development plans.

II. GENERAL USE CASES AND BASIC SCENARIOS

A. Customer side and Provider side Federation

We define two general use cases for (1) federating cloud resources on the provider side, or (2) creating federated multi-provider infrastructures and services to deliver federated cloud services to the customer. We define the following main actors and roles adopting the Resource-Ownership-Role-Action (RORA) model proposed in [14]:

- **Cloud Service Provider (CSP)** as an entity providing cloud based services to customers, on their request and based on the business agreement that is expressed as Service Level Agreement (SLA). We need to admit specifics of business relation in clouds due to the fact that majority of cloud services are self-services and they are governed under general or individualized SLA.
- **Cloud Broker** is an entity that may play a role of the third party in offering cloud service, adding value of negotiating with many CSPs or customer groups and in some cases managing complex multi-provider services.
- **Customer** is an entity that requests cloud services. In a simple case, customer may be an end-user of the requested services, or in more general case, may be an organization (e.g. enterprise or university) requesting cloud based services for the members of their organisations and manages these services.
- **User** is an end-user consuming cloud based services. In cloud services provisioning model, an end-user may be also a customer.

Note, we do not define the broker at this stage because for the basic scenarios discussed here the broker functions can be substituted with either CSP or Identity provider (IdP) role. We will provide definition of the cloud broker role in section IV for the multi-provider Intercloud environment. Figure 1 illustrates two cases when (1) the cloud based services and/or infrastructure needs to be integrated/federated with the existing user accounts and enterprise infrastructure, or (2) cloud based public services can use external IDP and in this way already existing user accounts with the single or multiple 3rd party IDPs (such as Google+/GooglePlay, Facebook, Microsoft, or other OpenID providers).

Figure 2 illustrates the major actors and their relation in the provider side federation that is typically created between cloud service providers to share and/or outsource their cloud resources when providing a final service to the customer

B. Federated Access Control and Identity Management

Federated Identity Management (FIDM) is the main component of the federated cloud infrastructure. This issue has been recognized by industry and addressed by the OASIS Cloud TC

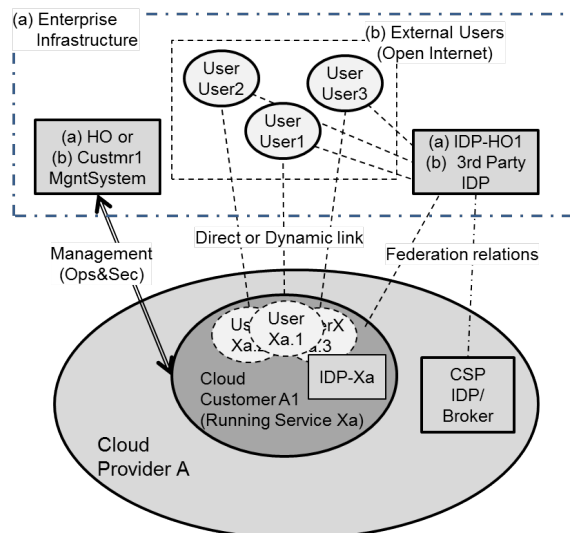


Fig. 1. Customer/user side federation for delivery of the federated cloud services to (a) enterprise customers running their own IDP and (b) for user access federation for public cloud based services.

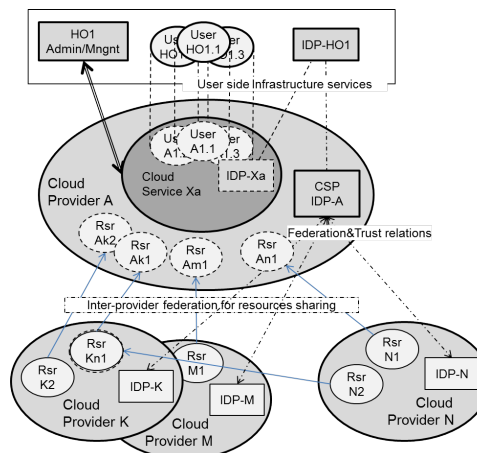


Fig. 2. Provider side federation for resources sharing and outsourcing

[15]. In the typical distributed inter-cloud infrastructure, the broker outsources the authentication and attribute management to a 3rd party IDP, either regular or cloud-aware which we will refer to as Federated IDP (FIDP). Similar to the general federation scenarios, we identify two scenarios for FIDP: a single user (actually representing individual users of the public services) and users of a customer organization (that can also be referred to as "Home Organization (HO)") that have their accounts at their HO in which their identities are confirmed by the HO-IDP.

1) *A single-end user scenario:* In this scenario, the FIDM at broker site needs to support standardized IDP protocols such as OpenID, SAML, OAuth to interoperate with public IDP, as in Figure 3.

2) *Company/organization scenarios:* When the customer is an organization or a company, there are possible IDP deployments. First, due to sensitive IdP information, some

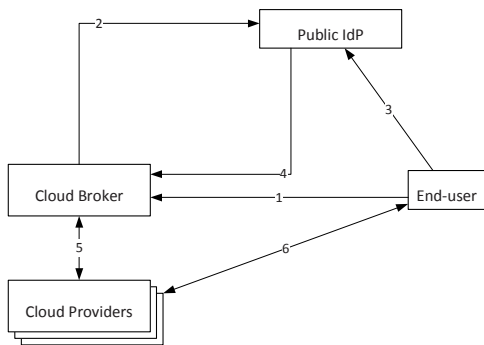


Fig. 3. Multi-provider federation with a Public IDP

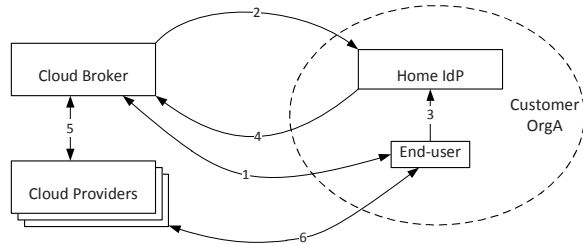


Fig. 4. Corporate customer running an on-site IDP service

organizations choose to deploy their own private IdP on-site, which need to collaborate with the FIDM Broker as in Figure 4. The vital requirement here is broker need mechanisms to discover the customer’s IDP to connect for retrieving end-users’ attributes and logon statuses.

In other scenario, a "light-weight" customer may want to out-source their identity management service to a cloud provider (i.e. IDP as a Service – IdPaaS). In this case, the IDP services are provisioned and collaborate with the FIDM cloud broker. The on-demand IDP service should support followings:

- Support service provisioning life-cycles.
- Manageable by the cloud customer for their own organization.
- Integrate with access control services for the cloud resources.

C. Policy and Security Context Management

Policy and security context management are important components of creating, operating and managing federated access control infrastructure. Authors’ previous works the XACML (eXtensible Access Control Markup Language) policy format provides all necessary functionality for multi-domain policy expression and attributes definition [16], [17]. XACML policy identification and attributes format allow for using different namespaces and attributes semantics. The proposed Generic AAA Authorisation framework [18] allows multi-domain attributes validation and mapping when evaluating access control request. Another important component in managing federated access control infrastructure is authorization session security context management what can be achieved with using tickets and tokens as session credentials. Proposed in [19], [20] authorization tickets and pilot tokens can support inter-domain

security context communication, delegation and federation management.

D. VO based Federation in Grids

The problem, which underlies the Computational Grid concept, is coordinated resource sharing and problem solving in dynamic, multi-institutional Virtual Organizations (VO). VO are defined as a collection of individuals, institutions and resources that access and share resources within the Grid [21]. Developing Intercloud federation framework we intend to re-use Grid community experience in building robust inter-organisational services, in particular using VO and a federation mechanism for managing dynamic security associations [22] The following security services and related functionalities are identified for the VO [22]:

- 1) Identity management service, normally provided by IDP.
- 2) Attribute management service that issues attributes bound to user or resource identity that primary can be used for authorization decision when accessing VO resources or services.
- 3) Authorization service to enforce access control to the resource or service based on entity’s attributes/roles and authorisation policies.
- 4) Policy management service to provide VO-wide policies related to authorisation, trust management, identity federation, mapping of identities, attributes and policies.
- 5) Trust management service that may include CA and associate PKI management services that allows establishing and managing trust relations inside VO.

In contrast to clouds, all VO services may be provided (and managed) by member organizations on behalf of the VO. Services provisioning in clouds typically includes also identity provisioning that may be linked to (or federated with) the existing user identity.

III. INTERCLOUD ARCHITECTURE FRAMEWORK

The Intercloud Architecture Framework, introduced in [8], address the interoperability and integration issues in the current and emerging heterogeneous multi-domain and multi-provider clouds that could host modern and future critical enterprise and e-Science infrastructures and applications, including integration and interoperability with legacy campus/enterprise infrastructure. The ICAF consist of the flowing components:

- 1) **Multilayer Cloud Services Model (CSM)** for vertical cloud services interaction, integration and compatibility that defines both relations between cloud service models (such as IaaS, PaaS, SaaS) and other required functional layers and components of the general cloud based services infrastructure;
- 2) **Intercloud Control and Management Plane (IC-CMP)** for Intercloud applications/infrastructure control and management, including inter-applications signaling, synchronization and session management, configuration,

monitoring, run time infrastructure optimization including VM migration, resources scaling, and jobs/objects routing;

3) **Intercloud Federation Framework (ICFF)** to allow independent clouds and related infrastructure components federation of independently managed cloud based infrastructure components belonging to different cloud providers and/or administrative domains; this should support federation at the level of services, business applications, semantics, and namespaces, assuming necessary gateway or federation services;

4) **Intercloud Operation Framework (ICOF)** which includes functionalities to support multi-provider infrastructure operation, including business workflow, SLA management and accounting. ICOF defines the basic roles, actors and their relations in sense of resources operation, management and ownership. ICOF requires support from and interacts with both ICCMP and ICFF.

The ICFF is the main framework which creates the Intercloud it self. The primary focus in the paper lies on the ICFF.

IV. ICFF DEFINITION AND REQUIREMENTS

As defined in [9], [23] the ICFF allows clouds from different administrative domains to form a federation. The federation allows for end-users to view the cloud as one, while the individual cloud providers can differentiate based on location, infrastructure and network connections to the outside world.

A. Intercloud Federation Framework.

The Intercloud federation framework is responsible for coordinating allocation of resources in a unified way. Figure 5 illustrates the main components of the federated Intercloud Architecture, specifically underlying the Intercloud gateway function (GW) that provides translation of the requests, protocols and data formats between cloud domains. At the same time the federated Intercloud infrastructure requires a number of functionalities, protocols and interfaces to support its operation:

- Trust and service brokers,
- Service Registry
- Service Discovery
- Identity provider (IdP)
- Trust manager

B. Service Broker

To overcome these shortcomings of decentralized non-coordinated allocation of resources with in multi-provider multi-domain heterogeneous cloud services, we introduce a service broker to solve allocation of resources. We identify the broker as the key component for federation, which does not have to be exclusive. The role and responsibility of the service broker is to solve the resource brokering problem. We defined as the problem as follows: "Allocation of resources and services across the multiple cloud resources such as computational clusters, parallel supercomputers, storage clusters that belong to different administrative domains".

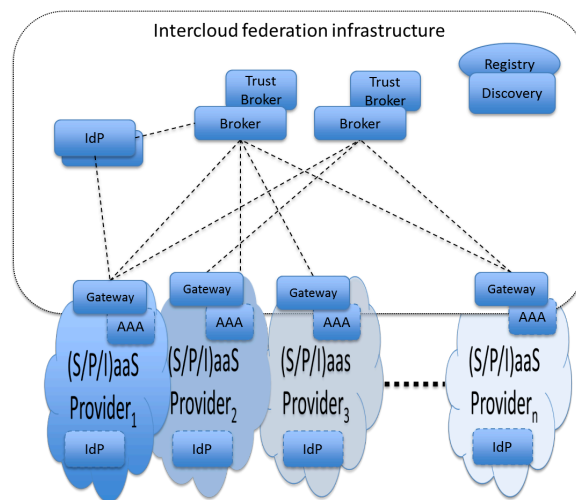


Fig. 5. Intercloud Federation Framework, where the broker has a central role for connecting to multi-cloud providers and presenting this as one Interface to the end-user. In addition, it has support for dynamical trust and IdP.

To solve the brokering problem, the service broker has interaction with both customers to allocate and de-allocation resources across multiple cloud providers on behalf of the customers. Having a broker allocate resources on behalf will simplify administration for cloud providers, as cloud provider only have to do accounting for service brokers, instead for every customer.

To have a service broker as opposed to having no brokers (such as a root directory [24]) in the federation, is to have a unified interface to all cloud providers as opposed to have different interfaces to each cloud provider in the federation. In that sense, the broker together with the cloud provider's gateway provides and ensures the interoperability between different participating clouds. Thus, the brokers provide interface for allocation of resources for their customers.

To provide identity management over moreover the brokers have interfaces to service registry, service discovery, identity provider (IdP), and trust manager, see Figure (5) for details.

C. Service registry

The service registry is a directory where cloud providers can provide information regarding IaaS, SaaS and PaaS services, which includes details of allocation of resources as well as service level agreements and policies. The broker can query Service registry information about services, and can negotiate SLA and policy with the clients. In addition, this information can be used to allocate resources in a specific cloud provider.

D. Identity Provider

ICFF operates across security domains, which are involving different cloud entities, from cloud providers to cloud consumers [2]. In this context, ICFF needs to support and integrate with the identity and trust management for these entities for both provider and customer sides.

The dynamic resource provisioning in the collaboration scenarios of cloud providers require the trust management to carry

out trust establishments between them. The trust management in the ICFF needs to support following requirements:

- Dynamic trust establishment between indirect cloud entities: Current relationships between cloud entities often rely on SLAs, which are mostly suitable for direct relationships. ICFF scenarios require a cloud provider or cloud consumer could connect to other unknown entities, through a chain of direct SLA relationships, which is known as dynamic trust relationship [25].
- Interoperate and extend standardized mechanisms on multi-domain identity management and trust management, which are SAML [26], OAuth [27] to support on-demand provisioned clouds.
- A fine-grained trust management policy language.

ICFF should take into account federated identity management in its operation management:

- Compatible with existing public identity management systems.
- Interoperate between identity management with the on-demand access control services to manage cloud resources.

E. Grid vs Cloud Federation

The main idea behind cloud computing is that infrastructure that is not used, is rented to third parties. This includes storage, computational, and services in an on-demand and pay-as-you-go model. Except for the on-demand and pay-as-you-go model, the ideas of grids grid are not quite different. Grid federation is based on institutions that want to cooperate, such that users, can access computational resources quicker. The hierarchy is mostly flat, with a 'super scheduler' to schedule all jobs on the combined resources using queue's. To scale vertically, i.e. creating hierarchy can only be done with software such as [28]. Clouds on the other hand, are mostly providing services to their customers, and have competition on the market. Horizontal scaling and federation can both be done with brokering. In addition, brokering allows for hierarchical scaling as a broker of broker can be created. Clouds provide a services oriented model, such as IaaS, PaaS and SaaS. Together with brokering, this allow independent clouds and related infrastructure components federation of independently managed cloud based infrastructure components belonging to different cloud providers and/or administrative domains; this should support federation at the level of services, business applications, semantics, and namespaces, assuming necessary gateway or federation services.

The vital difference between grids and clouds is that the amount of computation is mostly unknown with clouds, hence it is mainly used for running services while grids are to run predefined computational jobs with budgets. While grids can be run on clouds using grid software [22] the other way around is not trivial task. In addition, clouds are mostly used workloads that are not pre-defined, such as services, while grids run mostly budget or time constrained computation jobs.

V. CLOUD FEDERATION MODELING

This section provides short overview of the test-bed that we used for modeling overlay network and which we are redesigning to support modeling of the basic federation models in provisioning federated cloud resources. The test-bed consists of a Broker, which connects users and Different cloud providers, which includes Amazon AWS and Brightbox, with each other and is such a way that users can create VM (IaaS) over multiple provide. The broker provides an interface to OpenID IdP provided by google [27] to provide accounting, authentication, and authentication. The test-bed provides an interface to the end- users such that they can instantiate a layer 2 overlay network using VPN's. The interface provides also addressing IPv4 and IPv6 for created IaaS nodes in an automated fashion. After the overlay network is created and addressing is assigned, the interface provides an option to enable IPv4 or IPv6 routing based on Quagga [29]. This allows uses to create on-demand overlay network in multi provider cloud environments. The authors believe that at the time of conference the proposed test-bed will collect valuable information to estimate performance of the basic federation use-cases when realized with the AWS infrastructure.

VI. RELATED WORK

Federations of computational resources come in different forms, but one federation that's on large scale is grid computing. The problems of federation in Grid computing shows many resemblance with cloud computing.

The main idea behind grid computing is to use computation and storage resources for other computational goals if they are not used. This idea was then fully extended to multiple locations, multiple administrative domains, different architectures, etc., and link together with software. In Grid computing, the federation problem The Grid resource brokering, also know as super-scheduling, problem is defined as: " scheduling jobs across the grid resources such as computational clusters, parallel supercomputers, desktop machines that belong to different administrative domains". Brokering in computational grids is facilitated by specialized application schedulers such as Nimrod-G [30], Condor-G [31], AppLeS [32], APST [33] Legion and WorkFlow Engines. Grid Brokering activity involves:

- Querying grid resource information services (GRIS) for locating resources that match the job requirements,
- Coordinating and negotiating Service Level Agreements;
- and job scheduling.

The grid resources are managed by their local resource management systems such as Condor. These systems manage job queues, initiate and monitor their execution.

VII. FUTURE DEVELOPMENT

The paper presents an on-going research at the University of Amsterdam to develop the Intercloud Architecture (ICA) addresses the problem of multi-domain heterogeneous Cloud based applications integration and inter-provider and inter-platform interoperability. The presented research is planned

to be contributed to the Open Grid Forum Research Group on Infrastructure services On-Demand provisioning (ISOD-RG) [27], where the authors play active role. In addition, we planned to extent our test-bed, in such way that it enables dynamic provisioning of federation infrastructure.

VIII. ACKNOWLEDGMENTS

This project is supported by the Dutch national research program COMMIT and the FP7 EU funded Integrated projects The Generalized Architecture for Dynamic Infrastructure Services (GEYSERS, FP7-ICT-248657), GEANT3 (FP7-ICT-238875).

REFERENCES

- [1] "NIST SP 800-145, "A NIST definition of cloud computing"." <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>, accessed February 2013.
- [2] "NIST SP 500-292, Cloud Computing Reference Architecture, v1.0." http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505, accessed February 2013.
- [3] "Amazone Web Services." <http://aws.amazon.com/products/>, accessed February 2013.
- [4] "Microsoft Windows Azure." <http://www.windowsazure.com/>, accessed February 2013.
- [5] "Google Cloud Platform." <https://cloud.google.com/>, accessed February 2013.
- [6] "Rackspace Cloud." <http://www.rackspace.com/cloud/>, accessed February 2013.
- [7] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," *Algorithms and architectures for parallel processing*, pp. 13–31, 2010.
- [8] Y. Demchenko, C. Ngo, M. Makkes, R. Strijkers, and C. de Laat, "Defining inter-cloud architecture for interoperability and integration," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 174–180, 2012.
- [9] Y. Demchenko, C. Ngo, C. de Laat, J. Garcia-Espin, S. Figuerola, J. Rodriguez, L. Contreras, G. Landi, and N. Ciulli, "Intercloud architecture framework for heterogeneous cloud based infrastructure services provisioning on-demand," 2013.
- [10] B. Khasnabish, "Cloud reference framework." draft-khasnabish-cloud-reference-framework-04.txt, 2012.
- [11] "European Grid Infrastructure (EGI)." <http://www.egi.eu/about/EGI.eu/>, accessed February 2013.
- [12] "Geant project." <http://www.geant.net/pages/home.aspx>, accessed February 2013.
- [13] "Generalised Architecture for Dynamic Infrastructure Services (GEYSERS Project)." <http://www.geysers.eu/>, accessed February 2013.
- [14] J. Garcia-Espin, J. Riera, S. Figuerola, and E. Lopez, "A multi-tenancy model based on resource capabilities and ownership for infrastructure management," 2012.
- [15] "OASIS IDCloud TC: OASIS Identity in the Cloud TC.." <http://wiki.oasis-open.org/id-cloud/>, accessed February 2013.
- [16] Y. Demchenko, M. Cristea, and C. de Laat, "XACML policy profile for multidomain network resource provisioning and supporting authorisation infrastructure," in *Policies for Distributed Systems and Networks, 2009. POLICY 2009. IEEE International Symposium on*, pp. 98–101, IEEE, 2009.
- [17] G. Garzoglio, I. Alderman, M. Altunay, R. Ananthkrishnan, J. Bester, K. Chadwick, V. Ciaschini, Y. Demchenko, A. Ferraro, A. Forti, *et al.*, "Definition and implementation of a saml-xacml profile for authorization interoperability across grid middleware in osg and egee," *Journal of Grid Computing*, vol. 7, no. 3, pp. 297–307, 2009.
- [18] Y. Demchenko, A. Wan, M. Cristea, and C. De Laat, "Authorisation infrastructure for on-demand network resource provisioning," in *Grid Computing, 2008 9th IEEE/ACM International Conference on*, pp. 95–103, IEEE, 2008.
- [19] L. Gommans, L. Xu, Y. Demchenko, A. Wan, M. Cristea, R. Meijer, and C. De Laat, "Multi-domain lightpath authorization, using tokens," *Future Generation Computer Systems*, vol. 25, no. 2, pp. 153–160, 2009.
- [20] Y. Demchenko, O. Koeroo, C. de Laat, and H. Sagehaug, "Extending XACML authorisation model to support policy obligations handling in distributed application," in *Proceedings of the 6th international workshop on Middleware for grid computing*, p. 5, ACM, 2008.
- [21] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International journal of high performance computing applications*, vol. 15, no. 3, pp. 200–222, 2001.
- [22] Y. Demchenko, "Virtual organisations in computer grids and identity management," *Information Security Technical Report*, vol. 9, no. 1, pp. 59–76, 2004.
- [23] Y. Demchenko, M. Makkes, R. Strijkers, and C. de Laat, "Intercloud architecture for interoperability and integration," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pp. 666–674, Dec. 2012.
- [24] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud - protocols and formats for cloud computing interoperability," in *ICIW (M. Perry, H. Sasaki, M. Ehmann, G. O. Bellot, and O. Dini, eds.)*, pp. 328–336, IEEE Computer Society, 2009.
- [25] C. Ngo, Y. Demchenko, and C. de Laat, "Toward a dynamic trust establishment approach for multi-provider intercloud environment," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pp. 532–538, IEEE, 2012.
- [26] S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and protocols for the oasis security assertion markup language," *OASIS Standard (March 2005)*, 2005.
- [27] "Using OAuth 2.0 to Access Google APIs." <https://developers.google.com/accounts/docs/OAuth2>, accessed February 2013.
- [28] P. Andreetto, S. Borgia, A. Dorigo, A. Gianelle, M. Marzolla, M. Mordacchini, M. Sgaravatto, F. Dvorák, D. Kouril, A. Krenek, *et al.*, "CREAM: a simple, grid-accessible, job management system for local computational resources," *CHEP 2006, Mumbai, India*, 2006.
- [29] "GNU quagga routing software." <http://www.quagga.net/>, accessed February 2013.
- [30] A. Natrajan, M. Humphrey, and A. Grimshaw, "Grid resource management in legion," *INTERNATIONAL SERIES IN OPERATIONS RESEARCH AND MANAGEMENT SCIENCE*, pp. 145–160, 2003.
- [31] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-g: A computation management agent for multi-institutional grids," *Cluster Computing*, vol. 5, no. 3, pp. 237–246, 2002.
- [32] F. Berman, "High-performance schedulers," *The grid: blueprint for a new computing infrastructure*, vol. 67, pp. 279–309, 1999.
- [33] Y. Yang, K. van der Raadt, and H. Casanova, "Multi-round algorithms for scheduling divisible loads," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 11, pp. 1092–1102, 2005.

A Cloud Platform to support User-Provided Mobile Services

Vincenzo Catania, Giuseppe La Torre, Salvatore Monteleone and Daniela Panno
University of Catania - Italy
first.last@dieci.unict.it

Abstract—The rapid evolution of mobile computing, together with the spread of social networks is increasingly moving the role of users from information and services consumers to actual producers. Currently, since most of the critical aspects related to user generated contents have been addressed, the main issues related to service generation represent the next challenge. Dealing with services, aspects like ease of creation, discovery, security and management should always be taken into account. To cope with this kind of problems, we propose the *webinos* platform, which is based on a cloud architecture and enables user devices to share features and services among them.

Keywords—*webinos, cloud, user-provided mobile services*

I. INTRODUCTION

The growing popularity of Internet-enabled devices and the consolidation of social networks have increased the amount of multimedia contents generated by users. Everyday people live a second life on social networks generating original contents such as pictures, videos, comments and so on [1]. Table I contains some statistics about user content generation.

TABLE I. STATISTICS RELATED TO USER-GENERATED CONTENTS

Average amount of tweets per day	190 million
Average pictures uploaded to Flickr per minute	3000
Total amount of articles hosted by Wikipedia	17 million
Total pieces of content shared on Facebook each month	70 billion

This phenomenon has been encouraged by the spread of many kinds of Internet-enabled devices such as smartphones, tablets and entertainment devices.

Shipments of Internet-enabled devices are projected to hit 503.6 million units in 2013, up from 161 million in 2010. By 2015, however, shipments of Internet-enabled consumer devices are projected to break three-quarters of a billion units - at 780.8 million units - exceeding PC shipments of 479.1 million units [2]. Mobile devices give a new experience to users, offering them the possibility to obtain information about the surrounding environment through several built-in sensors (GPS, accelerometer, gyroscope). All these information let users create context-related contents, like geolocalized photos or tweets, which embed current user's position. A key role in this scenario is played by end-users, which are becoming the main contributors of the contents available on the web. The most likely next step in this direction will be the generation of services by non-expert users. Generating new service implies the creation of a set of API to interact with the service itself. According to the Service Oriented Architecture (SOA) paradigm, a new service could also be generated by composing one or more existing services. The result of this operation is

commonly referred as “mashup”. In this paper, we want to emphasize that in a not too distant future, services will be not only generated but also provided by users, primarily through mobile terminals. In particular, we refer to common users who do not have an advanced computer knowledge. A series of both software and hardware resources are necessary in order to support the user in generating and providing a service, especially if this is provided by means of a mobile device. Devices such as smartphones or tablets have peculiar characteristics due to their portability and small size. Battery life, reception problems, reduced computational and storage resources are just an example of the limitations which characterize this kind of devices. In addition, issues related to the publication of a new service, its discovery, privacy and access control raise the need of a platform to support the user in the generation and supply of services through mobile devices. In this paper, we describe *webinos*, a cloud platform for running applications and services over heterogeneous devices belonging to different domains. In the following, we will show how *webinos* can be adopted to solve typical problems of generation and supply of mobile services.

II. USER-PROVIDED MOBILE SERVICES

The aim of this section is to explain what is meant by mobile services and then outline the main issues that there are when this kind of services is provided by users through their devices. We have already said that users are increasingly involved in the generation of multimedia web content. The role of users gains even more and more importance also in the field of service generation. The emergence of Services Oriented Computing (SOC) allows end-users to develop applications by composing existing services. In this context, tools such as Yahoo Pipes [3] provide users the possibility to create own mashups composing web services. As a result, the Web is rapidly progressing towards a highly programmable platform and end-user programming has become a very popular and common trend nowadays. This enables end-users to take advantage of different Application Programming Interfaces (APIs) to create and publish their own contents and services. Major companies like Facebook, Google and eBay have already provided interfaces to their services extending their market possibilities. In this article, we focus on those services generated by users based on other applications or services provided by other mobile devices.

Mobile services are those services designed to be accessed through mobile devices. Their main aspect is the mobility for what concerns both their invocation and their supply. The difference with traditional services is remarkable: a service that

allows a user to view bus timetables can be provided through a web site and can be accessed in the same way on a personal computer or on a smartphone. The same service designed to be used on the move will take into account the user's context. For example the mobile service could give information for only those buses which route is close to the user's position that can be obtained through smartphone's GPS.

The potentialities of mobile services are huge. To date, there are already many context-aware applications for smartphones allowing users to benefit from mobile services. Considering the evolution of user's role from consumer to producer of content and services, is presumably that in the next few years, the average user will be able to create applications for his smartphone making a mashup of services also offered by other devices. As an example, suppose that the mobile phone owned by an elderly person provides the ability to be managed remotely. In this way, using this "device ability" a more experienced user could help the elderly to perform operations such as the remote phonebook's management.

There are several issues to consider in the creation and sharing of services across multiple devices. In particular, there would be the need of:

- A protocol to describe services and their exposed features.
- An access control mechanism to specify, through policies, the access / composition constraints of each service.
- Hosting environments (service providers) where to run services.
- Repositories where services have to be registered.
- A discovery mechanism to retrieve services (eg. by exposed features).
- A toolkit to help users to create, deploy and manage services.

In the next sections, we will give an overview of the state of the art in the field of user-generated mobile services. We will also present the *webinos* platform and how it can help to satisfy the aforementioned requirements.

III. RELATED WORK

The scientific interest about User Generated Service (UGS) and User Generated Content (UGC) fields is growing in these last years. Zhao et al. present in [4] a comprehensive survey of current state of art in UGSs. They give the specific description of UGS by comparison with the concept of UGC, and then go through different technologies to analyze the challenges of UGS describing advantages and limitations of each approach. Jensen et al. describe in [5] some guidelines to support users creation and management of services. Tacken et al. investigate in [6] the state of the art and the requirements to let the vision of the super prosumer concept become true. They review the current technologies for an easy creation and discovery of mobile services and list the identified requirements for user generated mobile services. In [7], authors discuss the concept of mobile-services generated by the user itself. They investigate some conceptual requirements and concluded with

an architecture proposal for IT service providers. Authors also provide a proof-of-concept system development performed within the European-funded project *m:Ciudad*. The European FP7 research project *m:Ciudad - a metropolis of ubiquitous services* - aims at the empowerment of users to create services on mobile terminals. The project demonstrates various scenarios in which users either act as creator of services or interact with the system to search for services or service construction components. *m:Ciudad* envisions a system for service providers, which enables a mobile user to create and consume mobile services on the fly on his mobile device. *m:Ciudad* architecture is exhaustively described in [8]. In the next section, we are going to introduce another European funded project called *webinos*. In particular, we are going to describe how *webinos* can be adopted as a platform to allow mobile service to be created and shared among users. The main advantages of *webinos* compared to other platforms will be discussed.

IV. WEBINOS

Webinos[9] is an Open Source Cross-Device Platform for widgets and mobile/web applications that allows developers to write applications able to run on multiple devices belonging to different domains (mobile devices, TV and automotive). In fact, the main goals of the project are applications' interoperability across devices and usability in order to create a multi-device user experience based on data synchronization and context-awareness taking into account the related security aspects.

Webinos provides a web runtime extension for browsers, which supports widget and web applications written with standard web technologies such as HTML, CSS and Javascript. *webinos* further provides a set of device-specific Javascript APIs to

- Provide access to hardware and software capabilities offered by a device such as address book, telephony manager, messaging manager, information about device status and so on.
- Access to capabilities on remote devices inter or intra Personal Zones.

The first characteristic allows developers to interact with the device, for example sending an SMS or getting geolocation and contacts information using the set of Javascript APIs. The second characteristic represents the most innovative contribution of *webinos* and allows applications running on a device to use APIs provided as services by other devices. This mechanism will be further described in the rest of this section along with a comprehensive description of the *webinos* architecture. *Webinos* introduces the concept of *Personal Zone* (PZ), defined as the set of all devices owned by a user. Each PZ has a main component called *Personal Zone Hub* (PZH), which is the point where the devices are registered and also provides data synchronization, communication among other PZs and secure access to the PZ from Internet. Multiple PZHs, one for each user, may also be linked together creating relationships among users as it happens in social networks. Figure 1 describes the overall *webinos* architecture.

Each *webinos*-enabled device placed inside a PZ has two main components called *Personal Zone Proxy* (PZP) and

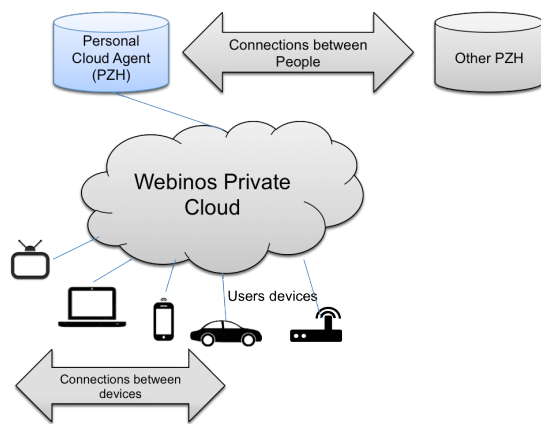


Fig. 1. An overview of the *webinos* architecture

webinos runtime (WRT). The WRT represents the environment where the apps are executed. *webinos* provides two kinds of WRTs: the first is a browser extension for the execution of web applications, the second is a widget runtime for the execution of locally stored applications (widgets). *Webinos* provides a WRT version for each the considered domains (mobile, PC, in-car units and home media), this means that the same application may run over all these domains without the need of a code refactoring.

The PZP connects the device to the PZH and enables the communication among devices inside the same PZ and exposes the *webinos* APIs. WRT and PZP act respectively as browser and local server, allowing each device to communicate with each other passing through the PZH (canonical way) or through a direct communication PZP-to-PZP in those situations where an Internet connectivity is not available. Also devices belonging to different PZs can communicate if their PZHs are connected. The PZH is responsible to issue identities (through PKI mechanism) and acts as messaging hub for devices and as a synchronization agent for data. User's data and services can be shared securely with other people connecting together multiple PZHs using a permission-based infrastructure. Both PZP and PZH represent the main components of *webinos* cloud architecture. Each user's content, such as an address book's contact, a calendar's event and so on, could be synchronized in every devices belonging to the user. Contents thus, are not related to a single devices but they are stored in the cloud. Although this concept is not too distant from Apple's iCloud, the most significant innovation provided by *webinos* is the possibility to share not only contents among devices but also services. In such way, devices belonging to different domains, with different OSs and produced by different manufacturers could seamlessly interoperate with each others.

Using *webinos*, users get all the benefits of a cloud platform with also the possibility to ensure privacy for their contents: *Webinos* also provides to each user the possibility to get all the benefits of a cloud platform *Webinos* provides users with all the benefits of a cloud platform offering also the possibility to ensure privacy for their contents by setting up a PZH in a private device. Figure 2 shows a detailed representation of PZP and WRT modules placed inside each *webinos* enabled device.

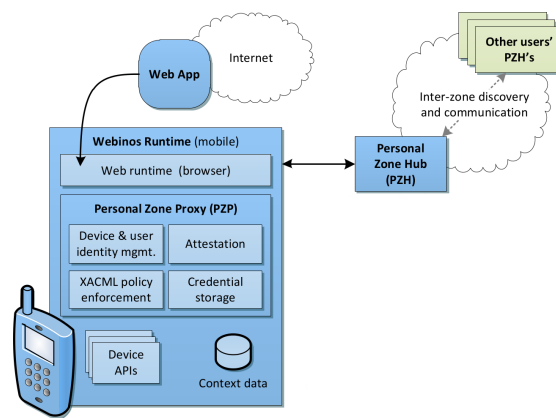


Fig. 2. Personal Zone Proxy and *webinos* runtime

Other components inside PZH and PZP, called managers, are responsible for authentication, policy management, context handling, messaging, etc.

The main characteristic, which differentiates *webinos* from other apparently similar platforms such as Phoneygap or Titanium or even respect mobile operating systems like Android or iOS, is the possibility to consider each API as a service provided by the device. As a consequence of this approach it is possible to create applications by invoking API on devices different from the one where the application is executed.

One of the demos presented in the *webinos* context, which mainly stands out the potentiality offered by the platform, is the *webinos Travel* application [10]. It enables user to manage his point-of-interests while a user is traveling. POIs are automatically synced between the user's devices. There is no 3rd party server integrated, where the information is stored. Syncing mechanism of the app is based on the *webinos* personal zone middleware. All data is owned by the user and resides inside zone. The application enables the interaction with the in-car navigation system. POIs can be pushed for guidance to the in-car navigation software. When the vehicle is parked, the smartphone can pick up the guidance.

V. *Webinos* AS A PLATFORM FOR USER-PROVIDED MOBILE SERVICES

Webinos introduces new scenarios for the generation and sharing of mobile services. Figure 3 shows a use-case where user has registered a personal computer and a car inside his PZ. Each of these devices has a PZP, which implements and exposes the *webinos* geolocation API. In the case of the example, a user is watching his car's position through an application running on his PC, which uses the geolocation API provided by the car. Thus, each *webinos* API implemented by a PZP can be considered as a service provided by a device. The PZP then turns each device in a server able to accept requests from other devices

Webinos provides both the mechanism for dynamic registration of new services and for discovering these services by searching the devices able to provide them. For example when a new device is added to a user's personal zone, the PZH registers all the services exposed by this new device and makes them discoverable, or not, according to the security policy set

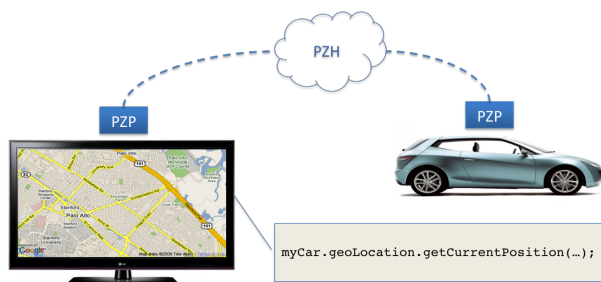


Fig. 3. An example of using “API as service”

by the user. All services provided by devices registered inside a PZ could be retrieved using the *webinos.discovery* API. We have said in the previous section that *webinos* provides the possibility to connect each other multiple PZH. Each PZH represents a user and his devices. Linking together multiple PZH means that when a user search for a service (for example the geolocation service) his PZH will query not only devices inside his PZ but also those devices belonging to linked PZs. *M:Ciudad* project considers only user generated services provided by smartphones, *webinos* instead takes into account different domains such as automotive, home-media devices and even smart objects belonging to the domain of Internet of Things. Especially in the case in which more PZHs are mutually connected, a mechanism for controlling access to services is of fundamental importance. Each PZP in fact, has an access control module based on XACML [11] specifications, which checks whether the request from an external device to a certain API may or may not take place.

Besides the possibility of calling APIs as services provided by other devices, *webinos* offers the possibility to create applications that can communicate with other applications installed on different devices. The *webinos* App2App messaging API specification defines interfaces to create, send and receive messages between applications in the *webinos* system. It provides generic messaging primitives, which can be applied in different application scenarios. The messaging is indirect, meaning that applications do not directly address each other but use a channel to route the messages to connected applications. A unique namespace (within a PZ) is used as a key to find and connect to channels. This API can be used by third-party application developers to implement custom message-based protocols by taking advantage of the features offered by the *webinos* message handling system and overlay networking model. The App2App API represents a starting point to allow the creation of new applications in the form of services, realized as a mashup of existing other services.

The possibility offered by *webinos* application to call an API exposed by another device may give rise to some problems of content management. Suppose that an application running on Alice’s tablet was able to access the *webinos* Contacts API provided by Bob’s smartphone to read and save locally Bob’s contacts. In this case, which assumes that Bob had given access control rights to Alice, privacy concerns may arise if a third person, such as Carol, uses the Contacts API provided by Alice’s tablet to read Bob’s contacts.

Our future work will be exploiting the potential of *webinos* and in particular of the App2App API in order to make it

possible for users to create and share *webinos* services obtained from the composition of services provided by multiple devices. In particular, we would like to

- Extend the registration and discovery mechanism to ensure that each new service created is associated with semantic information.
- Extend the current security mechanism in order to solve problems related to data handling and privacy of contents.

VI. CONCLUSIONS

In this paper, we have highlighted the metamorphosis of user’s role from a simple consumer to a producer of contents and services in the Web. We described what is meant by mobile services and the problems that may arise when those services are provided through a mobile device. We also described *webinos*: a European project still that aims to define a platform for the development of user-centric applications for cross-domain targets (mobile, PC, in-car units and home media). We envision that, if properly extended, *webinos* can become the reference platform for the generation and sharing of services through users’ devices.

ACKNOWLEDGMENTS

The research described in this paper was funded by the EU FP7 *webinos* Project (FP7-ICT-2009-5 Objective 1.2).

REFERENCES

- [1] Statistic Brain. (2013, Feb.) Social networking statistics. [Online]. Available: <http://www.statisticbrain.com/social-networking-statistics/>
- [2] TG Daily. (2013, Feb.) Internet-enabled devices to outpace pc shipments by 2013. [Online]. Available: <http://www.tgdaily.com/hardware-features/57784-internet-enabled-devices-to-outpace-pc-shipments-by-2013>
- [3] Yahoo. (2013, Feb.) Pipes: Rewire the web. [Online]. Available: <http://pipes.yahoo.com/pipes>
- [4] Z. Zhao, N. Laga, and N. Crespi, “A survey of user generated service,” in *Network Infrastructure and Digital Content, 2009. IC-NIDC 2009. IEEE International Conference on*, Nov. 2009, pp. 241–246.
- [5] C. S. Jensen, C. R. Vicente, and R. Wind, “User-generated content: The case for mobile services,” vol. 41, no. 12, Dec. 2008, pp. 116–118.
- [6] J. Tacken, S. Flake, F. Golasowski, S. Prüter, C. Rust, A. Chapko, and A. Emrich, “Towards a platform for user-generated mobile services,” in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, Apr. 2010, pp. 532–538.
- [7] D. Werth, A. Emrich, and A. Chapko, “An architecture proposal for user-generated mobile services,” in *Mobile, Ubiquitous, and Intelligent Computing (MUSIC), 2012 Third FTRA International Conference on*, Jun. 2012, pp. 142–147.
- [8] A. Emrich, A. Chapko, and D. Werth, “Context-aware recommendations on mobile services: The m:ciudad approach,” in *Smart Sensing and Context*, ser. Lecture Notes in Computer Science, P. Barnaghi, K. Moessner, M. Presser, and S. Meissner, Eds. Springer Berlin Heidelberg, 2009, vol. 5741, pp. 107–120.
- [9] C. Fuhrhop, J. Lyle, and S. Faily, “The webinos project,” in *Proceedings of the 21st international conference companion on World Wide Web. WWW ’12 Companion, New York, NY, USA, 2012*, pp. 259–262.
- [10] Webinos. (2013, Feb.) Webinos travel. [Online]. Available: <https://developer.webinos.org/webinos-travel>
- [11] OASIS. (2013, Feb.) Oasis extensible access control markup language (xacml) tc. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

CloudState: End-to-end WAN Monitoring for Cloud-based Applications

Aaron McConnell, Gerard Parr, Sally McClean, Philip Morrow, Bryan Scotney
 School of Computing and Information Engineering
 University of Ulster
 Coleraine, Northern Ireland

Email: a.mcconnell@ulster.ac.uk, gp.parr@ulster.ac.uk, si.mcclean@ulster.ac.uk, pj.morrow@ulster.ac.uk, bw.scotney@ulster.ac.uk

Abstract—Modern data centres are increasingly moving towards more sophisticated cloud-based infrastructures, where servers are consolidated, backups are simplified and where resources can be scaled up, across distributed cloud sites, if necessary. Placing applications and data stores across sites has a cost, in terms of the hosting at a given site, a cost in terms of the migration of application VMs and content across a network, and a cost in terms of the quality of the end-to-end network link between the application and the end-user. This paper details a solution aimed at monitoring all relevant end-to-end network links between VMs, storage and end-users. With this knowledge at hand, it becomes easier to optimise the arrangement of VMs and content with a distributed cloud environment such that resident applications respond in a timely manner, both between cloud-based application components and in the delivery of the application to the end-user. Results show how this system provides network information which influences the choice of location for hosting applications and data.

Keywords—Cloud Computing; WAN Monitoring; Cloud Networking

I. INTRODUCTION

Recent years have seen data centres, firstly adopting virtualisation solutions in order to consolidate servers, and then moving to Cloud environments where Cloud instances can be scaled across distributed resources depending on load [12][5]. Clouds allow applications to be migrated to remote hosts outside of the physical realm of the local data centre [1]. Applications may be statically hosted at a remote location, or it may happen dynamically in a "fail-over" event, i.e., when local data centre resources are saturated and a resource-starved application is temporarily migrated to a remote location where available resources are such that it performs adequately. Depending on the network conditions between data centre locations, moving an application to another location may be ill-advised. It may be that the application (or service) communicates heavily with another application at its original location or with a particular data-store. It may also be the case that the proposed location is further away, in network terms, from the end-user of the application. If the network conditions between the proposed location and the end-user are sufficiently poor then the delivery of the application to the end-user will not be acceptable, despite the application having ample resources within the physical data centre.

It is therefore necessary to have a periodic, automated means of measuring the state of the WAN link between any two addresses relevant to the successful delivery of an application to end-users. This measurement should be taken periodically, with the time between polls being short enough that sudden changes in the quality of the WAN are observed, but far enough apart so as not to flood the network with monitoring traffic. Data collected from polls should also be logged at a central location in order that decision-making about application performance and placement can be made, by a cloud management solution, with a full view of the distributed data centre available, including WAN metrics. Intelligent use of relevant WAN data can enable decision-making to occur which can pre-emptively and reactively lead to action which will ensure applications meet their Service Level Agreements (SLAs). It is also possible, given a WAN history between two addresses, to observe trends related to time of day and workload.

Providing live WAN information, specifically for network-aware placement of cloud-based applications and services, is of commercial importance to vendors of existing cloud vendors where hybrid cloud scenarios are used when private cloud resources are saturated. It is envisaged that network-awareness will be more important in future cloud topologies, where users may frequently migrate their content between cloud vendors in order to save money [11] or increase performance. It is with this in mind that a new solution, called CloudState, has been developed to provide real-time information on the state of the end-to-end WAN link between any two communicating entities related to the operation of the Cloud-based application and its delivery to the end-user. The end-user is defined, during the course of this work, as a corporate customer for a Cloud Provider. The end-user address communicated with is one at the edge of a customer's LAN, typically a router with a WAN IP address.

A. Related Work

Commercial solutions exist which monitor WAN links and provide optimisation, both to the link and to the placement of virtual applications at the end of a link. Ipanema's Ip—Engines [6] are placed at either end of a WAN link, one entity at the data centre and one between the customer's edge

router and LAN. Ipanema's central management software then provides monitoring data for all WAN links relevant to an application component. Given that an Ip—Engine is required at each end of a WAN link, the cost of installing the Ip—Engines, and the intrinsic financial cost involved in scaling up such a WAN monitoring system, it is clear that an improvement may be made, in terms of a WAN monitoring solution, in the case where a Cloud implementation can be scaled up quickly across various locations (and cloud providers). Work has also been carried out with a focus on Application-Layer Traffic Optimisation (ALTO) [9] [4], which has been developed by the Internet Engineering Task Force (IETF). The ALTO protocol may well develop into a standardised means of acquiring network and traffic information, but at present it requires at least one ALTO server to be put in place and a number of ALTO clients, which should be integrated with end-user Web applications. The perspective of the work presented in this paper is that Cloud WAN monitoring system should ideally be simple, non-invasive (i.e., require no special hardware or servers to be configured), scalable (i.e., run inside a VM so that it can be migrated and cloned) and low-cost, both in terms of financial outlay and in terms of resource requirement.

GEANT's perfSONAR provides a range of software to monitor networks and report performance measurements [3]. PerfSONAR aims to provide monitoring data from networking entities, e.g., routers, along the end-to-end network path. This system assumes that, although the entities may be in different domains of ownership, sufficient performance data will be made available, by the commercial vendors involved, in order that end-to-end network performance can be quantified. It is also unlikely that detailed per-hop performance data is required in the case where a decision is to be made about where to place a virtual instance of an application or service. The decision about placing the application or service is only concerned with the quality of the end-to-end link, between the host server and the end-user, and not with the specific performance of each entity en route. PerfSONAR would offer a comprehensive network monitoring solution in the case where multiple cloud providers agree to implement the system and where interfaces for acquiring performance data is shared (as is the case with the OPTIMIS project [15]). An assumption cannot be made that this communal arrangement exists in a cloud computing scenario. Therefore, there is scope for the design and development of a simple solution which monitors only the end-to-end network path.

The Network Weather Service (NWS) [14] is another distributed system for monitoring network performance, with a focus on dynamically forecasting the performance or networking entities. Like perfSONAR, this system requires that, in the case where the end-to-end network path crosses different domains of ownership, performance data and forecasting information are made available to various commercial cloud vendors. The NWS is quite complex in the regard that it requires a name server, memory host, sensor host and forecaster host. As is the case with perfSONAR,

there remains scope for a simple end-to-end monitoring solution. The next section provides a description of the CloudState model, followed by a section describing the prototype solution. Experimentation and results are described in the subsequent section and the final section discusses conclusions and further work.

II. ARCHITECTURE DESCRIPTION

CloudState is aimed at providing a software-based VM-embedded, scalable, migratable WAN analyser for Cloud Computing. A full, up-to-date view of all LAN/WAN links is possible with CloudState, with instances of the application strategically placed throughout a cloud.

The aim of this work is to provide a means of monitoring network capabilities across LANs and WANs, for distributed applications on privately-owned hardware, for elements running on third-party equipment and for a continuous assessment of the link to the end-user. The resultant data, combined with the other cloud performance metrics, provides the means by which QoS guarantees can be ensured, via SLA-compliance, and for optimisation mechanisms to ultimately ensure that the VPC is making best use of available resources.

A. A VM-Embedded Solution

CloudState is designed to reside inside a VM. The central reason for this is because it reduces the amount of work required to install CloudState, configure it and place it within the VPC. Current network assessment tools require hardware installation or special servers and clients, as mentioned in section I-A. It is necessary, for a dynamically-changing distributed topology, that the level of installation, configuration and engineering required is minimised. The ultimate aim with CloudState is that it can be cloned and migrated to a remote location and run with minimal setup. It is designed to require a list of IP addresses, representing the other end of the links to test, its own IP address and connectivity to a centralised Cloud Management Database (CMDB) in order to log data.

B. Multiple Link Monitoring

CloudState is designed to communicate with a list of IP addresses if required. These addresses may represent a number of end-user locations which can potentially use the cloud location where the CloudState instance is resident. These addresses are repeatedly polled at a defined time interval and the results stored in the CMDB. Figure 1 illustrates a typical topology scenario. The centralised CMDB is used in order that requests for network performance data can be made from a single source in order to assess the suitability of numerous sites for placing an application or service.

C. Communication Protocols

The current architecture of CloudState uses the Internet Control Message Protocol (ICMP) protocol to send echo-request packets to and from the destination IP address. This approach is used in order that any IP address can be queried

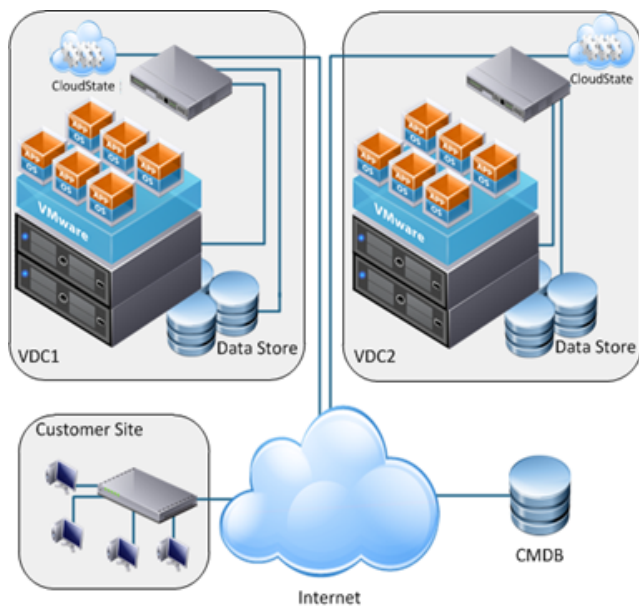


Figure 1. CloudState locations within the cloud

and so that no specialised hardware or software is required at the destination address.

III. CLOUDSTATE PROTOTYPE

CloudState uses BWPing [2] as a library to test each link for bandwidth, latency and packets dropped. BWPing is used to send a number of user-sized ICMP echo-request packets to a destination IP address. The response from the destination, and the volume of packets transmitted, allows bandwidth, latency and packets dropped to be calculated.

The network metrics acquired using BWPing provides the basic functionality of CloudState. The native BWPing C class was converted to C++, an object of which was created in the main CloudState class. Nokias QT signal slot libraries [13] were used to allow BWPing to emit signals each time a ping process was completed. It is necessary for the destination address being pinged by CloudState to be configured so that it responds to ICMP echo-requests, otherwise CloudState will not provide metrics. This is an existing limitation of CloudState, but one which will be addressed in future versions. At present it is possible to provide an IP address to CloudState for which ICMP packets are permitted through a firewall.

CloudState runs on a host at the same physical location as the hosts running VM-based user applications. This means that WAN metrics, returned by CloudState for each address, should match those experienced by each application. From this location, CloudState can communicate with any address relevant to the delivery of each application, e.g., databases, Web Services and end-users. The destination address focused on in this work is the end-user. CloudState is used to gather link metrics between the host, resident at the CloudState location, and the user.

The CloudState user is presented with an interface, show

in Figure 2. From this interface the user may define operational parameters, e.g., packet size, transfer speed, transferred volume, for the underlying BWPing echo-request operation.

CloudState provides the administrator with an interface by which a connectivity parameters can be defined so that the CMDB can be reached. CloudState connects to the remote CMDB and writes the results of each poll to the CloudState database table. Poll metrics, as shown in table I, are stored in the CMDB, along with the address of the CloudState agent, the destination address and a time stamp for the ping operation. Connectivity is achieved using the QT QMySQL Linux driver.

IV. EXPERIMENTATION

A controlled test environment was created in order to validate the metrics returned by CloudState, and to assess the impact of CloudState on both the source host, the destination addresses and the network. The test environment comprised of a Dell R515 Server with two AMD 6-core processors, 16 GB of main memory and twelve 1 Gb network interface cards (NIC). The VMWare ESX 4.1 hypervisor was installed on this server (with load balanced across the twelve NICs) and a CloudState VM placed on it using VMWare vCenter 4.1, which was installed on another networked machine. CloudState was installed within a Ubuntu Linux VM with 1 virtual processor, 512 MB of RAM and a 4 GB virtual thin-provisioned hard drive.

CloudState was used to assess the bandwidth and latency of a known 100 Mbps link. A VM was polled, running on another identical Dell R515 host, connected by a single 100 Mbps switch. Results for the bandwidth returned varied depending on the parameters used for the underlying BWPing operation. Packet sizes ranging from 500 bytes to 1500 bytes (the largest allowed by Ethernet at the network layer) were tested as well as a range of transmission data volumes. The aim is to momentarily saturate the network so that the bandwidth can be quantified, but for the monitoring load to be create minimal intrusion on the network and destination address.

Two methods were used to validate the results returned by CloudState: IPerf [7] was used to ensure the bandwidth values returned by Cloudstate were similar to those of IPerf, when no emulated network degradation was forced, and a

TABLE I
CLOUDSTATE CMDB FIELDS

Parameter	Description
Source	The source IP address of the CloudState application
Host	IP address of the represented host
Target	The target IP address of the ping operation
PacketSize	The size of each packet transmitted to the target
Totalpkts-tx	The number of packets transmitted to the target
Totalpkts-rx	The number of packets received from the target
Vol-tx	The number of bytes transmitted (packetSize x totalpkts-tx)
Vol-rx	The number of bytes received from the target
time-secs	The time taken for the complete operation
Speed-kbps	The bandwidth of the link
Rtt-min	The minimum round-trip-time taken
Rtt-max	The maximum round-trip-time taken
Rtt-mean	The mean round-trip-time taken
Date/Time	Date and time of ping

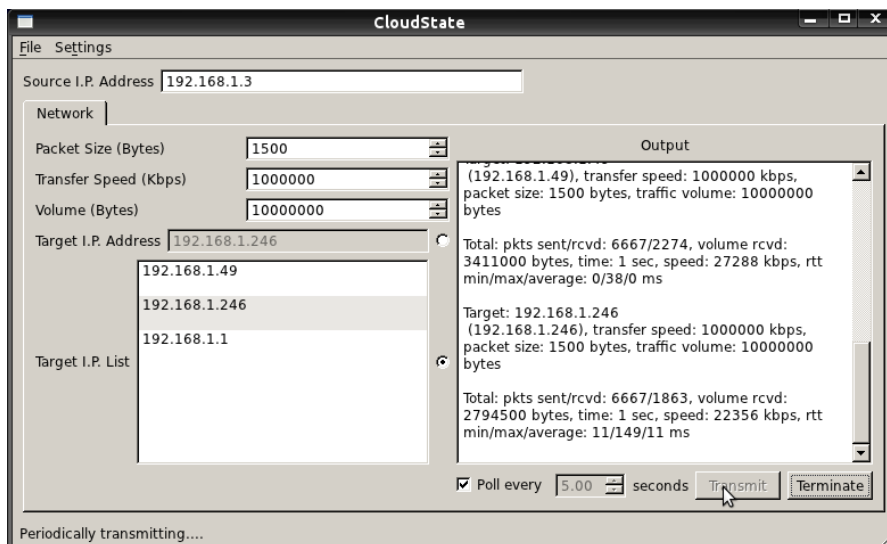


Figure 2. The CloudState user interface

simple Linux ping was used to assess the latency. Over the native 100 Mbps link, IPerf returned a mean (of 5 polls) bandwidth rating of 82.34 Mbps, with an average volume of 98.2 MB of data transferred at each poll. The mean (of 5 polls) latency returned by ping was 1 ms. CloudState returned a slightly lower average bandwidth value of 78.99 Mbps with a packet size of 1500 bytes and a volume of 10 MB. It is acknowledged that further work is required in order to ensure the bandwidth values returned match those returned by other bandwidth measurement tools, but at this point there is an argument that a less accurate measurement is acceptable given that only 10.18% (of the volume of data IPerf used) was loaded onto the network to return a bandwidth value which was 95.93% accurate. The inaccuracy is probably due to the fact that the volume of data transmitted does not fill the bandwidth available. An algorithm is required that increases the volume at each poll until a small percentage of packets is dropped, indicating bandwidth limitations.

The fixed values for packet size and volume currently used proved to be most accurate in assessing bandwidth and latency for each link tested. Figure 3 illustrates the bandwidth returned as the packet size was changed from 500 bytes to 2000 bytes in steps of 500 bytes. A traffic volume of 10 MB was used for each test. The graph shows a bandwidth increase up to a packet size of 1500 bytes followed by a steep decline in bandwidth when a packet size of more than 1500 bytes was used. More packets are sent when the packet size is small, e.g., 500 bytes, in order to achieve the same transfer volume. This increases the per-packet delay because each packet must be processed. The level of throughput in a given timeframe is therefore reduced. Large packets can also reduce throughput because of the time required to process the amount of data in each packet. This is evident in Figure 3 when the packet size is increased beyond 1500 bytes. Figure 4 shows the bandwidth returned as the volume was increased from 5 MB to 25 MB,

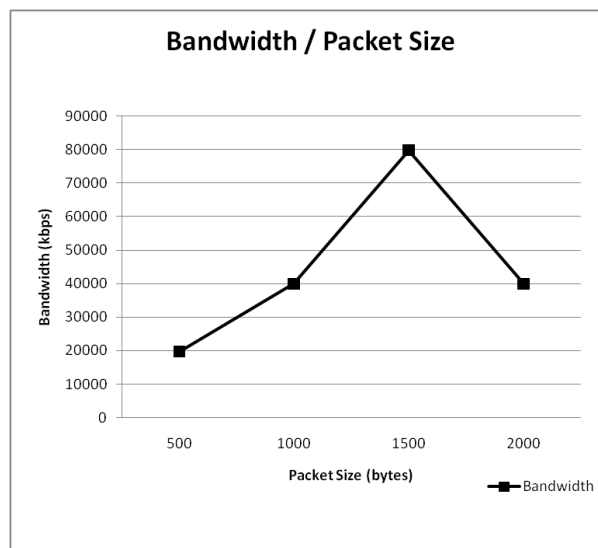


Figure 3. The bandwidth values returned as the packet size is increased

with a constant packet size of 1500 bytes [8].

Figure 6 shows the latency values returned as the latency for a given link was artificially increased using WANem [16]. Latency values returned were an average of 2 ms higher than those set, the extra being introduced by the processing of the WANem gateway. Similarly, Figure 7 shows the reported bandwidth compared with the artificially-set bandwidth using WANem. The reported bandwidth is slightly lower than that set, except for when the set bandwidth is higher than 80 Mbps. At this point the limitations of the 100 Mbps physical link prevent the actual bandwidth reaching that set with WANem.

Given that a CloudState instance runs at each data centre location within a cloud, it is important to calculate the likely impact on a destination node when it is polled by numerous

TABLE II
BANDWIDTH AND LATENCY VALUES RETURNED AS THE NUMBER OF POLLED ADDRESSES IS SCALED UP

No. Targets	Min Lat (ms)	Max Lat (ms)	Mean Lat (ms)	BW (kbps)	Time Taken (secs)
1	< 1	13	< 1	79956	1
2	< 1	13	< 1	79956	1
4	< 1	16	< 1	79887	2
6	< 1	13	< 1	79896	2
8	< 1	14	< 1	79920	1
10	< 1	12	< 1	79992	1
12	< 1	15	< 1	79860	1
14	< 1	13	< 1	79932	1

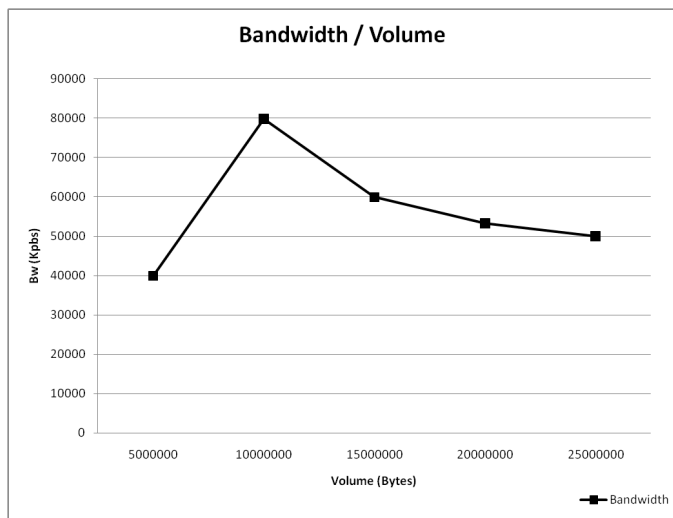


Figure 4. The bandwidth values returned as the data volume is increased

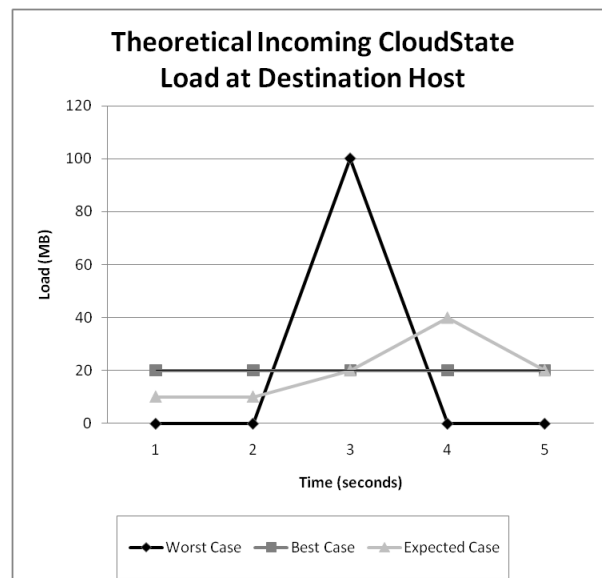


Figure 5. Theoretical impact of CloudState load at a destination address

CloudState instances. The amount of traffic in bytes received by a destination address for each CloudState poll will be:

$$U = \sum_{i=1}^N S_i P_i \tag{1}$$

where U is the amount of traffic received by a destination address given i number of CloudState instances communicating with it, each with a given packet size S_i and number of packets transmitted P_i . Figure 5 illustrates theoretical incoming CloudState load trends at a destination host with which 10 CloudState instances communicate, with each CloudState instance transmitting 10 MB of data at intervals of 5 seconds each. Three different situations are possible, one where there is a momentary spike in the incoming load where all of the CloudState instances transmit at the same time. This is the worst case scenario because it is possible that the destination host will encounter a momentary I/O outage. This also will affect both the delivery of the application running on the host as well as the results returned by each CloudState instance. The available bandwidth value will not accurately reflect the average state of the link to that destination host. The best case scenario is one where 2 of the 10 CloudStates poll at the same time and the load is kept almost constant at the destination. The expected case is that there are some spikes in CloudState load but not an “all or nothing” scenario. The effect of the

possible CloudState trends on application performance, at the destination host, will differ depending on the amount and the profile of application traffic. It is clear that the poll period for each CloudState should be reduced if it is found to impair the delivery of an application at any host.

V. CONCLUSIONS

Cloud Computing has become a key paradigm in the area of distributed computing. The underlying virtualisation means that, unlike Grid Computing, distributed host node resources are both fragmented for application placement and consolidated to offer more resources to facilitate a resource-hungry application when needed. It is this dynamic, virtual arrangement of resources that causes both a problem in terms of the quality of the WAN link between them at any given time, and an opportunity for intelligent placement of applications such that their WAN needs are satisfied. CloudState provides a low-cost, highly-scalable solution to this problem. Instances can be easily deployed on base systems or within VMs. VM resource usage is low - approximately 712.69 Mhz is used for the CloudState VM under a load of 14 destination addresses. Memory usage equates to 348.16 MB of main memory for the entire VM under the same load of destination addresses. No

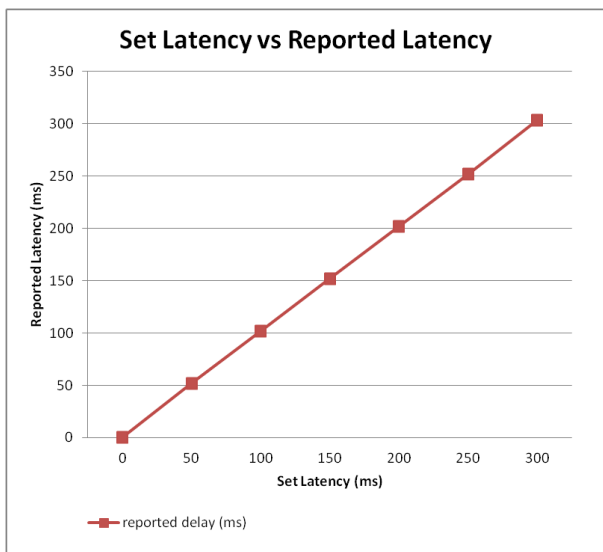


Figure 6. CloudState set latency vs reported latency (ms)

special hardware, servers or clients are required throughout the distributed cloud, except for the requirements that all addresses respond to ICMP echo requests.

A. Further Work

CloudState is currently installed within a Ubuntu 10.10 Linux VM which contains the Gnome desktop environment and a range of applications which are installed with Ubuntu by default. A bare-bones Linux install, e.g., Ubuntu JeOS, would be more suitable for hosting CloudState where unnecessary programs are not installed and the footprint, both in terms of hard-drive space and run-time resource requirements, are minimised. There is an argument for removing the GUI from CloudState and having operational parameters passed as run-time arguments and/or stored in settings files. This would mean that a Linux desktop environment is not required and would make CloudState a very lightweight VM.

CloudState currently transmits a user-defined data volume at each poll. An algorithm should be included that starts off

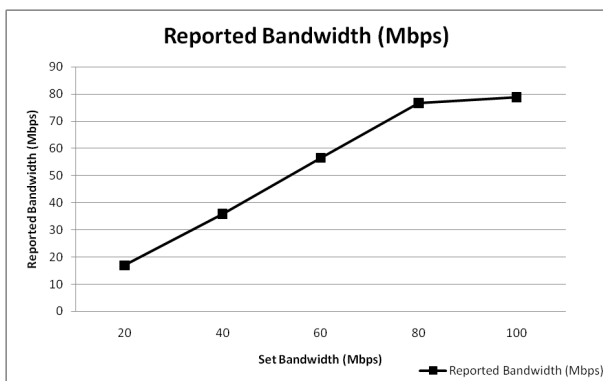


Figure 7. CloudState set bandwidth vs reported bandwidth (Mbps)

with a small amount of data to be transmitted, which increases with each poll until the bandwidth of the link is established. The volume of data to be transmitted over that link should be recorded and then only periodically checked thereafter. There may be some scope in examining a correlation between the volume of data sent, the link bandwidth and the number of packets dropped at each poll. It is speculated that an overloaded link will discard packets and the volume should be set at the point where a small percentage of packets are dropped. This requires further research.

CloudState may be improved by incorporating a different approach to gathering network performance statistics. Gathering performance data at the application level is possible [10] using Web client probes, especially with a focus on ALTO. This approach would ensure that firewalls are not an issue in gathering performance data and would provide data that may be more accurate, given that it is gathered at the end-user and not at other points on the path between the VM-hosted application and the end-user, e.g., at a router or switch.

REFERENCES

- [1] R. Buyya, Chee Shin Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on, pages 5 –13, sept. 2008.
- [2] Oleg Derevenetz. Bwping - open source bandwidth measurement tool. <http://bwping.sourceforge.net/>, June 2011. [retrieved: March. 2013].
- [3] GEANT. perfsonar. <http://www.perfsonar.net/>, 2012. [retrieved: March. 2013].
- [4] V.K. Gurbani, M. Scharf, T.V. Lakshman, V. Hilt, and E. Marocco. Abstracting network state in software defined networks (sdn) for rendezvous services. In Communications (ICC), 2012 IEEE International Conference on, pages 6627–6632, june 2012.
- [5] Wei Hao, I-Ling Yen, and B. Thuraisingham. Dynamic service and data migration in the clouds. In Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International, volume 2, pages 134 –139, july 2009.
- [6] Ipanema. Ip engine. <http://www.ipanematech.com/en/ip-engine>, June 2011. [retrieved: March. 2013].
- [7] Iperf. Iperf. online, <http://iperf.sourceforge.net/>, December 2008. [retrieved: March. 2013].
- [8] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos. Variable packet size buffered crossbar (cicq) switches. In Communications, 2004 IEEE International Conference on, volume 2, pages 1090 – 1096 Vol.2, june 2004.
- [9] S. Kiesel, L. Popkin, S. Previdi, R. Woundy, and Y.R. Yang. Application-layer traffic optimization (alto) requirements. Internet Engineering Task Force, Internet-Draft draft-ietf-alto-reqs-00, 2009.
- [10] Myung-Sup Kim, Young J Won, and James Won-Ki Hong. Application-level traffic monitoring and an analysis on ip networks. volume 27, pages 22–42, 2005.
- [11] Jeffrey S. Klaus. Follow-the-moon scheduling to lower energy costs. <http://www.datacenterknowledge.com/archives/2012/10/22/follow-the-moon-scheduling-to-lower-energy-costs/>, 2012. [retrieved: March. 2013].
- [12] Lijun Mei, W.K. Chan, and T.H. Tse. A tale of clouds: Paradigm comparisons and some thoughts on research issues. In Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE, pages 464 –469, dec. 2008.
- [13] Nokia. Qt sdk. <http://qt.nokia.com/products/qt-sdk>, June 2011.
- [14] NPACI. Network weather service. <http://nws.cs.ucsb.edu/>, 2004. [retrieved: March. 2013].
- [15] Optimis. Optimis. <http://www.optimis-project.eu/>, 2013. [retrieved: March. 2013].
- [16] WANem. Wanem - the wide area network emulator. online, <http://wanem.sourceforge.net>, December 2009. [retrieved: March. 2013].

Transparent Access on Encrypted Data Distributed over Multiple Cloud Infrastructures

Luca Ferretti, Michele Colajanni, and Mirco Marchetti
University of Modena and Reggio Emilia
Modena, Italy
{luca.ferretti, michele.colajanni, mirco.marchetti}@unimore.it

Adriano Enrico Scaruffi
Doxee SpA
Modena, Italy
ascaruffi@doxee.com

Abstract—Using cloud infrastructures to store and backup data is becoming a popular alternative that guarantees performance and scalability at reasonable prices. However, standard cloud solutions could raise some concerns about data confidentiality and dependency on a single provider. We aim to address these issues by using cloud storage of multiple cloud providers. Our solution ciphers, partitions and replicates data among multiple cloud architectures, thus augmenting availability and confidentiality, and avoiding lock-in of one cloud provider. The proposed model is implemented through open source software that leverages data storage offered by multiple providers. This prototype demonstrates the effectiveness of the geographically distributed architecture in several real case scenarios.

Keywords- cloud; storage; encryption; file system; replication

I. INTRODUCTION

Cloud storage is an interesting alternative that allows users to leverage huge size disk spaces characterized by high availability and scalability at pay-per-use cost models. However, when companies outsource their information to the cloud, there are many concerns about data confidentiality and complete dependency on one cloud provider. Issues such as law restrictions [1], vendor lock-in and unavailability cases causing service interruptions and data losses (e.g., [2]) are limiting a widespread adoption of cloud storage solutions.

This paper proposes a novel architecture that aims to augment data resiliency and confidentiality, and to avoid possible lock-in related to one cloud provider. The idea is to implement a virtual file system where data are encrypted, replicated and disseminated among different cloud providers. In such a way, there is no dependence on one provider, and adopted encryption schemes are robust even against insider attacks and colluding providers. Moreover, we consider it important to provide users with a transparent encrypted access to such virtual file system. Thanks to the proposed standard file system interface, any application operating on files can leverage the proposed architecture without software modifications. In this paper, we demonstrate the efficacy of the proposed architecture by running a relational database on top of it.

Existing solutions [3]–[5] concerning data confidentiality, integrity and replication for untrusted storage services do

not meet all requirements about encryption, replication and transparency. For example, data replication is not considered in [3]. Unlike our architecture based on the Infrastructure as a Service (IaaS) paradigm, the system described in [4] refers to the more sophisticated and expensive Storage as a Service paradigm. This scheme transparently provides customers with advanced techniques for elasticity, scalability and availability, but it requires the implementation and maintenance of dedicated drivers for each cloud storage API, thus causing additional cloud lock-in problems. The interesting solution proposed in [5] has two drawbacks: it is not quite transparent to the customer because it requires changes at the level of application logic; moreover, it is not resistant against colluding cloud providers.

The proposed architecture guarantees data confidentiality and integrity at rest, in motion and in use. To provide users with complete confidentiality of outsourced data we adopt encryption techniques and algorithms of proven security. Data are replicated in a multi-tenant architecture built over multiple cloud storage services. In this paper, we describe the overall model, the details of the architecture components, and the guidelines for its implementation.

The remaining part of this paper is structured as following. Section II analyzes other solutions related to our proposal. Section III describes the architectural model and the main requirements. Section IV reports the internal details of the proposed architecture and the main functionalities. Section V presents an example of a relational database that can leverage the proposed architecture. A summary of the results is reported in Section VI.

II. RELATED WORK

Data confidentiality on untrusted storage was initially guaranteed by encrypted file systems (e.g., [3], [6]) that allow a customer to encrypt all data stored in a cloud IaaS. However, these solutions do not allow to slice and to replicate data among several cloud providers as provided by previous architectures including that proposed in this paper.

Some academic and commercial proposals guaranteeing data confidentiality and integrity by using multi-tenant cloud services are recently appearing. The solutions most related to this paper are iDataGuard [4] and Depsky [5]

iDataGuard is a middleware that leverages the cloud Storage as a Service paradigm. This approach differentiates iDataGuard from our solution that is based on the standard IaaS paradigm. Cloud storage services can take advantage of several benefits with respect to IaaS, because they transparently provide customers with advanced API-based solutions for elasticity, scalability and availability. These techniques facilitate the low level implementation of iDataGuard, but they require the software implementation and maintenance of dedicated drivers for each specific cloud storage API. As a consequence, this solution limits portability and reduces the possibility of avoiding cloud provider lock-in. We should also observe that iDataGuard does not transparently replicate information among the cloud storage services, but data are managed by users as distinct storage units.

Depsky [5] proposes an interesting storage architecture that allows key-value access to data and guarantees data consistency also in the worst case of Byzantine faults. Depsky requires clients to access intermediate trusted components that provide key distribution by means of a Shamir secret sharing scheme [7]. This does not guarantee data confidentiality in the case of colluding cloud providers. Another problem is that applications based on Depsky require changes at the software level, because this architecture comes with a non-standard interface for data management.

Other papers (e.g., [8], [9]) aiming to guarantee confidentiality of information stored in untrusted storage servers can avoid data encryption. For example, they guarantee k-anonymity [10] by splitting sensitive data among multiple subsets, each managed by an independent cloud provider. Since data are not encrypted, each cloud can obtain some information on a portion of data. Moreover, such techniques require a complete awareness of the underlying data structure, that are against our main design requirement that the proposed solution must be transparent to the applications. In order to guarantee data confidentiality in the cloud database paradigm, full homomorphic encryption [11] is described as the final solution for single client computing scenarios [12]. In practice this approach is not yet feasible because of the prohibitive computational costs on possible operations.

A different set of proposals are oriented to cloud database services that differ from the architecture proposed in this paper because of the logical software level, and lack of transparency and portability. For example, some DBMS engines provide users with advanced proprietary techniques to encrypt data at storage level (e.g., Transparent Data Encryption (TDE) [13]). These features can replace the encryption layer of the proposed architectures, and can improve performances thanks to data caching and selective blocks retrieval. However TDE implementation is related to some specific DBMS, and many database services do not propose any similar solution. On the other hand, we remark that the proposed architecture aims to be transparent of any specific DBMS and cloud-related solution. Cloud

database as a service (e.g., [14]–[17]) is an interesting alternative to support database in cloud infrastructures. They have the advantage of executing database SQL computations directly on the cloud infrastructure and to leverage intrinsic scalability and reliability of a cloud provider. However, there are no proposals that are oriented to federate databases among multiple cloud providers.

III. MODEL OVERVIEW

An architecture guaranteeing maximum availability and security on untrusted storage services should satisfy the following main objectives.

- Confidentiality must be guaranteed for data at rest, in motion and in use without any risk of information leakage due to cloud insiders and collusive providers.
- Service availability must not depend on one cloud provider.
- The proposed architectures should be transparent to the supported applications in the broadest sense, that is, no modifications must be required at the application level.

To satisfy all the previous objectives we propose the architectural model that is represented in Figure 1.

Let us consider an application that executes some operations requiring accesses to data storage. This is a *plain data* scenario where the application does not provide any solutions to guarantee data confidentiality. The application executes virtual data operations on a file system, as if it were on local storage. As transparency is one of the main objectives of the proposed architecture, our solution adopts a standard file system interface that guarantees the required level of transparency. In practice, data are not stored in local devices nor in a local network environment as it is usually done in private data centers. Instead, all data are stored in multiple cloud infrastructures. Other main logical components of the proposed architecture are the *data encryptor*, the *data slicer* and the *data replication* modules. The combination of all of them guarantees confidentiality, availability and resiliency of data managed by the application.

To give a high-level description of the architecture model, we initially identify a trusted area and untrusted areas. The trusted area is under the direct control of the data owner, and can be accessed by only trusted third-party subjects. Plain data must never access the untrusted area before being encrypted. All security policies and decryption keys must be managed by trusted parties.

Each application executes operations on plain data and does not require any software modifications in order to guarantee the correct execution of the security techniques that our solution applies. It is the proposed architecture that provides applications with a standard file system interface allowing them to manage data as in local storage devices, although data are really stored on several Infrastructures as

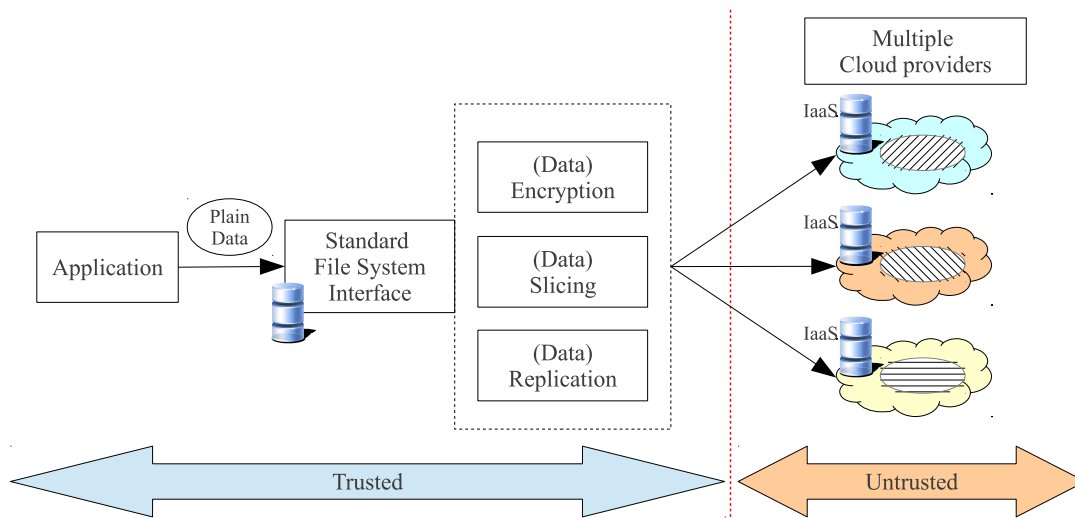


Figure 1. Architectural model.

a Service (IaaS) resources that are under the direct control of multiple cloud providers.

Cloud IaaS is not the only choice to use cloud storage because cloud providers offer also cloud storage as a service solutions through high level APIs facilitating data management. Our choice of preferring cloud IaaS instead of the cloud storage as a service paradigm was motivated by the following three reasons.

- 1) The IaaS paradigm allows us to directly manage virtual machines and disk resources that are standard; consequently, we can install and configure the best solutions to satisfy the architectural requirements of transparency and data confidentiality.
- 2) Cloud storage as a service requires data to be managed through proprietary APIs. This may cause some forms of cloud lock-in and may limit the portability of the solutions.
- 3) Cloud storage as a service can transparently provide advanced replication techniques to guarantee advanced resiliency. However, these benefits can be achieved also through the proposed architecture without any additional reliance on non-standard cloud services.

In our proposal, plain data received from an application are subjected to two types of manipulations:

- encryption to guarantee information confidentiality;
- distribution over multiple cloud infrastructures to increase availability and avoid dependency on one provider.

Slicing and replication reinforce security in the worst case scenario of collusion between a cloud provider and an internal (theoretically trusted) subject, because it prevents a cloud provider from accessing the whole data set. Moreover, they are useful to increase performance because they allow

the parallelization of some data operations, and reduce space overhead caused by replication.

In the following Section IV we describe the details of the architecture and outline its implementation.

IV. ARCHITECTURE

The paper proposes a novel architecture that allows clients to leverage remote storage of multiple cloud providers. While internally managed infrastructures allow data owner to directly control data security policies, the cloud paradigm has the advantage to reduce costs and augment scalability, availability and resiliency. On the other hand, it opens user concerns in terms of data confidentiality and dependency on one provider.

We describe the implementation of the architectural model described in Figure 1 by referring to the architecture represented in Figure 2. A possible alternative based on a broker implementation is outlined in Figure 5.

Users applications transparently execute data operations on a logical file system, that is implemented by the interface layer of the proposed solution. Data replication strategies over multiple cloud storage servers are implemented by the **secure data management** (SDM) component. It guarantees that all data are stored in the infrastructures of at least two cloud providers (*high reliability*), and no provider owns all data (*high confidentiality*).

The Secure Data Management (SDM) represents the core of the proposed architecture that is typically implemented on an intermediate server. This proxy executes encryption and data distribution schemes over all application data, making use of multiple cloud providers to store encrypted encrypted data. The main modules of the SDM component are represented in Figure 3 and described below.

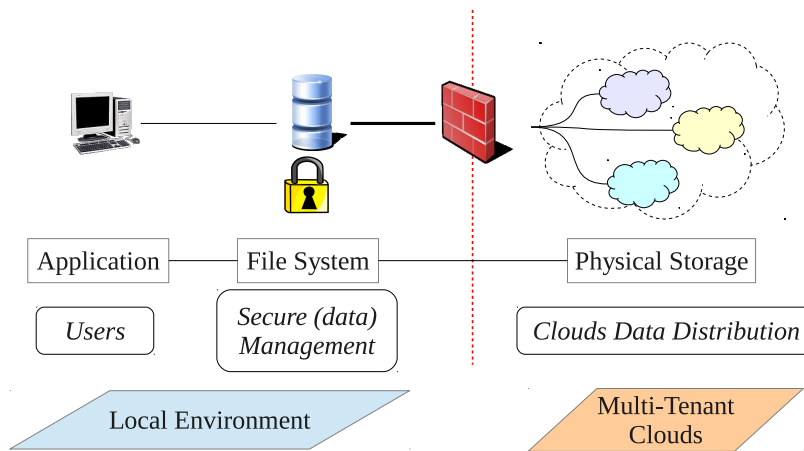


Figure 2. Architecture based on clients and cloud providers.

- A. The file system interface implements a logical standard layer to the client applications.
- B. The cache manager uses local storage to cache previously accessed data.
- C. The disk encryptor implements well known encryption algorithms, such as AES [18], guaranteeing data confidentiality.
- D. The distributed file system operates slice and replication policies on user data over multiple cloud providers. The possible alternatives and details are presented below.
- E. The virtual private network guarantees confidentiality on untrusted channels of communication by encrypting all data in transit and authentication schemes for the cloud services.

Plain data of the user applications flow through the software modules of the intermediate proxy that fulfills all main requirements described in Section III. The most visible interface for the client applications is a **logical file system**. When stored data are accessed or modified by a client application, the logical file system searches for a hit in its local cache. If no match is found, then the request is forwarded to the underlying SDM modules. The performance benefits of caching strategies in geographically remote cloud storages is of paramount importance as evidenced in [19].

The **encryption module** transparently encrypts all data received from the logical file system. We use a software block mapping device that maintains a unique correspondence between an underlying encrypted storage and a logical interface to an unencrypted virtual device. In this version of our architecture, we use Dm-Crypt [20] that is a valid block mapper solution integrated in modern Linux kernels.

The underlying encrypted data are stored in the **distributed file system** (DFS) that replicates data among multiple cloud services. Since cloud IaaS services are commonly accessed by a global IP address as a remote host, any DFS

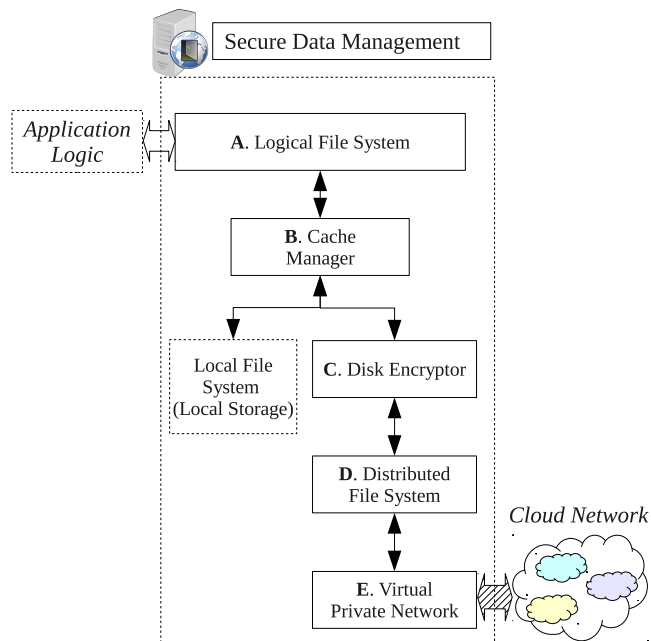


Figure 3. Software modules of the Secure Data Management (SDM) component.

can be used without any modifications. GlusterFS [21] is the chosen DFS satisfying our requirements. It installs and configures software components at the local side (clients) and at the remote cloud side (servers). Different file systems can operate different slicing and replication policies by using data at different system levels, such as blocks, files, volumes. The proposed architecture does not restrict the use of any specific policy, but our implementation choice (GlusterFS) distributes data at the file level, and guarantees integrity of data though hashing algorithms.

A **virtual private network** (VPN) adds a further level of confidentiality. It is not strictly necessary and it can be

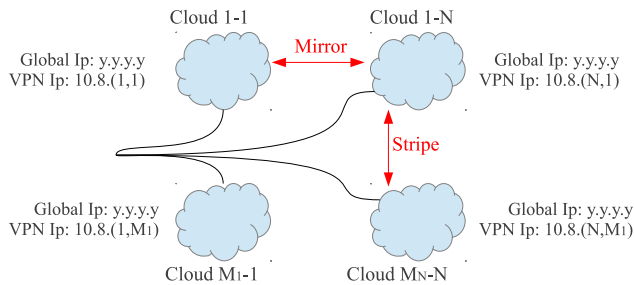


Figure 4. Network configuration of the multiple cloud storages.

avoided when performance becomes an issue. Through the VPN we can also configure the distributed cloud storages as if they were in a local network. Administrators can configure the network of cloud services by using common secure network mechanisms, such as firewalls, subnets and virtual LANs. OpenVPN [22], which is our choice for the present version of the software, is deployed at the local (server) and cloud sides (clients).

Benefits given by the use of the distributed file system and the virtual private network are represented in Figure 4. Each cloud IaaS is identified by the global IP address, and the VPN allows the configuration of a virtual network among the cloud services and the host that executes our software solution. Hence, we can associate each cloud service with a local network address. In the represented scheme, G cloud storage are grouped in N sets. Each group of clouds n has M_n members, such that $\sum_{n=1}^N M_n = G$. Clouds of the same set share the same subnet in the VPN network and are configured on a striping replication configuration. The striping configuration avoids that one cloud provider can manage the entire data set. The different subnets are configured in a mirror replication to increase availability and to break dependency on a single cloud provider. We notice that the possibility of using groups of different sizes is allowed only if the distributed file system can administrate data striping independently for each replicated data. Using groups of different sizes can be useful to balance data among infrastructures with different resource capabilities (we depend on network bandwidth and storage) and respective costs.

It is also important to specify that the proposed architectural solution can be deployed through a third party broker that implements the SDM components. This alternative has the great advantage of avoiding that a customer company must manage the complexity of the SDM, and additional secure infrastructures. This alternative is represented in Figure 5 and outlined below. In such a case, clients communicate with the broker proxy gateway through standard Internet protocols. The broker can be a different company that has direct contacts and contracts with multiple cloud providers. It implements the entire virtual file system and, thanks to

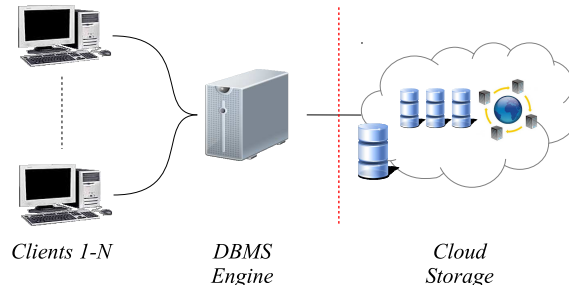


Figure 6. Example of a DBMS deployed over a cloud infrastructure.

an intermediate proxy server, it provides a storage service to the users. The trade-off of this alternative configuration should be clear: the customer can benefit from a simplified interface that avoids any implementation complexity; on the other hand, the broker must be a trusted subject.

V. USAGE SCENARIO

A relational database (DBMS) is a typical application that can take advantage of the proposed architecture. We initially consider an existing scenario, represented in Figure 6, where the database engine is deployed in a local environment, while the data storage is moved to a storage service related to a cloud provider. In a similar architecture, the data owner can take advantage of scalability and adequate resilience, but it does not have any guarantees about confidentiality of data stored to an external cloud service. Moreover, availability and data accessibility depend on one cloud provider that must be trusted by the data owner. While this scenario could be acceptable for some private customers using a cloud storage to backup non-critical information, most companies require additional guarantees about confidentiality and availability before outsourcing data.

Thanks to the proposed architecture we can guarantee that data stored in the cloud is confidential, and that a cloud provider cannot prevent the data owner from accessing its data. We show the configuration related to the broker-less solution in Figure 7.

Clients execute database operations to the local DBMS engine that is connected to the interface of the secure file system to manage data to/from the cloud storages. In this example, we use four cloud providers, where two groups of two clouds are internally organized in a striping configuration, and two groups are configured in a mirroring configuration. Each cloud provides us with an infrastructure as a service paradigm (IaaS), where we can install the distributed file system servers and the virtual private network clients. The encryption layer encrypts all data that are sent by the DBMS, and decrypts all requested data by imposing the database storage in the virtual space created by the device mapper. The distributed file system guarantees that no cloud providers can store the entire data set, because each of them

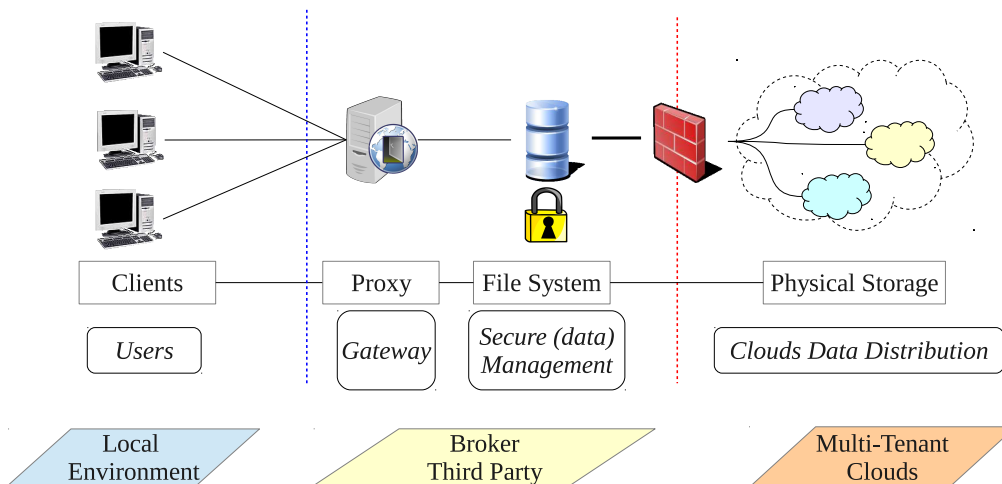


Figure 5. Architecture based on clients, third party broker and cloud providers.

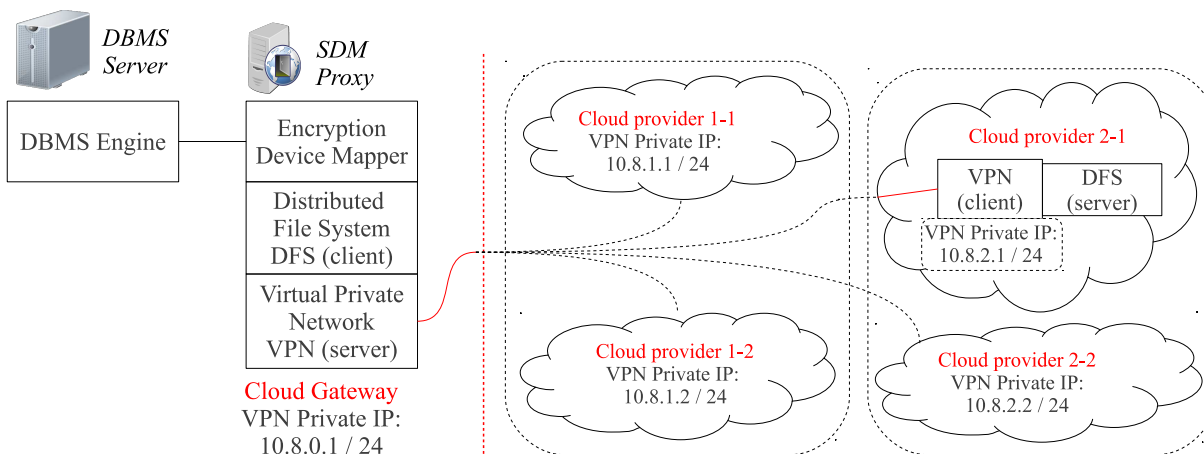


Figure 7. Example of a DBMS deployed over multiple cloud infrastructures (local manager).

has at most half of the entire data set, and all data are stored in at least two cloud providers.

The virtual private network allows us to guarantee security over the access to the clouds, and to configure the replication as if it were in a local area network. As described in Section IV, the clouds that are configured in a striping distribution share the same subnet.

It is important to observe that all tools of the deployed DBMS engine can be used as in a full local environment. Users access policies can be managed as in a standard unencrypted database architecture, because encrypted data are transparently managed by the DBMS engine through the file system interface of the proposed solution. We highlight that this configuration performance can benefit of the DBMS engine caching policies, in addition to the caching mechanisms provided by our architecture (see Section IV).

In this example, the DBMS engine is implemented in PostgreSQL [23], that is a well-known open-source rela-

tional database. It can be deployed in the proposed architecture because it stores data in a standard directory that can be redirected to the file system interface of the proposed solution. Moreover, it allows us to leverage caching policies that are aware of the structure of the database and of the queries.

VI. CONCLUSIONS

This paper proposes a novel architecture to leverage multiple cloud storage services while guaranteeing data confidentiality and avoiding customer dependency on one cloud provider.

Data confidentiality is guaranteed by means of classical encryption schemes; data are replicated among several cloud providers through striping and mirroring techniques. Striping increases performance and data protection, because it prevents that one cloud provider stores the whole data set. The proposed architecture is transparent to the application

layer, as it provides the client with a standard file system interface.

We demonstrate how the proposed architecture can be implemented through open source software components. Moreover, we show that it is suitable to support any kind of applications working on storage service; in this paper, we consider the complex case of a relational database, but other applications are supported as well. This work was focused on the feasibility of the proposal, while performance tests for different workload models and network latencies represent an ongoing work.

REFERENCES

- [1] W. Jansen and T. Grance, "Guidelines on security and privacy in public cloud computing," NIST special publication, pp. 800–144, 2011.
- [2] The New York Times, "Amazon's trouble raises cloud computing doubts," <http://www.nytimes.com/2011/04/23/technology/23cloud.html>, March 2013.
- [3] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in Proceedings of the 2nd USENIX Conference on File and Storage Technologies, vol. 42, 2003, pp. 29–42.
- [4] R. Jammalamadaka, R. Gamboni, S. Mehrotra, K. Seamons, and N. Venkatasubramanian, "Idataguard: middleware providing a secure network drive interface to untrusted internet data storage," in Proceedings of the 11th international conference on Extending database technology: Advances in database technology. ACM, 2008.
- [5] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," in Proceedings of the 6th conference on Computer systems. ACM, 2011, pp. 31–46.
- [6] E. Zadok, I. Badulescu, and A. Shender, "Cryptfs: A stackable vnode level encryption file system," Citeseer, Tech. Rep., 1998.
- [7] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, 1979.
- [8] V. Ciriani, S. D. C. Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Keep a few: Outsourcing data while maintaining confidentiality," in Proceedings of the 14th European Symposium on Research in Computer Security. Springer, 2009, pp. 440–455.
- [9] P. Samarati, "Protecting respondents identities in microdata release," IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 6, pp. 1010–1027, 2001.
- [10] L. Sweeney, "k-anonymity: A model for protecting privacy," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 05, pp. 557–570, 2002.
- [11] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.
- [12] M. Van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," in Proceedings of the 5th USENIX conference on Hot topics in security. USENIX Association, 2010, pp. 1–8.
- [13] Oracle corporation, "Oracle advanced security," <http://www.oracle.com/technetwork/database/options/advanced-security>, March 2013.
- [14] H. Hacigumus, B. Iyer, and S. Mehrotra, "Providing database as a service," in Proceedings. of the 18th International Conference on Data Engineering. IEEE, 2002, pp. 29–38.
- [15] L. Ferretti, M. Colajanni, and M. Marchetti, "Supporting security and consistency for cloud database," in Proceedings of the 4th International Symposium on Cyberspace Safety and Security. Springer, 2012, pp. 179–193.
- [16] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in Proceedings of the 2002 ACM SIGMOD international conference on Management of data. ACM, 2002, pp. 216–227.
- [17] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in Proceedings of the 23rd ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 85–100.
- [18] J. Daemen and V. Rijmen, The design of Rijndael: AES-the advanced encryption standard. Springer, 2002.
- [19] M. Vrabie, S. Savage, and G. M. Voelker, "Bluesky: A cloud-backed file system for the enterprise," in Proceedings of FAST, 2012, pp. 237–250.
- [20] Dm-Crypt, "Linux kernel device-mapper crypto target," <http://code.google.com/p/cryptsetup/wiki/DMCrypt>, March 2013.
- [21] GlusterFS, "Open source, distributed file system," <http://www.gluster.org>, March 2013.
- [22] OpenVPN, "Open source vpn," <http://openvpn.net>, March 2013.
- [23] The PostgreSQL Global Development Group, "Postgresql," <http://code.google.com/p/cryptsetup/wiki/DMCrypt>, March 2013.

Forensics-as-a-Service (FaaS): Computer Forensic Workflow Management and Processing Using Cloud

Yuanfeng Wen, Xiaoxi Man, Khoa Le and Weidong Shi

Department of Computer Science

University of Houston

Houston, Texas 77204-3010

e-mail: {wyf, xman, ktle, larryshi}@cs.uh.edu

Abstract—Digital forensics is a critical technology for obtaining evidences in crime investigation. Nowadays, the overwhelming magnitude of data and the lack of easy-to-deploy software are among the major obstacles in the field of digital forensics. Cloud computing, which is designed to support large scale data processing on commodity hardware, provides a solution. However, to support forensic examination efficiently using cloud, one has to overcome many challenges such as lack of understanding and experiences on configuring and using digital forensic analytic tools by the investigators, and lack of interoperability among the forensic data processing software. To address these challenges and to leverage the emerging trends of service based computing, we proposed and experimented with a domain specific cloud environment for supporting forensic applications. We designed a cloud based framework for dealing with large volume of forensic data, sharing interoperable forensic software, and providing tools for forensic investigators to create and customize forensic data processing workflows. The experimental results show that the proposed approaches can significantly reduce forensic data analysis time by parallelizing the workload. The overhead of the investigators to design and configure complex forensic workflows is greatly minimized. The proposed workflow management solution can save up to 87% of analysis time in the tested scenarios.

Keywords—cloud computing; digital forensics

I. INTRODUCTION

Digital forensics is a technology to collect, examine, analyze, but still preserve the integrity of the data in modern high-tech crimes [1]. Digital forensics were conventionally used in physical hardware analysis, such as hard-disk, flash drives. As the ever increasing computing and storage needs arising in the Internet age, investigators in the public and private sectors are facing the same growing challenge when dealing with computer forensics [2], which is to examine an increasing number of digital devices (e.g., GPS gadgets, smartphones, routers, embedded devices, SD cards), each containing an immense volume of data, in a timely manner and with limited resources. At the same time, with proliferation of low cost and easy-to-access anti-forensic techniques (sometimes open source as well), offenders are becoming increasingly sophisticated and skillful at concealing information.

Computer forensic investigators and examiners are confronted with the problems of, (i) unacceptable backlog of information waiting for examination; (ii) miss of critical time window to follow the leads due to slowness of computer

forensic examination; (iii) lack of understanding of the computer forensics and consequent incapability by the detectives to take advantages of digital forensic techniques to advance investigations; and (iv) overlook of relevant data and waste of resources due to lack of understanding of crime investigations by the forensic examiners.

The cloud computing model provides ideal opportunities to solve these problems. Cloud computing is a rapidly evolving information technology that is gaining remarkable success in recent years. It uses a shared pool of virtualized and configurable computing resources (both hardware and software) over a network to deliver services, such as to host and analyze large datasets immediately. These resources and services can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing is almost everywhere. Governments, research institutes, and industry leaders are quickly adopting the cloud computing model to solve the increasing computing and storage demands. This trend has significant implications for digital forensic investigations.

However, current forensic research related to the cloud is mainly focused on the stage of data collection (e.g., [3]). The examination and analysis on the data are still performed on local machines instead of in the cloud. Extending the services to the cloud often calls for the external assistance and professional software/applications. Researchers have made efforts to build a forensic cloud. Sleuth-Hadoop [4] tries to integrate different forensic analysis tools into the cloud. However, Sleuth-Hadoop doesn't have the flexibility for the investigators to build and customize the desired analysis workflow for specific forensic datasets.

The main contribution of our work is to fill the gaps. We propose a domain specific cloud environment for forensic applications. We designed a cloud infrastructure framework for dealing with large forensic datasets, sharing forensic software, and providing a way for the investigators to build workflows using a common interface. We proposed a schema-based forensic analysis workflow framework. The framework allows the forensic investigators to define their requirements in XML configuration files. Supported with a collection of forensic applications, the framework can select the appropriate applications, generate the corresponding map-reduce drivers, and set up the workflow in the cloud, automatically for the

users.

The rest of this paper is organized as follows. Section II presents the system design of the forensic cloud. Section III shows the experimental results. Related works are discussed in Section IV. Section V concludes the paper.

II. BACKGROUND

Four categories of cloud computing are defined by NIST (National Institute of Standards and Technology) [5], i.e., private cloud, community cloud, public cloud, and hybrid cloud. Currently, most research focuses on the community cloud and public cloud.

In the community cloud study, there are many solutions proposed for data sharing and collaborations. At Hewlett-Packard Labs, Erickson et al. [6] use a cloud-based platform to provide content-centered collaboration in the Fractal project. Social sharing of workflows are studied by Roure et al. [7]. Globus Online [8] focuses on data-movement functions to deal with new challenges brought by data-intensive, computational, and collaborative scientific research through cloud-based services. Compared with these studies, our work mainly concentrates on the workflow management in computer forensics and domain specific cloud infrastructure. Various kinds of other community cloud are also studied, e.g., volunteer cloud [9], [10], Nebula cloud [11], social cloud [12]. However, none of those is specifically designed for computer forensics. For domain specific applications, the one size fits all approach would not work because the specific characteristics and requirements from each application domain often demand customized solutions built on top of the cloud infrastructure.

In the public cloud, since users have different purposes to run their applications, studies mainly focus on the general-purpose resource management. For example, public cloud such as Amazon EC2[13] uses a scheduler in Xen hypervisor to schedule virtual machines. Song et al. [14] proposed a multi-tiered on-demand resource scheduling scheme to improve resource utilization and guarantee QoS in virtual machine based data centers.

One of the most popular programming models in the cloud is MapReduce [15], which is for distributed processing of large-scale data on clusters of commodity servers. Ananthanarayanan et al. [16] proposed an optimized cluster file system for MapReduce applications. They use metablock that is a consecutive set of blocks of a file that are allocated on the same disk instead of the traditional cluster file system. Apache Pig [17] is a platform for analyzing large data sets using MapReduce on the top of Hadoop.

Digital forensics are performed in four phases [2], i.e., collection, examination, analysis and reporting. The investigators will execute the following separately, 1) identifying, recording, acquiring data from possible sources, while preserving the integrity of the data; 2) processing the data with a combination of manual and automated methods, and extracting data of particular interest; 3) analyzing the results of the examination with legally justifiable methods and techniques to derive useful information; 4) describing the results of the analysis.

Forensic software provides many different kinds of tools to investigate suspicious servers, desktops, and personal digital devices such as cell phones, GPS navigators, PDAs, etc. The investigations mainly focus on discovering forensic evidence, and identifying suspicious files and activities. Bulk_extractor [18] can scan suspicious files and email and extract data from the disk images, files, and directories. Many comprehensive tools, such as FTK [19], OSForensics [20], Intella [21], etc., provide the investigation functions. However, they are stand-alone software running on local machines. Supports for inter-operations and large scale automated parallelization are poor, or almost none. Open Computer Forensics Architecture (OCFA) [22] is an automated system that can extract metadata from files, create indices for the target disk images and ultimately output a repository containing the files and indices for further examination. OCFA is able to work with other third part analysis software or data mining tools. The limitation of the OCFA is that it is not integrated with the cloud.

Sleuth Kit [23] has a cloud-based version, Sleuth Hadoop, which integrates several forensic software and enables them to run in the cloud. However, the analysis workflow is fixed in Sleuth Hadoop [4] without the capabilities to configure and construct workflow dynamically. It doesn't support collaborative software development and workflow management.

III. SYSTEM DESIGN

A. System Overview

The forensic cloud infrastructure aims to deliver the services that go beyond today's models of "software-as-a-service" and "infrastructure-as-a-service", with the goal of providing not only elastic computing resources for on-demand computer forensic data processing, but also an environment for intelligent forensic workflow management, customization, and collaboration.

The forensic cloud comprises two main layers: a service layer and a physical resource layer, as shown in Figure 1. The service layer has three major components, the forensic data manager, the forensic application manager and the forensic workflow manager. The physical layer is composed of physical devices such as accelerators, physical servers, and storage servers for supporting forensic data banks. A set of virtual machines can be allocated for serving a particular forensic data processing task.

B. Forensic Data Manager

Forensic data manager provides supports for uploading, storing, and retrieving the large-scale forensic data in the cloud. Forensic data are collected from diverse sources (e.g., disks, cellphones, embedded devices). With elastic storage resources provided by the cloud, forensic investigators can process, analyze, and archive forensic data with reduced cost, improved efficiencies, and increased productivity.

Considering the scale of the data and the fact that most applications in the cloud use MapReduce [15] for parallelizing the applications and performing the analysis on the

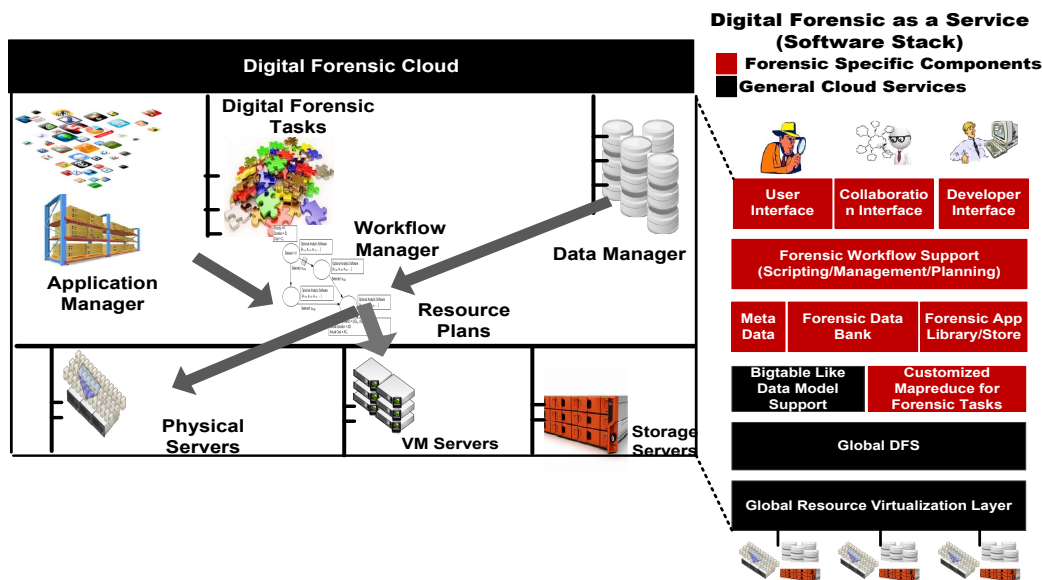


Fig. 1. Forensic Cloud Overview and Software Stack

data, the data manager uses HDFS (Hadoop Distributed File System) [24] to store the data. HDFS is a distributed file system designed to work on commodity hardware maintained as a Hadoop subproject. HDFS stores all the data in blocks. The block size is usually 64MB or 128MB. HDFS works more efficiently if the single file size is larger than the block size, which, however, is not necessarily always the case for all the files in a target disk image. To avoid the small-file problem, the data manager organizes the files in HAR files or SequenceFile formats [25]. Creating a working copy, is managed by the forensic data manager as well. The forensic data manager also flattens all the directory information, which exports all the nested files into one folder. This can mitigate the anti-forensic (AF) approach called, “circular references”. The “circular references” exploit uses symbolic links to point to a parent folder, which may make a search operation run for ever.

In addition, the data manager also maintains the metadata of the files in the HBase (an open-source, distributed, versioned, column-oriented store modeled after Google’s Bigtable [26]). The metadata contains useful data for the files, for instance, the directory structure information before flattening, the hash values (MD5) of the files. The information is often used in analyzing the forensic data. For example, National Software Reference Library (NSRL) [27] provides a comprehensive database with the hash values for almost all the commercially available software. This provides a Reference Data Set (RDS) of information [27], which can be used as digital signatures of the known, good software applications. Therefore, by comparing the hash values of the files in a target disk with the database, the investigators can filter out all the uninterested files. This Known File Filter (KFF) operation can significantly reduce the sizes of the data that requires examination. All other similar metadata are calculated by the data manager and stored in the HBase. This is a default step when new files are uploaded to

the forensic cloud and to be ingested.

With the help of the universal management of the data, forensic analysis and data mining experts who develop software for forensic data processing only need to submit their software to the cloud.

C. Forensic Application Manager

Forensic applications and software such as files/emails search, image/videos analysis, etc. are created through collaborative processes involving many forensic experts and computer science researchers. To accelerate productivity and expedite collaborations among them, it is necessary to reuse the software and workflow. Forensic software vendors can distribute the developed algorithms and software to a software/app library, the “forensic app store” where forensic workflow can be constructed using these software. Forensic examiners and investigators can on-demand create, invoke, and deploy tasks using the forensic software and workflow stored in the library. Consequently, the infrastructure will accelerate dissemination and deployment of new forensic techniques.

All the applications in the “forensic app store” are tagged and categorized by the application manager. The application manager periodically generates an XML schema and metadata for all the available software. The schema is used to generate a user-friendly front-end web page (maintained by the workflow manager) and to validate the XML-based workflow configuration file.

An example schema file and xml configuration file are shown in Figure 2. In the schema file on the left of Figure 2, all the four applications available in the “app store” are listed. The digital forensic front-end web page can read the schema file and generate a drop-down list with these applications when a forensic investigator selects the applications. The investigators only need to click several buttons to generate an XML configuration file as shown on the right bottom of

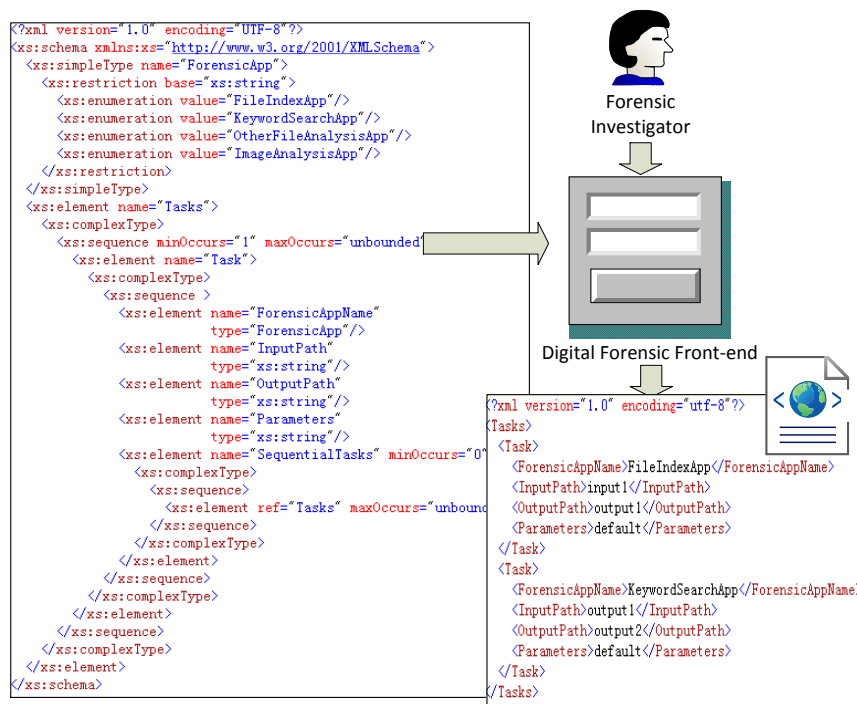


Fig. 2. An example of the schema XML generated by application manager and the XML workflow configuration file generated by the workflow manager. The file on the left is the schema XML listing all the four applications and the desired structure of the work configuration file; the file on the right bottom is the XML configuration file with two tasks.

Figure 2. This configure file is used by the workflow manger to generate MapReduce drivers and workflow assembly. For more advanced investigators, they can directly write the XML configuration file and use the schema to valid the file. This will reduce the chances of creating an invalid file. In reality, there could be more categories than the example provided.

The application manager provides a set of default categories of the applications, including FileIndexApp, KeywordSearch App, ImageAnalysis App, etc. Users can also add customized tags and categories into the cloud when uploading the new applications. In addition, more tags and supplementary categories could be created by users. Users are allowed and encouraged to rate the applications after using. The ratings are further used for the application recommendation. The applications are sorted from the highest rating to the lowest in the generated XML file. Therefore, highly qualified applications will be presented to users at the top of the candidate application list. The user ratings are the key criteria to evaluate the applications.

The application manager also provides recommendations. Currently, it is community oriented. Each application will be rated by all the users who have tried it. When the application manager generates the schema file, the rating information will be included. Therefore, when users select the application, they are aware of the information that can be used to evaluate the candidate applications.

D. Forensic Workflow Manager

Forensic investigators can send data processing jobs to the cloud. For example, an investigator can specify, the objectives of data processing, the input dataset (stored in the cloud using forensic data manager), and other constraints. The cloud can create a workflow by decomposing the user’s request into multiple processing steps. The workflow manager is responsible for setting up, optimizing, executing and reporting the workflow.

1) *Workflow Setup*: The workflow manager represents a workflow using an XML configuration file. The structure of this XML file is defined in the schema file generated by the application manager. Generally, the schema file contains two kinds of information. One is for all the available applications or software in the “application store”, which are defined in a simple type (xs:simpleType) or a complex type (xs:complexType); the other is the root element structure, called “tasks”. The “tasks” may contain one or more “tasks”, each of which needs the application name, input path, output path, and parameters for execution. All the tasks on the same level are independent and can be executed in parallel. If a user would like to define the dependency between two tasks, the second task should be configured as a “sequential task” of the first task. Figure 2 shows an example. Complex workflows can be also described by assigning the subtasks, which can be recursively built with arbitrary levels of dependencies.To facilitate the procedure of setting up a forensic workflow, the workflow manager uses the schema file to generate a user-

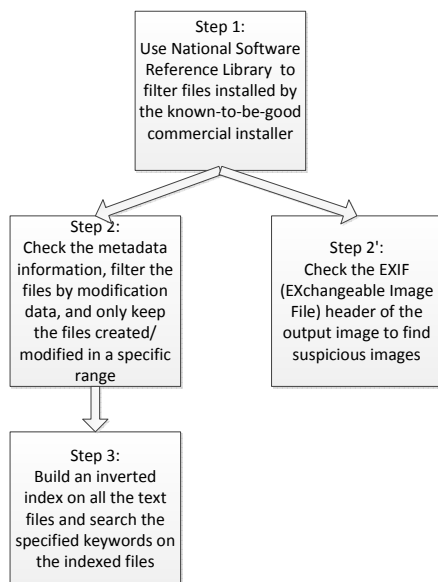


Fig. 3. A Workflow Example Constructed by the Workflow Manager friendly web portal, which allows forensic investigators to design the workflow and select the desired applications. After designing the workflow, the frontend will pass the workflow to the backend engine. This engine will generate an XML configure file and further generate the Map-Reduce drivers for each step and the necessary synchronization codes (if multiple steps are involved in the workflow) automatically for the forensic investigators. The fewer lines of codes to write, the less chance to generate errors.

2) *Workflow Recommendation*: Since each step could be completed by multiple candidate software with data dependent performance metrics, the workflow manager will try to make optimal selection/recommendation of software/workflow and allocate resources accordingly with the objective of achieving the best performance (result quality) for the input dataset with the help of user ratings and the pre-defined workflows. For example, the workflow manager recommends building indices before keyword search. Another example is that by default, the workflow manager will select the National Software Reference Library (NSRL) to filter out the typical contents created by the commercial installer, such as dll, exe, static data. The recommendations are based on the user ratings and evaluation. An example is shown in Figure 3.

3) *Workflow Execution*: To execute the workflow, the workflow manager allocates processing resources such as elastic machine hours based on an optimized resource plan and assigns workload to the allocated resources using the MapReduce model customized for data intensive forensic computations. Then, the allocated resources execute the assigned tasks on datasets retrieved from the cloud forensic data banks administrated by the data manager. The workflow manager will direct the workflow execution and track the status of each task in the workflow.

4) *Workflow Report*: Finally, after finishing the workflow, the workflow manager will generate a report to the users. In addition, the workflow manager also stores the status and report in its own database.

IV. EVALUATION

In this section, we present the results of a comprehensive evaluation of our system.

A. Experimental Setup

During our evaluation, we deployed a forensic cloud as described earlier using the Amazon' Elastic Compute Cloud(EC2) service. The deployment uses Medium Level-1 (M1) EC2 instances. According to Amazon, these are 64-bit instances with 3.75 GB of memory, 410GB of harddisk and one virtual core containing two EC2 compute units (ECU). One ECU is equivalent to a 1.0-1.2 GHz Xeon processor. The forensic cloud infrastructure is based on Hadoop 0.20 and HBase 0.20, which is managed by Cloudear Manager [28]. The data from a volunteer's hard drive image was uploaded to the forensic cloud. Notice that, the uploading time is not counted and evaluated in the following experiments. This is because, as mentioned previously, the data used are collected from different sources in a distributed way using the cloud as well. We simplified the process by uploading a dedicated image disk for studying purpose. Therefore, the uploading time is not considered.

B. Experimental Results

First, we compared the system outputs and analyzed the performance using the same disk image dataset, which is a working disk image from volunteer users. Figure 4 shows the forensic analysis time on the target image. The image size is 160GB. It shrinks to 10GB after applying the filer operations mentioned in the previous sections. The number of nodes used in the experiment increases from 1 to 10. With more nodes involved, the analysis time is reduced from 21 minutes to only 6 minutes, i.e., 71% of analysis time is saved. However, given a fixed size of test data, the analysis speed can't be further accelerated by adding more nodes. As shown in Figure 4, the forensic cloud with more than 8 nodes has almost the same performance. This is because when more nodes are involved, some of the MapReduce tasks are not executed at the same machine where the data are stored. Copying data between nodes cuts down the benefits. Figure 5 shows the percentage of the MapReduce tasks running locally. The percentage drops from 100% to 40% when the number of nodes changes from 2 to 10. This explains why the speedup of analysis time is only 3. However, when the size of data to be analyzed keeps increasing, more time can be saved, because more data blocks can be processed locally. As shown in Figure 6, when the size of the data increases by 200%, i.e., the size is tripled, the analysis time only increases by 100%. This gives us the clue that the forensic cloud can save more time when dealing with large amount of data.

In the second set of experiments, we compared the lines of codes (LoC) that is needed for the configuration with and without the workflow manager. Figure 7 shows how much effort could be saved in terms of LoC. Workflows with different sequential tasks are built up. Without workflow manager, to configure one workflow task, on average 40 LoCs

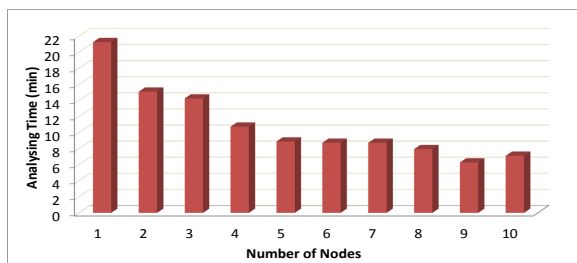


Fig. 4. Analysis Time under Different Number of Nodes in the Cloud

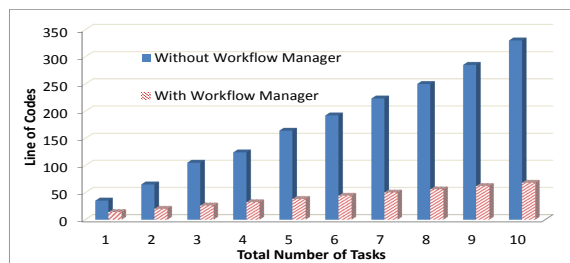


Fig. 7. Line of Codes Needed to Configure Different Tasks in a Workflow w/ and w/o the Dataflow Manager

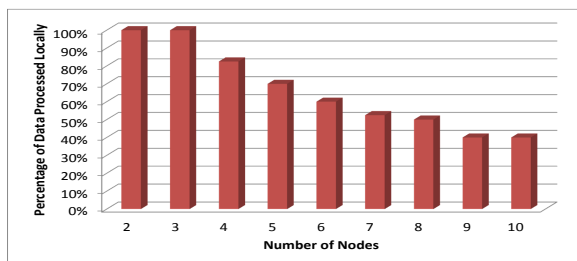


Fig. 5. Percentage of the MapReduce Tasks Processed Locally under Different Numbers of Nodes

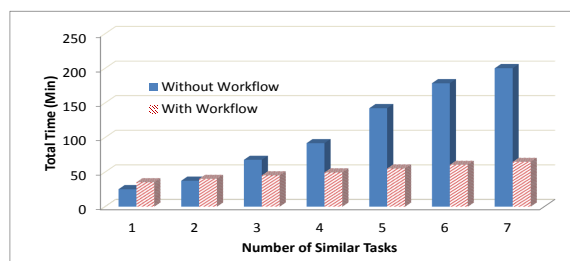


Fig. 8. Total Time Spent w/ and w/o the Workflow Manager's Optimization

are needed, but only 4 LoCs are actually required for the workflow XML file. The LoCs can be reduced by 90% when using the workflow manager to configure a forensic data processing task.

We further compared the performance with and without optimization performed by the workflow manager. We have ten similar tasks, i.e., searching for some keywords, in our experiments. The workflow can intelligently add an extra step of building indices before running all the ten tasks. As shown in Figure 8, the analysis time increases linearly with the number of tasks without the help of workflow management. With the workflow management and optimization, the total time is a little more than the time spent without the workflow management if there is one task executed. However, the total execution time increases slightly when more similar tasks are executed. This is because when the indices are built, further keyword search operations will be accelerated dramatically by the indices stored in the HBase.

V. CONCLUSIONS

We proposed and implemented a domain specific cloud environment for digital forensics. We designed a cloud based framework for supporting automated forensic workflow management and data processing. A schema-based forensic workflow framework is proposed. The experimental results show that using the proposed forensic cloud services can save at least 71% of the time with only 10 virtual machine nodes. Meanwhile, the lines of codes for specifying a workflow are also reduced to only 10% when using the proposed workflow management approach. For the investigators, it could be even easier by using the web-based portal, clicking buttons and selecting the desired applications from the dropdown lists. The automated and optimized workflow management approach can save 87% of the analysis time in the tested scenarios. The proposed framework provides valuable insights on designs of domain specific cloud environments using computer forensics as a target field. It demonstrates that, in addition to providing elastic computing resources, cloud can be used as an environment for workflow management and coordinated software development.

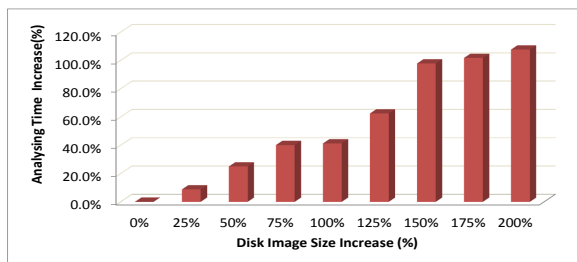


Fig. 6. Percentage of the Increased Analysis Time under Different Increased Image Sizes

VI. ACKNOWLEDGEMENT

We would like to thank the reviewers for their comments which significantly improved the paper. This research is partially supported by the Department of Homeland Security (DHS) under Award Number N66001-13-C-3002, and the National Science Foundation under Award Number CNS 1205708. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the opinions or policies of DHS or NSF.

REFERENCES

- [1] A. of Chief Police Officers, "Good practice guide for computer based electronic evidence," ACPO, Tech. Rep.
- [2] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to integrating forensic techniques into incident response," National Institute of Standards and Technology, Tech. Rep.
- [3] J. Dykstra and A. T. Sherman, "Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques," *Digital Investigation*, vol. 9, 2012, pp. S90–S98.
- [4] "Sleuth Hadoop," http://www.sleuthkit.org/tsk_hadoop/, retrieved April 2013.
- [5] P. Mell and T. Grance, "The NIST definition of cloud computing," <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.
- [6] J. Erickson, M. Rhodes, S. Spence, D. Banks, J. Rutherford, E. Simpson, G. Belrose, and R. Perry, "Content-centered collaboration spaces in the cloud," *IEEE Internet Computing*, vol. 13, September 2009, pp. 34–42.
- [7] D. D. Roure, C. Goble, and R. Stevens, "The design and realisation of the myexperiment virtual research environment for social sharing of workflows," *Future Generation Computer Systems*, vol. 25, no. 5, 2009, pp. 561 – 567.
- [8] I. Foster, "Globus online: Accelerating and democratizing science through cloud-based services," *Internet Computing, IEEE*, vol. 15, no. 3, May-June 2011, pp. 70 –73.
- [9] S. Caton and O. Rana, "Towards autonomic management for cloud services based upon volunteered resources," *Concurrency and Computation: Practice and Experience*, 2011.
- [10] S. Distefano, V. D. Cunsolo, A. Puliafito, and M. Scarpa, "Cloud@home: A new enhanced computing paradigm," in *Handbook of Cloud Computing*, B. Furht and A. Escalante, Eds. Springer US, 2010, pp. 575–594.
- [11] A. Chandra and J. Weissman, "Nebulas: using distributed voluntary resources to build clouds," in *Proceedings of the 2009 conference on Hot topics in cloud computing*. USENIX Association, 2009.
- [12] S. Xu and M. Yung, "Socialclouds: Concept, security architecture and some mechanisms," in *Trusted Systems*, ser. *Lecture Notes in Computer Science*, L. Chen and M. Yung, Eds. Springer Berlin / Heidelberg, 2010, vol. 6163, pp. 104–128.
- [13] "Amazon EC2," <http://aws.amazon.com/ec2/>, retrieved April 2013.
- [14] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-tiered on-demand resource scheduling for vm-based data center," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. *CCGRID '09*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 148–155.
- [15] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, Jan. 2008, pp. 107–113.
- [16] R. Ananthanarayanan, K. Gupta, P. Pandey, H. Pucha, P. Sarkar, M. Shah, and R. Tewari, "Cloud analytics: do we really need to reinvent the storage stack?" in *Proceedings of the 2009 conference on Hot topics in cloud computing*, ser. *HotCloud'09*. Berkeley, CA, USA: USENIX Association, 2009.
- [17] "Apache Pig," <http://pig.apache.org/>, retrieved April 2013.
- [18] "Bulk Extractor," https://github.com/simsong/bulk_extractor/wiki/Introducing-bulk_extractor, retrieved April 2013.
- [19] "FTK (Forensics Toolkit)," <http://www.accessdata.com/>, retrieved April 2013.
- [20] "OSForensics," <http://www.osforensics.com/>, retrieved April 2013.
- [21] "Intella," <http://www.vound-software.com/>, retrieved April 2013.
- [22] E. Huebner and S. Zanero, *Open Source Software for Digital Forensics*. Springer, 2010. [Online]. Available: <http://books.google.com/books?id=2g17k8PbIFYC>
- [23] "The Sleuth Kit," <http://www.sleuthkit.org/>, retrieved April 2013.
- [24] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, ser. *MSST '10*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–10.
- [25] "Apache Hadoop Wiki-Sequence File," <http://wiki.apache.org/hadoop/SequenceFile>, retrieved April 2013.
- [26] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: a distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, ser. *OSDI '06*. Berkeley, CA, USA: USENIX Association, 2006, pp. 15–15.
- [27] "National Software Reference Library," <http://www.nsl.nist.gov/>, retrieved April 2013.
- [28] "Cloudera," <http://www.cloudera/>, retrieved April 2013.

Fuzzy Subtractive Clustering Based Prediction Approach for CPU Load Availability

Kadda Beghdad Bey

Laboratoire de Systèmes Informatiques
Ecole Militaire Polytechnique
Algiers, Algeria
bey_kadda@yahoo.fr

Farid Benhammadi

Laboratoire de Systèmes Informatiques
Ecole Militaire Polytechnique
Algiers, Algeria
benhammadif@yahoo.fr

Faouzi Sebbak

Laboratoire de Systèmes Informatiques
Ecole Militaire Polytechnique
Algiers, Algeria
faouzi.sebbak@gmail.com

Abstract—Distributed processing environment has emerged as a new vision for future network based calculation, allowing the federation of heterogeneous computing resources to incorporate the power. Cloud computing is a new computing paradigm composed of a combination of grid computing and utility computing concepts. In cloud computing, the prediction methods play a key role in managing large scale of computation capacity. In this paper, a modelling approach to predict the future CPU load value is presented. The proposed approach employs a computational intelligence technique to classify the CPU load time series into similarity component group. This technique is based on the Fuzzy Subtractive Clustering algorithm and a combination of local Adaptive Network-based Fuzzy Inference System. The results of an exhaustive set of experiments are reported to validate the proposed prediction model and to evaluate the accuracy of their prediction. Experimental results demonstrate both feasibility and effectiveness of our approach that achieves important improvement with respect to the existing CPU load prediction models.

Keywords-Subtractive clustering; CPU load prediction; cloud computing; system modelling; ANFIS.

I. INTRODUCTION

Heterogeneous computer network environments involve effective utilization of the distributed resources to achieve high performance computing. Cloud computing is a new computing paradigm composed of a combination of grid computing and utility computing concepts. Cloud promises high scalability, flexibility and cost-effectiveness to satisfy emerging computing requirements; therefore, they can treat task scheduling and resource allocation over the virtual clusters [1]. In the literature, various architectures have been proposed to satisfy the user's needs in terms of computational power through the use of distributed computing resources [2]. In distributed environments, resources monitoring needs continual parameters monitoring in terms of CPU load, memory size, bandwidth and latency. Irrespective of the nature and the type of the used distributed processing environment, the creation of resource pools should satisfy several requirements for each parameter quality during the computation service. To efficiently provision computing resources in the cloud, the ability to accurately predict resource capabilities is of great importance since it permits to determine how to use time-shared resources.

Many interesting modelling strategies have been proposed to predict available CPU load in a grid computing environment [3,4,5]. The main contribution of the present paper relies on the integration of the subtractive clustering technique and the Fuzzy Inference System (FIS) to make short and medium-term predictions of CPU availability on time-shared environment systems. The proposed approach predicts the future value of CPU load based on a set of local Adaptive Network-based Fuzzy Inference System (ANFIS) predictors to perform short-term accurate and mid-term reliable prediction using the selection instances in several past steps. We also propose a deterministic approach for k-folds cross-validation that constructs representative rather random folds. Through this approach, we attempt to reduce the effects of using only a few instances for training.

The rest of this paper is organized as follows. Section 2 reviews the related works about CPU load prediction approaches in time-shared systems. Section 3 presents the proposed subtractive clustering-based ANFIS prediction model. This section also describes how this software is used to carry out experiments. Experimental results are reported in Section 4. Conclusions and directions for future work end the paper.

II. RELATED WORK

A Cloud computing platform offers to users a virtualized distributed system, where computing resources are dynamically allocated to satisfy a user's Service Level Agreement. Predicting the processor availability for a new process or task in computer network systems is a basic problem arising in many important contexts. Making such predictions is not easy because of the dynamic nature of current computer systems and their workload.

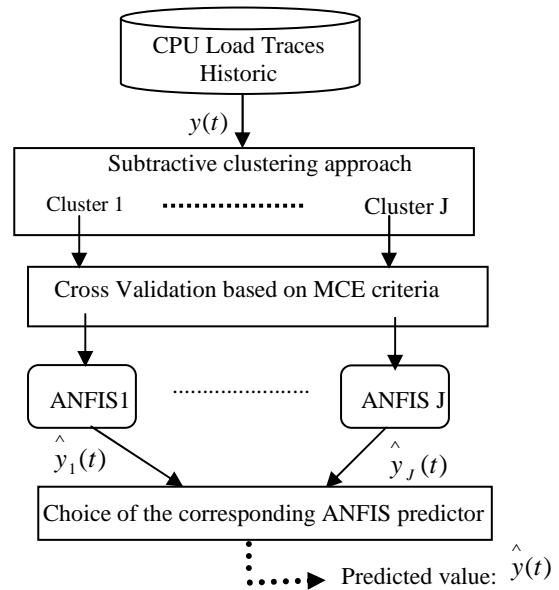
The Network Weather Service (NWS) [3] is the most famous system designed to provide dynamic resource performance forecasting. The predictive methods currently used in NWS include running average, sliding window average, last measurement, adaptive window average, median filter, adaptive window median, α -trimmed mean, stochastic gradient, and auto-regression (AR). Dida [6] studied different linear series models including autoregressive, moving average, autoregressive moving average, autoregressive integrated moving average and autoregressive fractionally integrated moving average models, for predicting future loads from 1 to 30 seconds. Huo et al. [7] evaluated four criteria to determine the

optimal order of AR models: Final Prediction Error (FPE), Akaike's Information Criterion (AIC), Minimum Description Length (MDL) and Bayesian Information Criterion (BIC). The authors claimed that the BIC criterion performs better than other criteria. An approach based on the Tendency-Based and Polynomial fitting method predictor is proposed by Yang et al. [8]. Liang et al. [9], presented a more-generic prediction scheme using both the autocorrelation of CPU load and the cross correlation between CPU load and free memory to achieve higher CPU load prediction accuracy. In [10], Zhang et al. tackled the problem of predicting available CPU performance in a time-shared grid system. Their strategy forecasts the future CPU load based on the variety tendency in several past steps and in previous similar patterns. Recently, non linear models have been tried for time series prediction [11,12,13]. Liu et al. [13] proposed a hybrid non-linear time-series segmentation algorithm to discover duration-series pattern. In the experiment, they compared six approaches including LAST, MEAN, Exponential Smoothing, Moving Average, AR and Network Weather Service.

The present framework is related to our prior efforts in CPU load prediction and complements the existing performance CPU load prediction schemes [11, 12] with a modification of the soft computing algorithm using a subtractive clustering method. The new prediction system combines the subtractive clustering method and ANFIS. A strong point of our model is that it contains the same set of predictors which are able to deliver accurate prediction in peaks, switch level and regular situations.

III. SUBTRACTIVE CLUSTERING-BASED ANFIS PREDICTION

Cloud computing has become a great solution for providing a flexible and dynamically scalable computing infrastructure for many applications. Cloud computing presents a significant technology trends, and it is already obvious that it is reshaping information technology process [19]. To realize the next generation of distributed computing, we need to be able to accurately predict resource utilization. In this work, we proposed a novel model to predict the behavior of computing resources. Fuzzy models have been shown to be very effective techniques for the modelling of nonlinear, uncertain and complex systems. Subtractive Clustering is a fast one-pass algorithm for estimating the number of clusters and determining the cluster centres in a set of data [14]. We use the subtractive clustering if we do not have a clear idea about how many clusters should be used for a given data set. After clustering the data set, the number of fuzzy rules and premise fuzzy membership function are determined. Then, the linear squares estimate is used to determine the consequent in the output membership function, which provides a valid fuzzy inference system (FIS). The proposed approach includes three major steps: CPU load time series clustering, the ANFIS clusters model prediction and the combination of local ANFIS prediction model. As shown in Fig. 1, before making predictions about future CPU load values, subtractive clustering is applied to divide the historic CPU load data into sub-clusters and generate more homogeneous data.



$\hat{y}(t)$: CPU load time series.
 $y_j(t)$: Output from ANFIS J at time t.
 MCE : Minimum Checking Error

Figure 1. Subtractive clustering-based ANFIS prediction.

A. CPU load time series clustering

The purpose of this step is to identify natural groupings of CPU time series from a large set of historic traces, and to produce a concise representation of the system's behaviour. For our problem, one does not have a clear idea about the number of clusters to be used for a given set of data. Subtractive clustering technique, proposed by Chiu [14], has been shown to be a fast way of estimating the number of clusters and their centres positions. This technique calculates the density function based on the positions of data points, which leads to a significant reduction of the number of calculations. Each data point is a candidate to become a cluster centre. A density measure at data point x_i is defined as:

$$D_i = \sum_{j=1}^n \exp \left(- \frac{\|x_i - x_j\|^2}{(r_a/2)^2} \right) \tag{1}$$

where r_a is a positive constant representing a neighbourhood radius. Hence, the more neighbouring points a data point has, the higher is its density. The density measure of each data point x_i is defined as follows:

$$D_i = D_i - D_{c_1} \exp \left(- \frac{\|x_i - x_j\|^2}{(r_b/2)^2} \right) \tag{2}$$

where r_b is a positive constant that defines a neighbourhood that has measurable reductions in density measure. Thus, the data points near the first cluster centre x_{c_1} will have significantly reduced density measure.

After updating the density function, the next cluster centre is selected as the point having the highest density value. This process continues until a sufficient clusters number is attained. Fig. 2 shows an example of CPU time series clustering based on the subtractive clustering method.

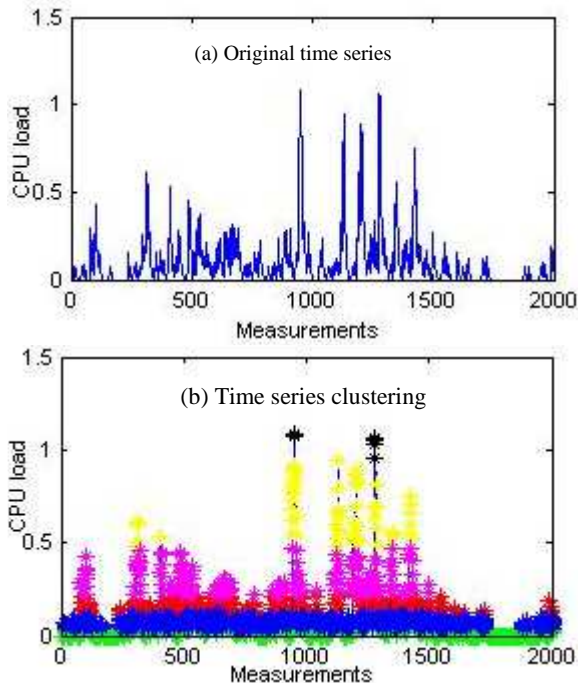


Figure 2. An example of CPU loads time series clustering.

For CPU load time series clustering, we use known values of the dynamical situation of the historic data up to time t . Let $Y(t)=\{y_1, y_2, \dots, y_t\}$ be the time series at time t . The dynamical situation Δy_t at time t is defined as follows:

$$\Delta Y_t = \{y_t - y_{t-1}, y_{t-1} - y_{t-2}, \dots, y_2 - y_1\} \quad (3)$$

The Subtractive clustering technique is used to cluster all time series y_t into clusters. It estimates the number of clusters and the cluster centres. This process assigns the CPU load data y_t using the cluster membership degree μ_j that represents the degree to which y_t belongs to cluster c_j . This assignment is computed using the following objective minimization function:

$$O = \sum_{j=1}^J \sum_{t=1}^{t=T} (\mu_j(t))^2 \|\Delta y_t - v_j\| \quad (4)$$

where v_j is the centre of cluster j and J is the number of clusters.

B. ANFIS Predictor

The Adaptive Network-based Fuzzy Inference System (ANFIS) proposed by Roger Jang [15] is one of the most commonly used fuzzy inference systems. It is a universal approximator used in various applications of predictions.

Moreover, it has been proven to be more powerful than other models for short term prediction. ANFIS is a 5-layer feed-forward network in which each node performs a particular function in incoming signals, as well as a set of parameters pertaining to that node. Similarly to ANFIS, the compensatory neural fuzzy network with n -dimensional input-data vector x_p and one-dimensional output-data vector y_p has 5 functional layers: input layer, fuzzification layer, pessimistic-optimistic operation layer, compensatory operation layer (fuzzy reasoning method) and defuzzification layer.

Let us suppose that the fuzzy inference system under consideration has four inputs and one output. If two fuzzy sets are associated with each entry variable, then the system presents 16 inferences rules R_j (24), that are of the first-order Sugeno fuzzy type:

$$R_j : \text{if } (x_1 \text{ is } A_{1j}) \text{ and } (x_2 \text{ is } A_{2j}) \text{ and } (x_3 \text{ is } A_{3j}) \text{ and } (x_4 \text{ is } A_{4j}) \text{ Then } y_j = f_i(x) = c_1jx_1 + c_2jx_2 + c_3jx_3 + c_4jx_4 = B_j \quad (5)$$

These rules correspond to the third category of fuzzy inference systems mentioned in [16]. One of the most important stages of the Neuro-fuzzy TSK (Takagi-Sugeno-Kang) network generation is the establishment of inference rules (Takagi and Sugeno 1985) [17] often used is the so-called grid method, in which the rules are defined as the combinations of the membership functions for each input variable.

C. Future CPU load Prediction

In this study, Adaptive Network-based Fuzzy Inference System based subtractive clustering has been used to predict availability of the CPU load. In our previous works [11, 12], a simple method for accuracy estimation is used. The dataset is randomly portioned in two disjoint subsets of $N/2$ instances. The first subset serves as the training set and the second one as the test set. The drawback of this method is that it makes inefficient use of data since typically a relatively large proportion of the instances is used for testing [18]. Cross-validation attempts to resolve this drawback by successively removing some instances from the initial set, treating them as a test set. In k -fold cross-validation, the dataset is randomly partitioned into k disjoint blocks (folds), of approximately equal size d ($d \approx N / k$). The learning algorithm runs k times. In the i^{th} iteration, the i^{th} training set is formed by the initial dataset without the i^{th} fold, while the test set is formed using the i^{th} fold alone [18]. The aim of directing similar instances to different folds is to reduce the pessimistic effects caused by the removal of instances from the dataset. The principle for constructing representative folds in unsupervised stratification is to channel similar instances to different folds in order to reduce the effects of using fewer instances for training.

For the final decision of CPU load time series prediction, we have used cluster predictor to select the adequate ANFIS predictor. After the application of the subtractive clustering method above the dataset, the instance space is partitioned into clusters. The next step is to determine the appropriate cluster, which aims at predicting future CPU load cluster based upon the observed history. The appropriate cluster for final

decision of CPU load prediction is defined by the largest similarity between the cluster centres and the input times series points, as show in Fig. 3.

```

For each time series point  $X_i$ 
Find the cluster centres  $C_j$ 
 $C_c =$  the closest centre to  $X_i$ 
For  $j=1$  to  $J$     $J$ : number of cluster
    /*Calculate the similarity  $Sim$  between
        the centre  $C_k$  and  $X_i$ 
         $S = Sim ( X_i, C_k)$ 
End
/* Find the largest similarity  $S_L$  between
 $X_i$  and all other centres
 $S_L = Max(Sim( X_i, C_k))$ 
 $C_c = C_k$ 
End
    
```

Figure 3. Selection of appropriate cluster

IV. EXPERIMENTAL RESULTS

In the previous section, we have presented a new prediction approach for CPU load availability. In the present one, we assess its performance with respect to other methods. For this purpose, we carry out series of experiments on different CPU load time series with a variety of statistical properties collected by Dinda [19]. These CPU load traces were collected for two time periods on roughly the same group of machines. The traces used are in two column whitespace-delimited ASCII format. The first column gives the time stamp in seconds whereas the second one provides the floating point measured load value.

A. Prediction model validation

To generate a FIS using ANFIS, it is important to select the number of Membership Functions (MF) and the proper parameters for the learning and refining process. For training and testing data sets, we analyse the effect of these parameters on the final ANFIS performance including the training and testing minimum checking error (MCE). We evaluate and compare our prediction model with previous approaches using the Normalized Mean Square Error (NMSE) defined by:

$$NMSE = \frac{\sum_{t=1}^T (y_t - \hat{y}_t)^2}{\sum_{t=1}^T \left(y_t - \frac{1}{T} \sum_{t=1}^T y_t \right)^2} \quad (6)$$

where \hat{y}_t represents the CPU prediction value, y_t the actual measurement, and T the number of time series points.

The proposed ANFIS prediction model is based on the subtractive clustering process that resolves the problem of clusters number used for each CPU load time series. Though, this method determines the optimal number of cluster for each CPU load traces. Table 1 summarizes the prediction results of the CPU load time series from the proposed prediction model for four different machines traces collected by Yang [20]. This table shows that the

Subtractive Clustering-based ANFIS model achieves better performance than other strategies for the same four load traces. The converged RMSE is much smaller than for the models reported in [11,12].

TABLE 1. NMSE FOR DIVERSE CPU LOAD PREDICTION

Prediction of future value $x(t+1)$	NMSE	Error % (min/max)
<i>axp0Aug.180</i>	0.056297	9,44 / 10,7
<i>Abyss.1000</i>	0.031459	1,13 / 3,06
<i>Mystere.10000</i>	0.26987	6,18 / 10,02
<i>axp1Aug.120</i>	1.185	6,38 / 45,99

We also tested some other prediction models including ours, ANFIS without clustering and Mixture of ANFIS. Fig. 3 illustrates a comparison between these three prediction models for five machines using different CPU load time series. The Mean Error Prediction of the proposed subtractive clustering based-model is smaller than that of other models. The predictive results of one traces machines using the Subtractive Clustering-based ANFIS model are shown in Fig. 4. The obtained prediction mean error was 0.08% whereas the RMSE is less than 0.15%. This shows again the consistent improvements of the proposed approach on the prediction quality over the corresponding time series collected on these machines.

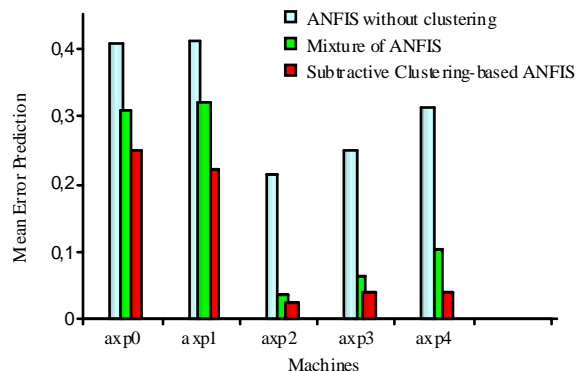


Figure 3. Comparison of three CPU load prediction models

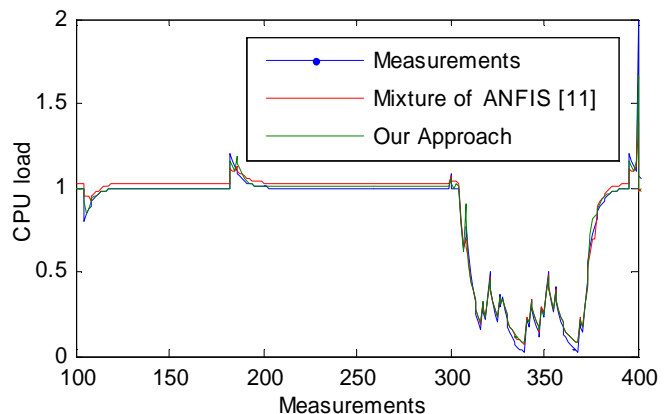


Figure 4. Comparative results of our predictor with Mixture of ANFIS [11].

B. Comparisons with other models

To evaluate the performances of the proposed prediction approach with respect to the existing ones, we have assembled test data from multiple datasets. The results of the subtractive clustering-based ANFIS prediction model on all the test time series are illustrated in Fig.5. These results show that the proposed prediction model performs well in general. The results of the approach based on Mixture of ANFIS [12] are better for various host traces. Therefore, it can be concluded that our model gives a good prediction on most of the host's time series and outperforms then the models reported in [10,12].

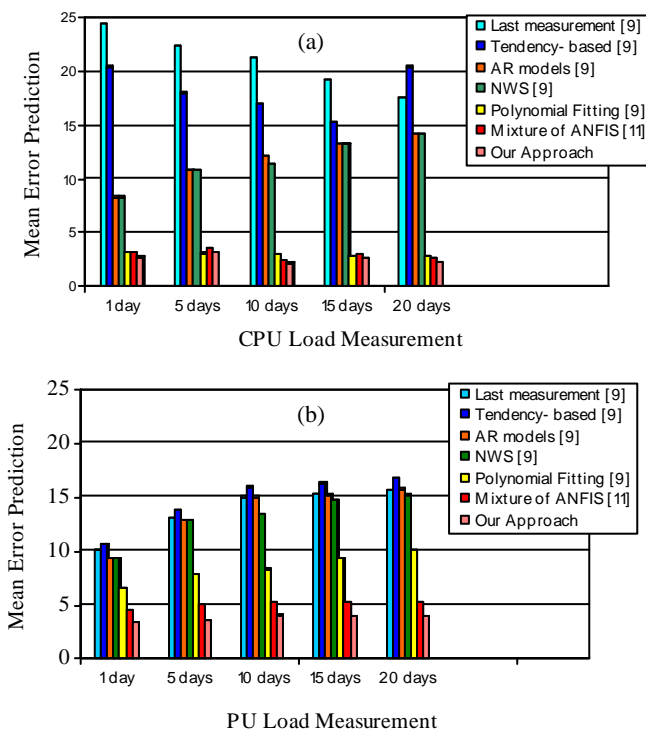


Figure 5. Mean error prediction of several models

V. CONCLUSION AND FUTURE WORK

Performance prediction is set to play a significant role in the resource management and distributed systems. Clouds computing are designed to provide services to external users, providers need to be compensated for sharing their resources and capabilities. The contribution of this paper is a new modelling approach to predict CPU load future value in distributed computing. This approach employs subtractive clustering technique to classify the CPU traces into similarity component group and a combination of local ANFIS. The proposed prediction model is validated and checked with a set of exhaustive experiments performed on a set of real and representative CPU load traces. In addition, we have shown that a significant reduction in prediction errors is experienced using the subtractive clustering-based ANFIS model since it always computes accurate predictions.

Predicting resource utilization is a fundamental need when running a virtualized system. It is necessary because cloud infrastructures use virtual resources on demand. As future work directions we will be building model considering virtualization and cloud environment. Furthermore, we will be developing prediction models based on monitoring metrics of application and services.

REFERENCES

- [1] A. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", in Proc. of the Grid Computing Environments Workshop, pp. 1-10, December 2008.
- [2] Z. Shi, H. Huang, J. Luo, F. Lin, and H. Zhang, "Agent-based grid computing", Applied Mathematical Modelling 30, pp. 629-640, July 2006.
- [3] R. Wolski and N. Spring, "The Network Weather Service: A distributed resource performance forecasting service for metacomputing", Future Generations of Computer Systems 15, pp. 757-768, October 1999.
- [4] P. A.Dinda and D. R. O'Hallaron, "Host load prediction using linear models", J. Cluster Computing Volume 3, Issue 4, pp. 265-280, 2000.
- [5] H. Koide, N. Yamagishi, H. Takemiya, and H. Kasahara, "Evaluation of the resource information prediction in the resource information server", IPSJ Transactions on Programming, Vol. 42, SIG3(PRO10), pp. 65-73, 2001.
- [6] P. A. Dinda and D. R. O'Hallaron, "An Evaluation of Linear Models for Host Load Prediction", the Eighth IEEE International Symposium on High Performance Distributed Computing, pp. 87-96, August 1999.
- [7] J. Huo, L. Liu, L. Liu, Y. Yang, and L. Li "Selection of the Order of Autoregressive Models for host Load Prediction in Grid", Eighth International Conference on Software Engineering, Parallel/Distributed Computing, IEEE, pp. 516-521, July 2007.
- [8] L. Yang, I. Foster, and J.M. Schopf, "Homeostatic and tendency based CPU load predictions", Proc. 17th Int'l Parallel and Distributed Processing Symp, pp.42-50, April 2003.
- [9] J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive multi-resource prediction in distributed resource sharing environment", In: IEEE International Symposium on Cluster Computing and the Grid, pp. 293-300, April 2004.
- [10] Y. Zhang, W. Sun, and Y. Inoguchi, "CPU Load Predictions on the Computational Grid", IEICE Trans. Inf. & Syst., Vol. E90-D, No.1, pp. 40- 47, January 2007.
- [11] K. Beghdad-Bey, F. Benhammadi, Z. Guessoum, and A. Mokhtari, "CPU Load Prediction Using Neuro-Fuzzy and Bayesian Inferences", Neurocomputing journal 74, pp. 1606-1616, May 2011.
- [12] K. Beghdad-Bey, F. Benhammadi, A. Mokhtari, and Z. Guessoum, "Mixture of ANFIS systems for CPU load prediction in metacomputing environment", Future Generation Computer Systems 26, pp. 1003-1011, July 2010.
- [13] X. Liu, Z. Ni, D. Yuan, Z. Wu, Y. Jiang, J. Chen, and Y. Yang, "A novel statistical time-series pattern based interval forecasting strategy for activity duration in workflow system", Journal of systems and software 84, pp. 354-376, March 2001.
- [14] S. Chiu, "Fuzzy Model Identification Based on Cluster Estimation", Journal of Intelligent & Fuzzy Systems, Vol. 2, No. 3, pp. 267-278, September 1994.
- [15] R. Jang, "ANFIS: Adaptive network-based fuzzy inference system", IEEE Transactions on Systems, Man and Cybernetics, 23 (3), pp. 665-685, June 1993.

- [16] L. Yang, J. M. Schopf, and I. Foster, "Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environments", In Proc. Supercomputing 2003, vol. 11, pp 15–11, April 2003.
- [17] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control", IEEE T. Syst. Man Cyb., SMC-15(1), pp. 116–132, January 1985.
- [18] N. Diamantidis, D. Karlis, and EA Giakoumakis, "Unsupervised stratification of cross-validation for accuracy estimation", Artificial Intelligence 116, pp. 1-16, January 2000.
- [19] F. Borko and E. Armando, "Handbook of Cloud Computing", Springer Book, 2010.
- [20] <http://www.cs.cmu.edu/~pdinda/LoadTraces> [April 2013].
- [21] <http://people.cs.uchicago.edu/~lyang/Load> [April 2013].

On the Analytical Characterization of a Real Life Virtual Network Function: The Italtel Virtual Session Border Control

Sergio Montagna and Pietro Paglierani
 Italtel S.p.A.: Product Unit Smart Networks
 Settimo Milanese, Italy
sergio.montagna@italtel.it; pietro.paglierani@italtel.it

Abstract— In this paper, we analyze a real-life application of virtualization: the Italtel Virtual Session Border Controller (VSBC). The measurements obtained in *ad hoc* loading experiments show that the VSBC performance is not linear with respect to variations in the call rate. Such a behavior is not in accordance with the theoretical results predicted by standard statistical tools based on queuing theory. As a consequence, particular attention must be paid to accurately assess the VSBC performance, because inaccurate estimates could lead to undue costs, or under-performing solutions. To overcome this problem, a novel approach to accurately predict the VSBC performance is proposed, which allows optimizing the system behavior and minimizing its costs.

Keywords—Virtualization; Telephony; SBC; RTP; Erlang.

I. INTRODUCTION

In the last decade, many efforts have been devoted by the telecommunication industry to develop in software some fundamental network functions, which could previously be provided only by specialized hardware equipment. Recently, however, the rapid advances in virtualization technologies and parallel computation have made the software implementation of network functions not only feasible, but also very attractive to network providers, as an effective alternative to proprietary hardware-based applications.

The adoption of Virtual Network Functions (VNF) [1] can significantly reduce the costs of network equipment. VNF, in fact, typically run on commercial servers, produced in high volumes and with large economies of scale to satisfy the huge demand originated by the Information Technology market. The use of a common hardware platform to implement a variety of different applications can also greatly simplify the network infrastructure, and therefore reduce its maintenance costs.

Finally, it is widely acknowledged that the use of VNF will enable scalability, rapid re-configuration and optimal allocation of network resources; hence it will give “elasticity” and “openness” to the network infrastructure, now “ossified” by the deployment of a *plethora* of closed

appliances based on proprietary hardware architectures [1]. In this paper, we present a simple technique to analyze and predict the performance (i.e., measurement of the virtual machine load, defined in the following [2]) of a complex VNF. As a case study, we consider the problem of assessing the performance of a virtual Internet Protocol Telephony function, namely a Session Border Controller (SBC), implemented and commercialized by Italtel. An SBC [3] operates at the edge of two separate networks, both on the control plane and on the media plane. On the control plane, it performs load balancing and call-control; on the media plane, the SBC provides media adaptation capabilities, i.e., it can adjust in real time the coding format of the speech signals transmitted by the users.

In this paper, we discuss the main problems encountered in the experimental characterization of the VSBC. In particular, we have observed that the standard performance analysis based on classical queuing theory [2] can provide inaccurate results. To overcome such a problem, we present a novel analytical framework, which allows predicting and optimizing the overall VSBC performance in an accurate way. The presented analytical solution can be easily extended to any VNF.

The structure of the paper is as follows. In the next section we briefly describe the VSBC. In Section III we report the experimental performance results observed in the lab in a number of *ad hoc* experiments. Finally, we propose the analytical solution and present our conclusions.

II. THE VIRTUAL SBC MODEL

In Fig. 1, we show a simplified scheme of the virtual SBC implemented by Italtel to handle up to 2K Erlang.

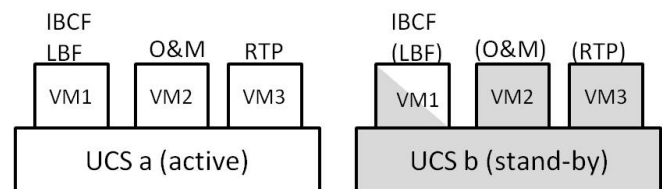


Figure 1. A simplified block-scheme of the virtual SBC architecture. In white, the active VM's, in grey, with the names in brackets, the stand-by VM's.

The used hypervisor is VMWare vSphere Hypervisor 5.1, the VM are based on the Linux operating system. The VSBC runs on two CISCO UCS B200 servers, with hyper-threading enabled, that in the following will be referred to as UCS a and UCS b, respectively. Each server runs three Virtual Machines (VM) implementing three different functions.

a) A first VM, operating on the control plane, implements the Load Balancing Function (LBF) and the Border Control Function (BCF). Such a VM runs on the active server UCS a. Two virtual Central Processing Units (vCPU's) are assigned to this VM, which will be indicated as VM_{LBiBCF}. A second VM operating on the control plane runs on UCS b. This second VM, however, performs only the BCF, while the LBF is in stand-by, thus protecting the LBF running on UCS a in case of fault according to an active/stand-by protection scheme. We will indicate this second VM as VM_{iBCF}.

b) A second VM is dedicated to providing the SBC Operation & Management (O&M) functions. Four vCPU are assigned to such a VM. Also the O&M function is implemented by adopting the active/stand-by redundancy scheme.

c) The third VM is equipped with 4 vCPU. This VM will be referred to as VM_{codec}; it performs Real Transport Protocol (RTP) [4],[5] media packet processing, both in the so-called Network Address Translation (NAT) scenario, and in the transcoding scenario. In the NAT scenario, only the media packet network address is modified, while the RTP header and payload are left unmodified; conversely, in the transcoding scenario the RTP header and payload are processed, so as to change the adopted coding scheme when forwarding the RTP packets from one network to the other. An equivalent stand-by VM is present on the USC b server, to provide redundancy.

A scheme that summarizes the basic call flow is shown in Fig. 2. One can observe that:

- a) The basic call includes seven Session Initialization Protocol (SIP) messages, {Invite,100 trying, 180 Ring, 200 OK, ACK, bye, 200 OK}
- b) In this scenario, the call is processed both by VM_{LBiBCF} (on UCS a) and by VM_{iBCF} (on UCS b).
- c) When the calling and called users adopt different speech codecs, namely G.711 and G.729 [4],[5], the scenario is labeled as Transcoding otherwise as NAT.

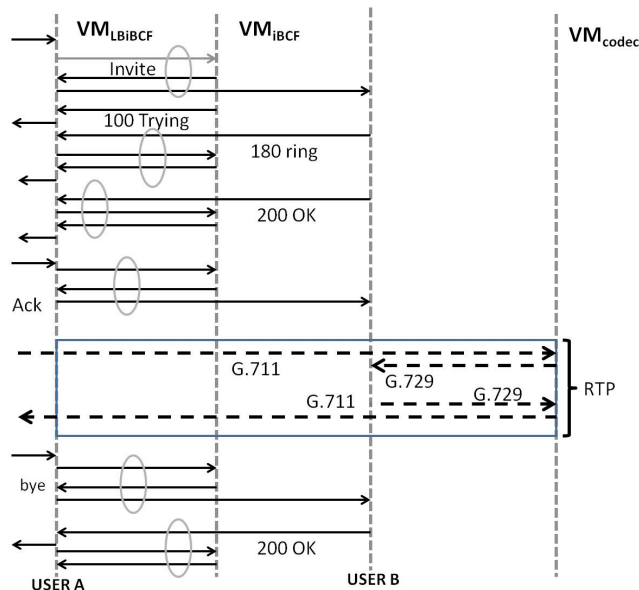


Figure 2. Control plane: SIP/RTP message flow

III. RESULTS

Fig. 3 shows the overall CPU load of VM_{codec}, ($\rho_{VM_{codec}}$) when a NAT call is considered, as a function of the “call per second” (cps) parameter. We can deduce the following conclusions:

- a) Effects due to the RTP packet processing on VM_{codec}: at equal call rates, the load observed on VM_{codec} can result different, due to the difference of offered Erlang values (**520 Erlang; 260 Erlang**).
- b) From the observed results, we can obtain the experimental values for $\{T_1; h_1^{Erl}(ptime)\}$, which represent the cost of the single call, and the contribution to the load due to the traffic expressed in Erlang in the NAT scenario, respectively. Such values can be used to estimating the load of VM_{codec} through the expression [2]:

$$\rho_{teor}^{VM} = \rho_{base} + \sum_{i=1}^2 [\lambda_i * T_i + \lambda_i * hold_i * h_i^{Erl}(ptime)] \quad (1)$$

where:

- ρ_{teor}^{VM} : is the offered load of VM_{codec}
- ρ_{base} : is the load of VM_{codec} without traffic
- λ_1 : is the offered call rate to handle NAT
- λ_2 : is the offered call rate to handle transcoding
- T_1 : is the cost of the call to handle NAT
- T_2 : is the cost of the call to handle transcoding
- hold₁**: is the length (in seconds) of the RTP phase of the call to handle NAT
- hold₂**: is the length (in sec.) of the RTP phase of the call to handle transcoding

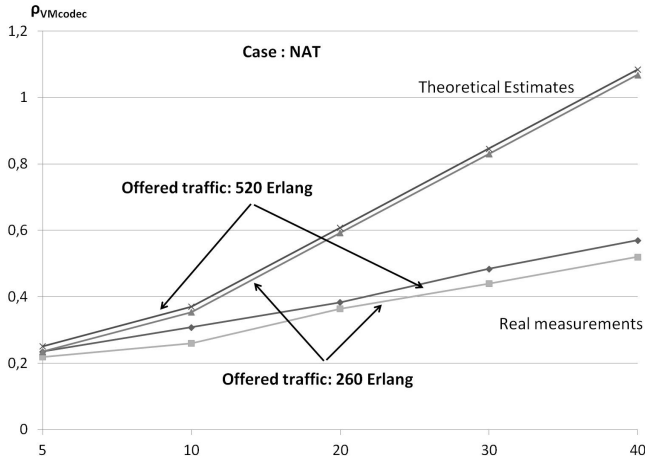


Figure 3. VM_{codec} load as a function of the offered traffic (call per second)

$h_1^{Erl}(ptime)$: is the cost to manage one NAT Erlang. This value depends on the packetization time ($ptime$) of the codec.
 $h_2^{Erl}(ptime)$: is the cost to manage one transcoding Erlang. This value depends on the packetization time ($ptime$) of the codec.

The results shown in Fig. 3 highlight that the theoretical load predicted by (1) can be significantly underestimated. For a given load value, for instance $\rho_{VMcodec}$ equal to 0.6, the theoretical estimate predicts an actual call rate of 20 call per second (caps), while the actual call rate that can be successfully processed is 40 caps.

IV. ANALYTICAL SOLUTION

The last observation suggests that a different approach must be adopted to predict the virtual SBC load in an accurate way. In Fig. 4, we show a simplified scheme that summarizes the new approach to estimate the load proposed in this paper. The new approach is based on the use of a reduction factor f applied to the theoretical estimate ρ_{teor}^{VM} of the load provided by (1). This quantity is lower bounded by the VM load without traffic, i.e., ρ_{base} ; the upper bound can be theoretically infinite. In our application, we assume that the upper bound is determined by the number of vCPU (N_{vcpu}) dedicated to the considered VM. Thus, the range for the load can be defined as $\rho_{base} \leq \rho_{teor}^{VM} \leq N_{vcpu}$.

We assume that the reduction factor f depends on the offered load (ρ_{teor}^{VM}) and on the value N_{vcpu} . Furthermore, we also assume that the reduction function f exhibits a linear behavior in the range $1 \leq f \leq N_{vcpu}$, with $N_{vcpu} \geq 2$. As shown in Fig. 4, the method requires that a load measurement is performed at low traffic, so that the observed load (ρ_{meas}) can be considered close to the base load (ρ_{base}).

This measurement is performed in order to estimate the processing cost of the single call on the considered VM. The estimated VM processing cost thus results equal to:

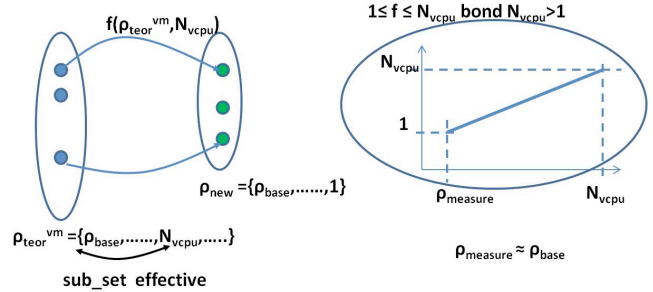


Figure 4 Simplified scheme of the proposed approach to estimate the VM load

$$\rho_{new} = \frac{\rho_{teor}^{VM}}{f} = \frac{\rho_{teor}^{VM} * (N_{vcpu} - \rho_{meas})}{(N_{vcpu} - 1) * (\rho_{teor}^{VM} - \rho_{meas}) + (N_{vcpu} - \rho_{meas})} \quad (2)$$

The application of (2) to the results previously discussed is shown in Fig 5.

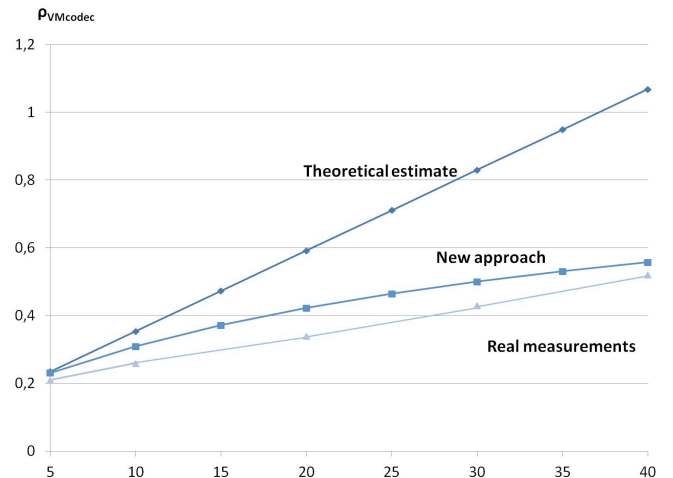


Figure 5. Load of VM_{codec} vs offered call rate (call per second)

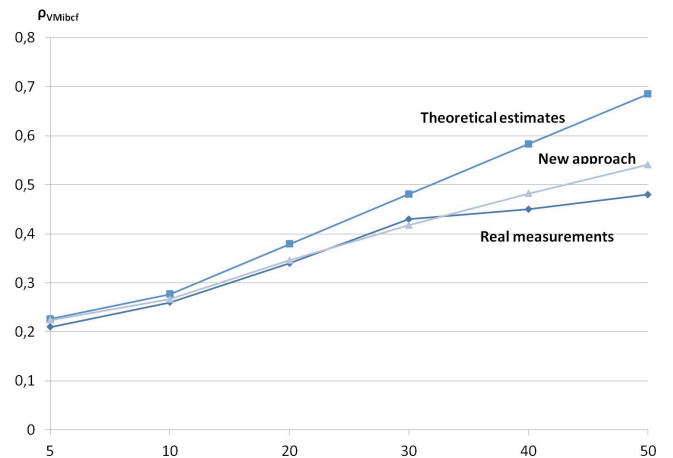


Figure 6. Load of VM_{codec} vs offered call rate

The performance estimates achieved with (2) result significantly more accurate than those provided by (1),

which are linear with respect to the offered traffic.

In Fig. 6 we show the load of VM_{iber} as a function of the call rate. In these tests, the quantity “offered Erlang” has been kept constant, equal to **260** Erlang; only the NAT scenario is considered.

The use of (1) provides over-estimated performance values; conversely, the performance predicted by the new approach results more accurate. The obtained measurements also indicate that the gradient of the load curve decreases for increasing traffic. This means that VNF's tend to optimize the cost of processing telephony traffic. Thus, when traffic increases, vCPU's dedicate more time to traffic handling.

V. CONCLUSIONS

The Italtel Virtual Session Border Controller (VSBC) has been developed to handle up to 2K Erlang of Voice over IP sessions (and in the near future, up to 4K Erlang). It has been implemented by adopting the concept of virtualization. The measurements carried out in our laboratory have shown that the VSBC performance (processing load of the virtual machines) is non-linear with respect to variations in the rate of processed calls. It has been observed that the virtual system tends to optimize the processing cost of the calls; as a consequence, the overall performance results better than the one predicted by linear models. We have also proposed a strategy aimed at matching experimental data with analytical predictions. The main result originated by this effort is a technique which allows to accurately dimensioning the deployed solutions, reducing their cost.

The proposed analytical method allows to reliably predicting the number of virtual CPU's that must be assigned to the VSBC, to achieve the target performance. This way, the cost per Erlang of the VSBC can be minimized, thus increasing the competitiveness of the system.

ACKNOWLEDGMENT

The authors wish to thank their colleagues Roberto Porfidio, who carried out the experiments and measurements discussed in this work, and Marco Beccari, for his support on the VSBC.

REFERENCES

- [1] M. Chiosi, *et al.*, "Network Functions Virtualization," presented at the "SDN and OpenFlow World Congress", October 22-24, 2012, Darmstadt, Germany.
- [2] Leonard Kleinrock, "Queuing Systems: Volume I, Theory", John Wiley and Sons, New York, 1975.
- [3] IETF RFC 5853 "Requirements from Session Initiation Protocol (SIP) – Session Border Control (SBC) deployment," April 2010.
- [4] Paglierani, P.; Petri, D.; "Uncertainty Evaluation of Objective Speech Quality Measurement in VoIP Systems," *Instrumentation and Measurement, IEEE Transactions on*, vol.58, no.1, pp.46-51, Jan. 2009
- [5] IETF RFC 3551, "RTP Profile for Audio and Video Conferences with Minimal Control"

Trusted Computing on Heterogeneous Embedded Systems-on-Chip with Virtualization and Memory Protection

Marcello Coppola

ST Microelectronics
Grenoble, France

marcello.coppola@st.com

Miltos D. Grammatikakis and George Kornaros

Technological Educational Institute of Crete
Heraklion, Greece

{kornaros, mdgramma}@cs.teicrete.gr

Alexander Spyridakis

Virtual Open Systems
Grenoble, France

a.spyridakis@virtualopensystems.com

Abstract—The paper examines the architecture of a secure and trustworthy cloud platform, which ensures strong logical and physical security on the client devices using a two-layer security mechanism: a) a hardware security module located on the SoC of the client device that protects incoming and outgoing communications (e.g., to/from an external memory) against physical attacks, and b) system software and hypervisor extensions that isolate virtual machines from one another and from the underlying hardware in order to protect against logical attacks.

Keywords—cloud computing; confidentiality; integrity; multicore SoC; protection; security; virtualization.

I. INTRODUCTION

A new era is emerging in consumer, industry, and government areas, where traditional consumer and mobile devices are replaced by intelligent, next generation systems, such as smart phones, smart TVs and smart tablets that provide innovative services, such as social networking and on-demand multimedia (e.g., Netflix* [8]), by connecting to the cloud. Meanwhile, content providers increase the availability of large-scale, high-quality libraries of web data with text, images, sounds, videos and animations. The technology races towards new generation, powerful, complex, smart devices promotes convergence of traditional video and Internet-based content deployed in cloud computing infrastructures and increases the possibility of security breaches.

For example, devices, such as Intellectual Property (IP) set-top boxes, residential gateways or media players, now provide a multitude of services, such as graphical user interfaces, digital rights management, secure transcoding protection, network provisioning and payment. Each service finds its physical representation in a mixture of hardware and software components, ranging from small security-critical software stacks running on basic processors or accelerators, up to commodity operating system (OS) on complex application processors. Since each of these highly heterogeneous software stacks uses sensitive data that must be protected, individual services must collaborate to enable global system security [5] [11]. This leads to a significant increase in complexity and associated development costs.

Security solutions for end-users (individuals, companies) connecting to the cloud using client equipment are of utmost concern in the era of cloud services and applications [1]. Cyber-secure architectural solutions for cloud environments must offer ways to fully secure system and end user

applications and services against cyber-criminal end-users, even for the components that will run on the client side. Today the lack of appropriate isolation of source code and data among trusted and untrusted applications is one the main challenges in building a secure architectural solution. On the other hand, offering trustworthy cloud computing services that would prevent from rogue administrators spying or altering end user data and computations requires significant hardware and software modifications in data center architecture. This implies that on the end user side, there is no trust to the cloud provider, especially if the end user stores confidential info. Therefore, a viable and economical solution is to enhance the security level of the connected smart device when accessing the cloud. This new idea could speed up utilization of cloud infrastructure by connected devices and allow service providers to trust sensitive computations performed by end users and consequently delegate processing tasks to them.

This paper describes work in progress that aims to provide a viable solution towards protecting the integrity and confidentiality of sensitive data (e.g., movie, photo, e-book) and software applications in a modern cloud infrastructure where approved devices are connected to the cloud. This work targets protection from two kinds of adversaries: (i) rogue applications such as virus, Trojans possibly launched by the user himself, (ii) physical adversaries such as probing, spying at or tampering with the communication link connecting the device to the external cloud environment.

Section II considers the current state-of-the-art in System-on-Chip (SoC) virtualization including existing memory protection strategies. This section lays out the path towards presenting the TRESCCA security approach in Section III. Finally, Section IV concludes the paper.

II. VIRTUALIZATION AND SECURITY

On top of a hardware platform, we have the software stack, including the OS, the middleware and the application layer. Security of the device that runs applications from different sources is usually under the responsibility of the OS. The OS uses software (e.g., virtual memory, file permission, memory protection) and ad hoc hardware mechanisms to isolate different applications sharing common physical and logical resources, such as software libraries, services and resources, e.g., printers, graphics accelerators. The complexity of modern OSs (large number of code lines, developed by different groups) creates different security vulnerabilities resulting from software misbehaviors. These are exploited by a cyber-criminal, who attempts to subvert

the security mechanisms supported by the OS and get control of the device and data. For instance, overwriting data or function pointers, dynamic memory allocation (double-freeing, referencing or writing to free memory, zero-length allocations, and buffer overflows are well known techniques used to bypass any security protection imposed by the OS.

Vendors are now using virtualization technology to isolate physical resources from applications and platforms that use them. This is performed by introducing the virtual machine (VM) concept that serves as a guest software environment that supports a stack consisting of an operating system (OS) and application software. Each VM is independent of other VMs and uses the same interface to processors, memory, accelerators, and I/O provided by a physical platform. VM isolation provides the means to regulate application access to computational resources, thus enabling malware detection capabilities. Isolation is achieved by inserting a hypervisor layer between the operating system and the hardware. This enables the hypervisor layer to govern all interactions that take place between the OS (and the layers above it) and hardware [4] [7].

In full virtualization, the hypervisor provides the same hardware interfaces as those in the physical platform, hence the guest OSs and applications do not need to be modified. Since full virtualization increases information sharing among different system layers, security maintenance becomes very complex. Thus, NIST has proposed security management recommendations that involve the host OS (if applicable), the hypervisor and the guest OS [9]. NIST best practices (policies and checks) for a secure hypervisor layer involve installing updates, monitoring, restricting access via authentication, encryption and integrity mechanisms, disconnecting/disabling unused hw/sw components and performing clock synchronization [10]. The specified practices affect hypervisor configuration, initiation, design and planning, implementation, operation, maintenance and disposition and ensure that data access and transmission threats are thwarted.

A. *Embedded Virtualization and Security*

Mobile platforms and set-top boxes are in the middle of a global transition period in which client devices manage to support high-level operating systems and middleware, quickly moving from a close or walled garden limited environment to a setting where a walled garden has to coexist with an open one. In this new scenario, devices are able to run any third-party application that may or may not be certified by the operator. In this context, it is crucial to ensure that third party applications cannot break security. Otherwise, if isolation is broken, sensitive content could be easily stolen or edge devices could be used as a Trojan horse to break cloud security. Hypervisors would allow vendors to isolate important trusted services (e.g., billing, authentication, phone service) from the open operating system layer and run them in isolated, tamper-proof virtual machines (VMs). Thus, trusted services are not affected even if the open environment is compromised.

Traditional virtualization technology resolves isolation of different applications at the processor level, but suffers from

non negligible drawbacks [7]. Indeed, it allows sharing of processing and shared memory resources efficient and secure on homogeneous SMP architectures that can be controlled on a common trusted basis. However, it is not secure for heterogeneous shared-memory multiprocessor systems-on-chip (MPSoCs). In fact, most connected smart devices architectures are heterogeneous, including different islands of computation such as GPU, DSP and hardware accelerators. Islands of computations cannot natively support virtualization, since they lack memory management units, and often do not offer inherent ways of establishing privilege levels. Therefore, applications running in such systems are able to access the whole address space, breaking the required isolation assumption imposed by virtualization. In order to address these issues, security hardware extensions to processor and interconnects are being considered.

A few years ago, bi-partitioning techniques introduced in ARM's TrustZone [3] extended the ARMv6 architecture by adding the concepts of "secure" and "non-secure" states and a "secure monitor mode" used for switching between the two. In addition, the AMBA3 AXI has been extended with two new signals (ARPROT/AWPROT) that indicate whether the respective read/write transaction is secure or non-secure. Nowadays, binary bi-partitioning cannot meet the security requirements of cloud-connected devices. Moreover, TrustZone technology cannot protect against bus probing which can be used to attack the software stacks.

MPSoC security must be addressed by a platform-wide protection mechanism covering the full communication infrastructure, instead of a processor-centric mechanism [12]; similar approaches have also been proposed in [5] [11]. The proposed concept defines a protection domain as a set of specific access rights to a shared address space and maps each software stack to a specific domain. Notice that software stacks may have right overlaps between them.

In order to make the security check the proposed approach may suffer from long latency, especially if there is a miss in the local permission look-aside buffer, and the missed entry has to be loaded from external memory. Thus, due to the granularity of the security checks, silicon cost is unacceptable for embedded devices.

The basic concept in our approach is to implement a low-cost solution at the Network-on-Chip (NoC network interface. With ideal distributed co-hosting of several protection domains, software stacks transparently and efficiently share resources (processors, memory and peripherals) issuing memory accesses through Direct Memory Access (DMA) controllers.

III. THE TRESSCA APPROACH

The TRESSCA architecture secures critical data in a fully end-user transparent way, without storing information in centralized pools that define an attractive attack point [13]. TRESSCA consists of a CPU cluster, on-demand media accelerators and storage interconnected in a heterogeneous shared memory MPSoC via a complex NoC (STM's Spidergon STNoC). Each CPU cluster is a symmetric multiprocessor (SMP) hosting OS execution.

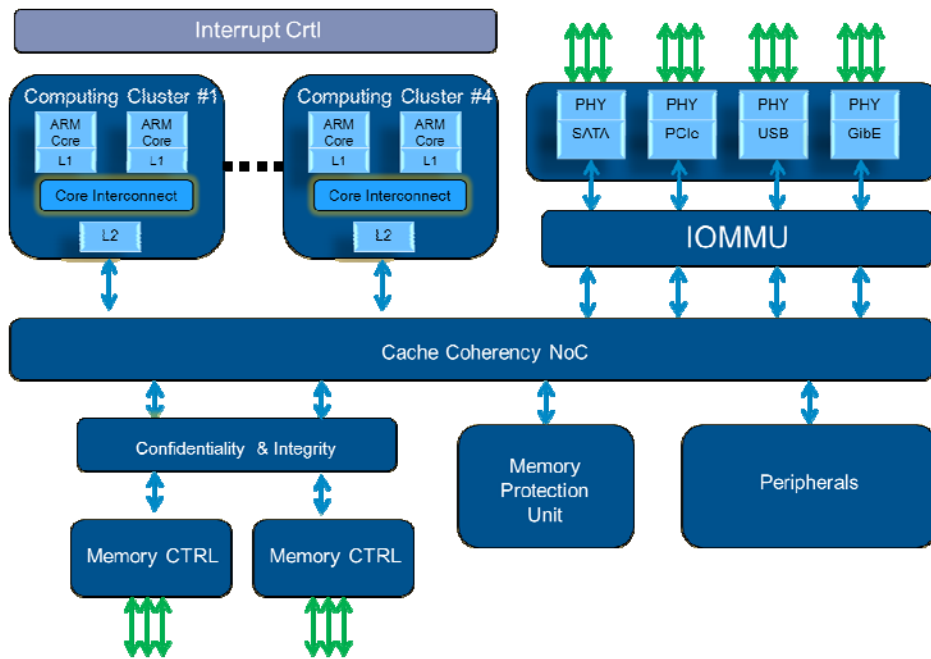


Figure 1. TRESCCA architecture with Hardware Security Module (HSM)

A. Secure Information Processing

TRESCCA introduces a novel security infrastructure that aims to protect the confidentiality and integrity of sensitive software against two types of adversaries:

- Logical adversaries like rogue applications: viruses, Trojans or malware launched by the end-user.
- Physical adversaries like the end user himself, with complete physical access to the system. For instance, the end user can issue a board-level attack by probing the bus between the SoC and its external memory or tampering with a system communication link.

Notice that these two kinds of attacks can also be combined, as has already been done recently against famous game consoles and other consumer equipment.

Protecting the system against logical adversaries will rely on virtualization techniques, while board-level physical attacks will be prevented by input and output data encryption and integrity checking. Both memory protection and virtualization techniques, implemented using hardware and tightly-coupled system drivers, will jointly reinforce a secure hypervisor kernel that isolates critical applications and

prevents memory tampering. The following subsections describe how TRESCCA enhances the NoC backbone by extending its network interface and how these extensions help the hypervisor build the required security infrastructure.

B. NoC Firewall

The NoC communication infrastructure enforces strong isolation of VM by tagging the underlying transactions. What this means is that a potentially compromised Guest OS in a Virtual Machine cannot access data that is tagged by another VM. Next, we use the term *domain* to refer to an isolated environment in the platform, to which a subset of the shared physical memory is allocated.

Using the virtualization concept, we can create a level of indirection between physical and virtual components. Each physical component is associated to many different virtual instances that are allocated to a domain and are referred to as the domain's assigned components. For modern CPUs, this is possible using hardware virtualization extensions [2], for other components, such as DMA or hardware accelerators, an IOMMU is used [6].

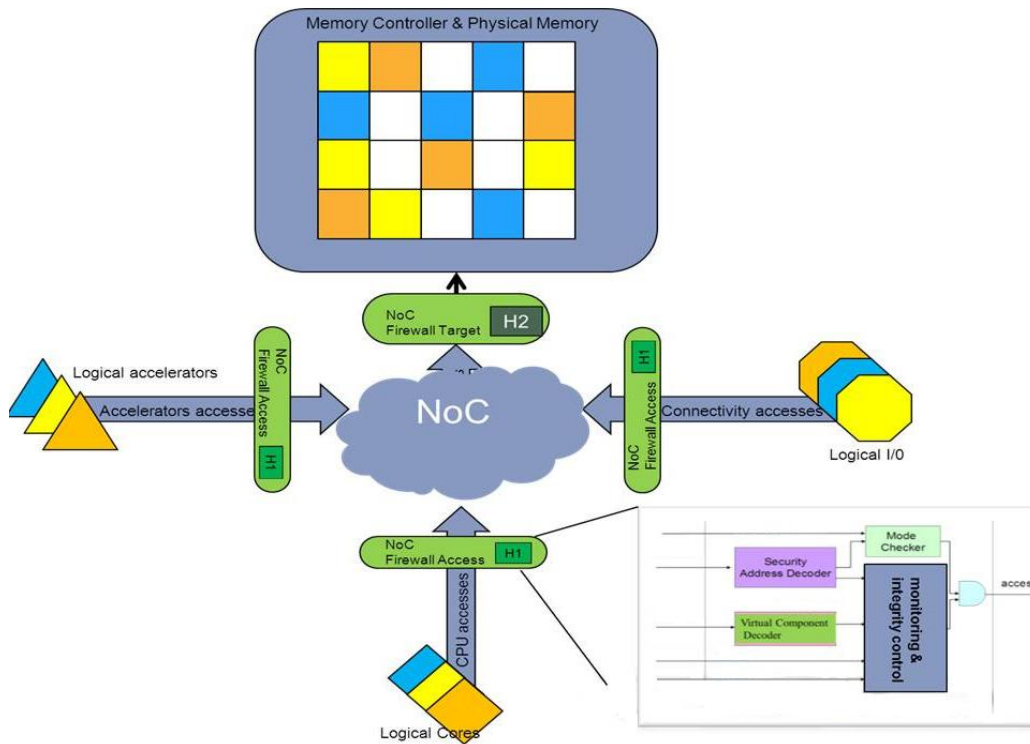


Figure 2. Partitioning of physical memory to different logical domains

Multiple domains can co-exist in a platform and a virtual component (e.g., a virtual CPU) can be mapped to a single domain. True domain isolation is achieved by blocking accesses (read/write NoC transactions) from resources outside the domain to their assigned physical memory; ideally, network routing paths are also balanced across VMs by assigning separate virtual circuits. This implies establishing a set of access rights on different address regions and ensuring that these rules are observed at each network interface. Our solution will be processor-independent, although interrelation to predefined processor privileged levels is desirable (e.g., ARM v7 PL0, PL1, etc).

Each initiator transaction is tagged with a corresponding VM and/or process identifier. The main innovation point for defining the set of access rights for each tuple (VM id, process id, and physical address) is introducing two levels of memory hierarchy. These consist of hierarchy 1 cache at all initiator interface and a hierarchy 2 cache at the target interface (resp. H1, H2 in Figure 2). In case of H2 miss, the NoC Firewall target interface is responsible for fetching the required entry from the physical memory containing the permission tables shared by the different NoC Firewall access points. A scalable NoC Firewall will enable flexible and efficient assignment of virtual components to an arbitrary number of domains, proving low latency and power-efficiency compared to past research, such as [12]. Moreover, by policing the NoC Firewall access point at the initiators, we would be able to detect and subvert Denial-of-

Service attacks, where malicious code attempts to saturate the NoC through massive unauthorized accesses.

At the physical level, NoC Firewall and associated cryptography will ensure that all transactions between the SoC and its external environment are protected through domain isolation, confidentiality and integrity [5]. Thus, it will be infeasible for an adversary to spy or alter sensitive data crossing the SoC boundary without issuing an interrupt.

C. STNoC Synthesis

We have synthesized STNoC using STM 32nm technology in order to estimate the area overhead of the NoC Firewall. Assuming 20 domains and a NoC with 80 initiator and 68 targets, a secure AXI read-only interface occupies 23 to 30K gate equivalent (GE), compared to 20 to 28K GE for the non secure case. Similarly an AXI write-only interface occupies 21 to 51K GE, compared to 19 to 49K GE for the non-secure case. Hence, the area overhead is 5 to 14% for read- and 3 to 11% for write-only interface, depending on the precise AXI configuration .

D. Extended Hypervisor Security

At the software layer, the TRESCCA hypervisor (KVM) must provide strict isolation by running different VMs on the connected devices. Thus, in our methodology, a *trusted VM* associated to a trusted domain, where data and code encryption is enforced, is assigned as the security master of the SoC resources, excluding any IO components. This VM

is responsible for creating and managing domains, for allocating physical memory to all domains and for setting up the necessary virtual and physical address mechanisms. For example, this VM can master the rendering functionality of a display, to avoid any malware execution that captures authentication data. Security is further enhanced by a set of approved applications that software encrypt (possibly via an on-chip hardware accelerator) communication with the external memory, provide integrity checking and dispose any unused network connection. This way each application is completely isolated and external attacks are not possible.

In addition, the hypervisor defines a *secure VM* managing all closed or corporate “walled garden” applications (cf., set-top box example). The secure VM is associated to a secure domain that may include I/O accelerators and provides services to connect to the external world, e.g., to an untrusted VM. The main difference from an application running on the trusted VM is that these applications can communicate through a firewall to the cloud for additional computing power and/or storage.

The remaining VMs can execute untrusted applications and connect to the external cloud environment. In these VMs there is always a risk that a downloaded application exploits security vulnerabilities. Therefore, mandatory monitoring and integrity control (MIC) protocols at the underlying NoC Firewall (see Figure 2) ensure that security policies are uniformly enforced at the hypervisor layer [1]. Our custom MIC hardware extensions are related to software security, similar to mandatory access control (MAC) extensions in SELinux, e.g., the Loki tagged memory architecture [14]. Restricting different workloads through our MIC ensures that viruses and other malicious code cannot spread from one VM or guest OS to another, and data cannot easily leak from an untrusted VM or guest OS to another one even if VMs start to misbehave.

IV. CONCLUSIONS AND FUTURE WORK

Cloud computing is an emerging technology that quickly goes mainstream, making our society increasingly online, with consumers using browsers embedded in mobile devices or modern TV sets to access e-mail and social media. Besides smart phones and TVs and tablets grabbing the headlines, in the near future game consoles, cameras, photo frames, radios, printers and set-top boxes will also be connected to the cloud. Depending on the nature of the threat, cloud security must encompass three components: confidentiality, integrity, and availability. Confidentiality is violated whenever sensitive information is disclosed to any unauthorized entity (human, program, or system). Integrity is violated whenever unauthorized code is executed or unauthorized data is used. Availability is violated when an attacker succeeds in denying services to legitimate users.

The ongoing TRESSCA project develops a lightweight, non-intrusive secure hardware and system software-based infrastructure, that supports multiple domains on top of virtualization technology, in order to realize separation among client’s broadband services (e.g., in Android) and

global broadcast services (e.g., in NDS, HbbTV). This client-centric, “walled garden” allows client control over its application code and media content. Moreover, virtualization technology will allow set-top box or smart TV to efficiently execute (and migrate, if necessary) multiple virtual machines enabling hardware consolidation, increased utilization and energy savings. Thus, different middleware and OSs can run simultaneously on a single device, laying the foundations for reducing cost, while promoting interoperability of secure and trustable interactive services and cross-platform application scenarios in heterogeneous virtualized multicore systems. Most project outcomes will be publicly released as open source software. Functional specifications of the architecture currently developed aim to characterize performance and silicon overhead with typical execution scenarios that run on top of an extended open source, secure KVM hypervisor.

ACKNOWLEDGMENT

This work is partially supported by the EC through FP7 collaborative project TRESSCA (GA No. 318036).

REFERENCES

- [1] A.M. Azab, "New system security mechanisms for cloud computing," PhD Thesis, NCSU, Dept. CS, 2011.
- [2] ARM, "Virtualization is coming to a platform near you," see <http://www.arm.com/files/pdf/System-MMU-Whitepaper-v8.0.pdf> [retrieved:4/2013]
- [3] ARM, "TrustZone: security foundation," available from <http://www.arm.com> [retrieved: 4/2013]
- [4] G.W. Chow and A. Jones, "A framework for anomaly detection in okl4- linux based smartphones," Proc. Australian Information Security Management Conf., 2(2), pp. 5-10, December 2008.
- [5] L. Fiorin, G. Palermo and C. Silvano, "A security monitoring service for NoCs," Proc. Conf. Hardware/Software Codesign and System Synthesis, New York, NY, USA: ACM, pp. 197–202, October 2008.
- [6] G. Kornaros, M.D. Grammatikakis, and M. Coppola, "Towards full virtualization of heterogeneous NoC-based multicore embedded architectures," Proc. Conf. on Embedded and Ubiquitous Computing, pp. 345-352, December 2012.
- [7] S. Nanda and T. Chiueh, "A survey of virtualization technologies," Technical Report, SUNY - Stony Brook, 2005.
- [8] Netflix, <http://en.wikipedia.org/wiki/netflix> [retrieved: 4/2013]
- [9] NIST, available from <http://www.nist.org> [retrieved: 4/2013]
- [10] K. Scarfone, M. Souppaya, and P. Hoffman, "Guide to security for full virtualization," NIST standard SP 800-125, January 2011.
- [11] G. Palermo, L. Fiorin, S. Lukovic, V. Catalano, and C. Silvano, "Secure memory accesses on networks-on-chip," IEEE Trans. Comput., vol. 57, no. 9, pp. 1216-1229, September 2008.
- [12] J. Porquet, A. Greiner, and C. Schwarz, "NoC-MPU: a secure architecture for flexible co-hosting on shared memory MPSoCs," Proc. Design Automation and Test in Europe, pp. 591-594, March 2011.
- [13] TRESSCA project, <http://www.tressca.eu> [retrieved: 4/2013]
- [14] N. Zeldovich, H. Kannan, M. Dalton, and C. Kozyrakis, "Hardware enforcement of application security policies using tagged memory," Proc. Symp. Operating Systems Design and Implementation, pp. 225-240, December 2008.

Using Cloud-based Resources to Improve Availability and Reliability in a Scientific Workflow Execution Framework

Sergio Hernández, Javier Fabra, Pedro Álvarez, Joaquín Ezpeleta
 Aragón Institute of Engineering Research (I3A)
 Department of Computer Science and Systems Engineering
 University of Zaragoza, Spain
 Email: {shernandez, jfabra, alvaper, ezpeleta}@unizar.es

Abstract—Different mechanisms, such as checkpointing, task replication, alternative tasks execution or task migration among different resources, for instance, have been traditionally applied in (heterogeneous) grid environments for fault-tolerance. Cloud based resources can easily improve both availability and reliability of a given system when used for recovering faulty tasks. In this paper we present how cloud resources have been included in a framework for the execution of scientific workflows and how this has helped in improving the framework in two different aspects: making it more scalable and more reliable, facilitating the application of very effective fault recovery policies.

Keywords—Fault tolerance; Scalability; Cloud computing; Heterogeneous computing infrastructures; Resource management frameworks.

I. INTRODUCTION

Grid systems are prone to faults [1][2][3]. Different fault-tolerance mechanisms (checkpointing, task replication, alternative tasks, or task migration, for instance) have been traditionally integrated into Grid middlewares and management systems in order to handle and minimize the impact of these faults [4][5]. Nevertheless, these mechanisms do not prevent end-users jobs from experiencing high failure rates when they are executed in this type of distributed computing infrastructures [6]. For that reason, users must play a vital role in the course of detecting these faults: checking execution logs and job outputs, for instance [1]. Undesirable behaviour is then notified to Grid administrators so that they can adopt the necessary steps to restore the Grid.

In the last years, the Grid computing community has concentrated its research efforts on integrating several heterogeneous Grids in order to generate more powerful computing infrastructures. Resource management frameworks have been developed to provide a transparent and easy-to-use access to the set of integrated computing infrastructures. Consequently, these heterogeneous infrastructures are viewed as a whole from the end-users' point of view. Some relevant examples of these solutions are GJMF [7], P-GRADE [8], SWAMP [9], GridWay [10], eNANOS [11], EMPEROR [12], or GMBS [13]. Obviously, this new model of solution requires new fault-tolerance mechanisms at the global level because frameworks

consist of internal services (schedulers, state monitors, resource registries, etc.) that are also prone to faults. These mechanisms must be compatible with the ones integrated into each local Grid. Currently, resource management frameworks use the monitoring and notification capabilities of their middlewares to detect faults. Then, resubmission techniques are integrated into their fault management components to recover the execution of failed jobs.

In [14], authors proposed an open framework for the flexible deployment of scientific workflows in heterogeneous Grid environments. From an architectural point of view, the framework was organized as a set of components connected through a central bus, which was used by the components as the mean to send and receive messages. At the beginning, the fault management was very simple and consisted of re-submitting the faulty task (either to the same computing resource or to an alternative one). In this paper, we try to improve framework availability and reliability by using cloud-based resources. The experience gained by solving complex computational problems has also allowed us to understand a wide variety of faults suffered by this type of distributed computing infrastructures. The use of cloud resources can help solve some of these faults or at least reduce their effects.

The paper is organized as follows. Section II briefly describes the architecture of the proposed framework for scientific workflows execution. The description is mainly focused on the components involved in fault management. Section III introduces the suggested fault classification and a discussion about their corresponding effects. Sections IV and V present two cloud-based solutions for solving availability and reliability problems. We have concentrated on situations caused by a large performance degradation of computing resources and bottlenecks in the common bus. Section VI describes the main related work. Finally, Section VII summarizes the main contributions of the paper.

II. BACKGROUND

As it was mentioned earlier, we proposed a framework for the flexible deployment and execution of scientific workflows. The flexibility has been achieved at different levels:

from a computing point of view, the framework is able to integrate heterogeneous computing infrastructures to create more powerful execution environments; from a programming point of view, workflows can be programmed independently of the computing infrastructures where related jobs will be executed and using different high-level languages widely accepted by the scientific community; and, finally, from a configuration point of view, new functionalities can be dynamically added/removed to the framework in order to meet the different needs of each application and user.

An integration model based on a *message bus* is key to achieve the flexibility of the proposed solution. More specifically, the cornerstone of the proposal is a bus inspired by the Linda coordination model [15]. This component provides an application interface (API) for sending and receiving messages in an asynchronous way, coding them as Linda tuples. The rest of system components offer their capabilities through the common bus, and collaborate by exchanging messages using the bus as the communication channel. This integration model has several advantages compared to other more traditional approaches: (1) a bus reduces the coupling between system components (they are connected by making use of an asynchronous message passing mechanism); (2) components can be dynamically added or removed without disturbing the execution of other existing ones (to adopt new functionalities, for example); (3) a bus favours the scalability and distribution of the solution; and, finally, (4) a bus supports complex message exchange patterns (publish and subscribe mechanism, content-based message routing, etc.) that facilitate more flexible integration strategies.

In this communication model, framework components are not aware of other components connected to the message bus. Each message is assigned an exclusive tag to identify the receiver and each component identifies the messages addressed to it with that tag. Thus, management components and mediators can be easily replicated to improve framework performance and reliability. Replicated components compete for the same messages and the message bus decides which mediator gets each message. As a consequence, components can be easily replaced to adopt new functionalities, change them or fix bugs.

Figure 1 shows the high-level system architecture which is composed of three different layers. At the top, the *User interface layer* is composed of the different programming tools that can be used to program scientific workflows (Taverna, Triana, Kepler, Pegasus, etc.). Resulting workflows are submitted to the framework for their execution. The components of the *Execution layer* are responsible to manage the workflow execution life-cycle. Internally, this layer is composed of the *message bus* and the components that provide the core functionalities. In order to provide this functionality, two types of components have been connected through the bus: *management components* and *mediators*. The first ones offer extra functionalities to enhance workflows, task life-cycle and framework capabilities (meta-scheduling, fault-tolerance, monitoring, etc.). On the other hand, mediators encapsulate

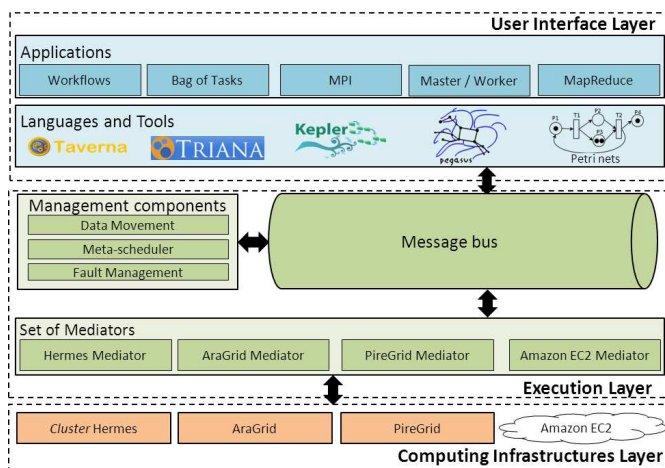


Figure 1: Architecture of the proposed framework for the execution of scientific workflows in multiple heterogeneous computing infrastructures.

the heterogeneity of each specific computing infrastructure to facilitate its integration into the framework (in more detail, a mediator must interact with a specific infrastructure to submit jobs, move input/output data, monitor job executions, detect undesirable states, etc.). Finally, at the bottom of the architecture, the *Computing infrastructures layer* is formed by different and heterogeneous computing infrastructures. At the beginning, three computing environments were integrated: the HERMES cluster hosted by the Aragón Institute of Engineering Research (I3A) [16], which is managed by the HTCondor middleware [17]; and two research and production grids managed by the gLite middleware [18] and hosted by the Institute for Biocomputation and Physics of Complex Systems (BIFI) [19], namely AraGrid [20] and PireGrid [21]. A more detailed description of the architecture can be found in [14][22].

In the first implementation of the framework, availability and reliability issues were deliberately ignored. In this paper, we propose the use of Cloud computing to add these requirements. In this Cloud-based approach the selected integration model plays a relevant role as it will be shown in the following sections.

III. INTEGRATION OF FAULT HANDLING MECHANISMS INTO THE FRAMEWORK

As we have already discussed, grids and computing clusters are prone to faults. In this section, we present various types of faults that can locally occur in these infrastructures and the techniques usually used to detect and handle them. Our discussion focuses on the user perspective and considers the effects produced by these faults in terms of availability (the ability of the system to be ready for successful job submission) and reliability (the ability of the system to successfully execute jobs even in the presence of failures during job execution). Other fault classifications can be found in [1][2][3].

Additionally, the proposed execution framework could also be affected by faults. The message bus is the most critical component of the architecture: if the bus fails, the whole system fails. Besides, the bus can become a bottleneck and, as a consequence, degrade performance (for instance, when a large number of application jobs are being executed by the framework). For this reason, mechanisms that improve the reliability and scalability of the framework must also be integrated.

Let us briefly describe the faults that can affect computing resources and introduce solution mechanisms. A more detailed description will be presented in the two following sections.

A. Faults at the computing infrastructures level

In this work, we have considered the following set of faults, identified from our experience in scientific workflows solving:

- *Computing resource failures*: A computing resource may suffer hardware, network or operating system faults that affect the jobs that are being executed on it. These faults are not critical because they only involve individual resources and can be easily repaired.
- *Hardware upgrades and maintenance*: These actions typically require shutting down the computing infrastructure causing unavailability periods. They involve the failure or cancellation of all jobs submitted to the infrastructure.
- *Software upgrades and maintenance*: Depending on the nature of the software upgrade, it may be transparent, it may cause some resources to be unavailable and some job failures, or it may cause total unavailability and the failure of all jobs. Also, it may affect only certain users. Additionally, these actions often lead to periods when the infrastructure is unreliable due to misconfiguration.
- *Environmental failures*: These faults are provoked by causes external to the computing infrastructure (power outages or cooling issues, for instance). The affected computing infrastructure can become totally unavailable and all running jobs may fail.
- *Deployment and configuration of new software*: The execution of some applications may need to install and configure new software and services. These operations must be performed by administrators and may take a large amount of time. Although this situation does not strictly involve any failure, it prevents users from executing jobs due to the deployment of new software and potential misconfiguration. During this period, the user views the computing infrastructure as totally unavailable.
- *Application-dependant problems*: When a service required for the execution of an application fails or is not available, administrators are responsible for restarting the service (users do not have the required privileges [1]). While the failure is being fixed, applications using the broken service fail. As a consequence, the resource is seen as unavailable for some users, while others remain unaffected.
- *Middleware failures*: Due to the distributed nature of grid middlewares, failures can involve different components

and their effects may vary. A failure in key components, which represent a single point of failure, may cause total unavailability and the failure of all executing jobs, whereas a failure in a secondary component may have no effect on users. In our particular case, since the framework could be seen as a meta-middleware, these failures may appear at the framework level and the computing infrastructures level.

The previous faults involve different availability problems ranging from situations where less resources are available to states where the complete infrastructure becomes unavailable. From the reliability point of view, there could be no effect at all or failures in all executing jobs. To detect and repair some of these faults, grid middlewares integrate different fault-tolerant mechanisms. In general, they are only able to detect the most simple ones (computing resource failures) and they cannot recover from all detected faults [3]. Additionally, some middlewares provide techniques to mitigate the effect of faults, such as checkpointing [5] or over-provisioning [23][24]. In any case, these techniques are only useful to recover from computing resource failures where some resource becomes unavailable and a few jobs fail. More critical problems involving total unavailability and unreliability are much more difficult to manage. These problems lead to situations where users cannot execute any job and lose a valuable time waiting for the fault to be fixed.

B. A hierarchical strategy for handling grid/cluster faults

Once the effects caused by these faults are understood, a strategy to handle them can be implemented and integrated. We propose a solution based on the hierarchical management of faults. Firstly, when the execution of a job fails, the fault is locally managed by the computing infrastructure where the job was being executed (a kind of local strategy). The local fault tolerance mechanisms are responsible for detecting and handling this kind of failures. In some cases, these mechanisms can collaborate with the mediator component that manages the access to the infrastructure in order to react to the fault: for instance, if the execution of a job has failed, the mediator can locally submit it to a different computing resource of the same infrastructure.

If the fault persists after taking corrective actions at the level of the computing infrastructure, it is dispatched to the execution layer. More specifically, the mediator of the faulty infrastructure sends a fault message to the message bus. A new management component for fault handling has been integrated into the framework execution layer. This component is responsible for catching fault messages and guarantees the successful execution of jobs using reliable computing resources. In our approach, the job could be submitted to another computing infrastructure or, as a last option, cloud computing resources could be used by the component to execute the job.

Therefore, for these types of faults the proposed solution consists of two levels of fault handling: firstly, at the specific computing infrastructure level, and, secondly, at the execution framework level.

C. Improving the framework reliability and scalability

The architecture of the proposed framework favours the management of faults at the software components level (mediators, management components and the message bus). When a mediator or a management component fails, its functionality is disabled. In the case of a mediator, the access to the computing infrastructure managed by it is closed; whereas in the case of a management component, the capabilities of the framework (scheduling, data movement, fault tolerance, etc.) are reduced. Both situations can be solved using the same solution: integrating into the framework multiple instances of the same component. Let us remember that in the proposed solution components can be added or removed without disturbing the execution of other existing ones and multiple instances can work together without interfering with each other.

On the other hand, the message bus is the core component of the framework. How can we make this component reliable and scalable? In order to deal with the first issue, the message bus has been deployed in a virtual machine provided by Amazon EC2 [25]. For the scalability issue, a new version of the message bus has been implemented. Now, the bus is distributed through several computing nodes (virtual machines) and new elastic capabilities (inspired by cloud behaviour) have been integrated into it. The bus is able to monitor its internal state (number of messages, response time, throughput, etc.) and predict when its performances or capabilities might be compromised. When some of these undesired states is detected, new computing nodes can be dynamically added to host message exchanges.

In the following sections, we go deeper into these aspects.

IV. MANAGEMENT OF AVAILABILITY AND RELIABILITY ISSUES

The characterization presented in Section III shows that, in large-scale distributed computing infrastructures, there are a lot of problems leading to temporal or permanent unavailability states and job failures due to reliability issues. As a result, users experience severe delays in both submission and termination of their jobs and unexpected end statuses. To tackle this problem, we have extended the mediators with monitoring capabilities. A hierarchical fault management mechanism is proposed, enabling the framework to manage faults at different levels using several fault recovery policies. This reduces the overhead of the message bus and the time required to handle failures. We also propose the use of public clouds as reliable computing infrastructures for the execution of jobs that systematically fail in the integrated computing infrastructures.

A. Solution design

Mediators have been extended with an *Infrastructure Monitor* and a *Local Fault Manager*. Figure 2 shows the mediator architecture for this approach. The *Job Submission* process and its related components have been simplified (for more details about the job submission, please refer to [14]), as we will focus on monitoring and fault management.

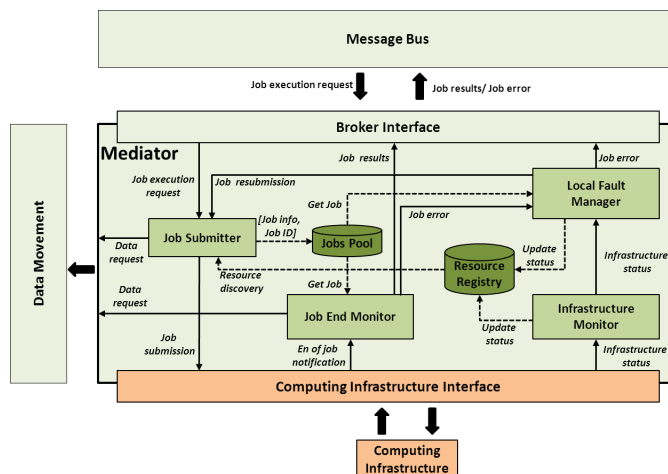


Figure 2: Simplified Architectural design of a generic mediator to lead with unavailability and failure events.

Let us briefly depict the process carried out in the mediator. First, the *Job Submitter* receives job execution requests, retrieves the input data required for the job execution (if necessary) and commands its execution to the computing infrastructure. After that, the job description and job identifier provided by the infrastructure are stored in the *Jobs Pool*. When a job finishes its execution, the *Job End Monitor* fetches the job description from the *Jobs Pool* using the job identifier. Then, it checks the log and error stream files as well as the existence of the output files defined in the job description. If an error is detected or the output have not been generated, the information about the error is passed to the *Local Fault Manager*. Otherwise, the output data are moved to the destination specified in the job description and the results are sent to the message bus.

The mediator can detect failures and unexpected job terminations. However, in order to avoid such situations, the *Infrastructure Monitor* periodically checks the status for resource availability. With this information, it updates the *Resource Registry* and notifies the *Local Fault Manager* if any availability problem is detected. The *Local Fault Manager* is the component responsible for taking decisions when a job fails or an unavailability state is detected. Its design is similar to the *Global Fault Manager* presented in [14]. A rule-based engine is used as the decision maker. The set of rules can be modified at runtime, providing adaptation capabilities for specific scenarios. Therefore, different policies can be used depending on the underlying computing infrastructures, execution traces or system load, for instance.

When a job fails or an unavailable state is detected, the *Local Fault Manager* can decide to either re-execute the involved jobs or notify the *Global Fault Manager*. In the first case, the re-execution process remains internal to the mediator. This approach reduces the overhead of the message bus and the time required to handle the failure. In the last case, a message with error information and the job description is sent to the *Global Fault Manager* (via the message bus).

Finally, the *Global Fault Manager* (namely *Fault Manager* in Figure 1) retrieves messages with information about faulty jobs and chooses a computing infrastructure to re-execute them on or notifies to the user if the fault is not recoverable (for example, because the server hosting input data is down). In case of a recoverable fault, the following approach is used: if it is the first failure, another computing infrastructure is selected; if it is the second failure, a reliable infrastructure is selected; finally, if the third failure is reached, the error is propagated and the user is notified.

We propose the use of public cloud resources as reliable infrastructures because they provide the opportunity of executing jobs in a well controlled and previously defined environment. Cloud resources are less sensitive to resource failures through virtualization and migration techniques. Clouds also provide high availability and reliability, and they supply "infinite" on-demand resources in a pay-per-use model.

B. Evaluation

In collaboration with the Intelligent Systems Group of the University of Santiago de Compostela (Spain), we have solved a computing-intensive problem in the field of linked data. The problem consists of extracting a set of significant terms from learning units. Each set of terms must be semantically annotated with relevant contextual information extracted from the DBPedia [26]. This problem requires the execution of about 20000 jobs for a whole week. As a consequence, it is very sensitive to faults. We have used this experiment as a benchmark for the proposed hierarchical fault management system.

Figure 3 shows the failure rates obtained using different policies in the local fault manager (no fault recovery, one resubmission and two resubmissions) and the global fault manager (no fault recovery, resubmission on an alternative computing infrastructure, resubmission on an Amazon EC2 resource and a combination of the two last ones). As it can be observed, using public cloud resources allows us to recover from any failure (except failures due to unreachable input data or bad definition of jobs). Otherwise, if we only use the integrated local infrastructures, there are some jobs that still fail after several executions due to unavailability and unreliability states of computing infrastructures.

Besides, the hierarchical approach presented reduces both the message bus overhead and the time required to handle the fault. In the experiments, we have observed that the average time required to handle a fault with our previous design was 1071.23 milliseconds, and the hierarchical design reduces this time to 143.21 milliseconds. When a job fails for the first time in the hierarchical approach, its management remains internal to the mediator. In the previous (non-hierarchical) design, a message was introduced into the message bus and then retrieved by the Global Fault Manager, which would take the decision of resubmitting the job to the same infrastructure (so a new message was written in the message bus and then retrieved by the corresponding mediator in order to submit the job again).

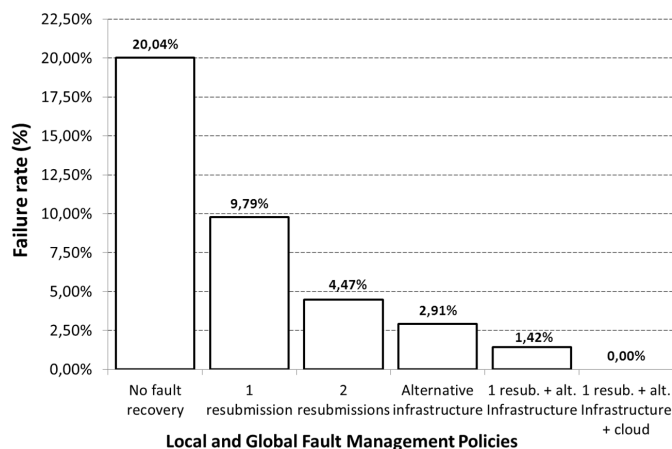


Figure 3: Failure rate for different fault management policies.

The percentage of faults detected by the mediator with respect to the number of total job failures has been also measured. Without infrastructure monitoring, some failures were not detected because the management middleware did not notify them, the middleware itself failed or the computing infrastructure was down. Currently, the Infrastructure Monitor is able to detect these situations and help mediator handle all failures. As a result, the percentage of job failures detected has increased from a 91.92% to a 99.99%.

V. IMPROVING FRAMEWORK SCALABILITY THROUGH AN ELASTIC MESSAGE BUS

Scalability is one of the main challenges of any distributed system. In cluster and grid computing, scalability focuses on the number of computing resources available as well as the flexibility to integrate new ones. The scalability of the management system plays a very important role in the improvement of the quality of service experienced by end-users in terms of response times and system crashes.

The message bus is the backbone of the proposed architecture. In order to make the system more scalable, we propose an elastic design, taking advantage of the dynamic scaling provided by cloud systems. As it will be shown, with respect to the previous message bus version, the use of a cloud-based solution improves both scalability and reliability.

A. Solution design

To deal with scalability issues, we have extended the original design of DRLinda, a distributed message bus based on the Linda coordination model [27]. The main idea behind DRLinda is the use of several nodes implementing message repositories to host messages in a distributed way. We have extended this approach to dynamically scale the number of nodes depending on the number of messages and message access frequency when the system is running.

The previous implementation of DRLinda could dynamically vary the number of nodes used to lead with bursts of requests. However, these changes must be performed manually and only local resources can be used. A cloud-based

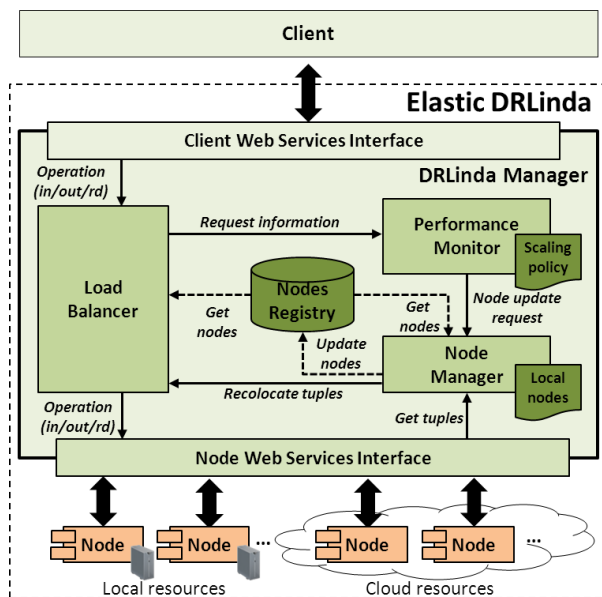


Figure 4: Message bus architecture overview.

elastic design allows self-configuration and auto-scaling of the number of nodes used at any moment. The new architectural message bus design is sketched in Figure 4. The approach includes two new components in addition to the existing *Load Balancer* [27]: a *Performance Monitor* and a *Node Manager*.

The *Performance Monitor* is a component that receives and processes information on client requests and collects metrics such as number of requests, response time or throughput, averaged for the last requests. The results of these metrics and time conditions (for example, time since the last scaling request) can then be used to define the scaling conditions. When a condition is satisfied (scaling up/scaling down), this component communicates with the *Node Manager* to deploy or release a node.

The *Node Manager* is responsible for allocating new resources and releasing unnecessary ones. When a new resource is requested, the *Node Manager* looks for a new local resource that becomes a DRLinda node. If there are no available local resources, it gets a cloud instance. In this way, physical local resources and virtualized cloud resources can be used at the same time to provide good quality of service. Also, when a resource must be released, the component selects the most appropriate one and manages message transfer between the involved nodes, via the *Load Balancer*. To reduce costs, cloud resources are only released when they are about to fulfil an entire hour of use (due to the hourly billing model of the cloud provider we have used). Consequently, if there is a pending release request when a cloud resource is going to complete an entire hour, that resource is released. Also, if a cloud resource can be released but there is no request, a local resource (if available) is used to replace it.

B. Evaluation

To measure the efficiency and scalability of both architectural designs, we have used the methodology proposed in [27][28]. In these experiments, a set of clients access the message bus. Every client warms-up the message bus by inserting a random number of messages (between 1500 and 5000) and then iterates 2000 times through the following sequence of operations: first, it executes an *out* operation (send a new message), then waits for a random time ($T_{delay} \in [200, 250]$ ms), and then retrieves the same message (operation *in*). When completed, the client terminates. A detailed justification of the parameters used for the experiment can be found in [28]. The size of the messages has been set accordingly to the problem we are managing. JSDL messages extracted from the experiments presented in [22] have been used, which an average size of 63 Kbytes.

Figure 5a shows the average response time observed both in the original DRLinda implementation and the new elastic design. In both cases, we have used m1.medium Amazon Elastic Compute Cloud (Amazon EC2) [25] instances as resources to host message bus components. For the experiments, the former DRLinda was deployed over 25 nodes. On the other hand, in the case of the elastic solution, only two nodes were initially used, and then new nodes were added under request (up to 70 nodes were registered during the experiment). Obviously, the dynamic scalability introduces an overhead as the message space must be redistributed. However, the overhead is not significant compared to the response time and the throughput in terms of Input/Output Operations Per Second (IOPS). As it can be seen in Figure 5a, the response time improves very significantly when using the cloud-based solution. This is due to a more efficient load balancing in every node. While in the case of the former DRLinda the nodes have to support a higher load, the use of an elastic approach allows to keep nodes at optimum levels of occupation and CPU and memory loads.

On the other hand, the results in Figure 5b depict the throughput in terms of IOPS. As shown, the use of a cloud-based elastic approach reports several benefits. First, the number of concurrent clients supported by the bus scales with no problem over the maximum number of clients. Moreover, the IOPS only decrease because of the overhead of space distribution, but remain in the range of [1100,1200] milliseconds for a huge number of simultaneous clients.

The experiments have also shown how the use of an elastic solution allows to extend the number of concurrent clients without suffering severe delays or service interruptions. Therefore, it is a successful mechanism to avoid bottlenecks in the message bus.

VI. RELATED WORK

There are several works seeking to improve understanding of failures in Grid environments. However, none of these studies analyse failure impact on end users. In [2], a taxonomy for the classification of Grid faults is proposed. The taxonomy presents several perspectives for the classification of Grid failures (origin, duration, consequences, etc.) but

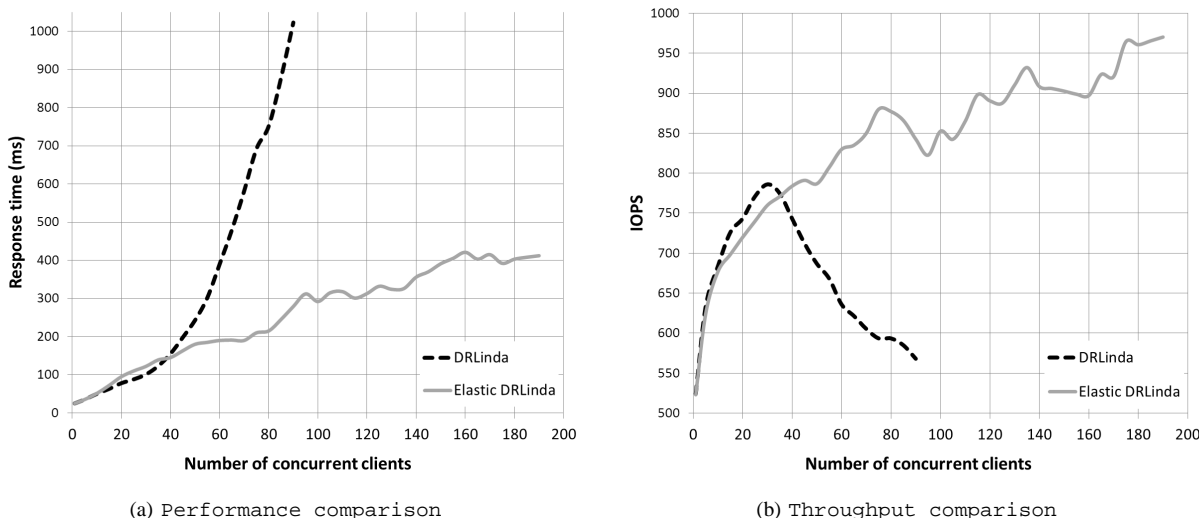


Figure 5: Performance comparison between former DRLinda and elastic DRLinda message bus in terms of: (a) response time and (b) Input/Output Operations Per Second.

it lacks an analysis of causes and effects that could help in handling failures. In [1] and [3], different surveys about Grid failures are presented. On the one hand, in [1], failures are classified as configuration, middleware, application and hardware failures. The main concerns and problems regarding fault management are studied showing that end users are highly involved in fault detection and recovery, failures are mostly due to misconfiguration and recovery mechanisms are application-dependant. On the other hand, in [3], hardware, operating system, middleware, task, workflow and user related failures are identified. Also, detection, prevention and recovering capabilities of several workflow management systems are analysed concluding that current systems are not able to properly manage faults.

With regard to scalability and dynamic autoscaling of resources, [29] analyses existing mechanisms to dynamically scale applications in clouds at three different levels: server, network and platform. [30] shows a technique to dynamically scale cloud resources up and down considering performance and budget information. This technique is based on acquiring enough instances to meet application deadlines and shutting down unnecessary instances when an hour is going to be fulfilled. In [31], look-ahead optimizations are used to predict future workloads and scaling applications while cost remains low. However, results are limited to scenarios with few resources and accurate predictions. On the contrary, in [32], profiles are used to provide just-in-time scalability for cloud applications in environments with unpredictable workloads. Profiles capture application characteristics, architecture and topology, scaling conditions and mechanisms to automate the deployment and release of new resources.

Finally, different proposals use public Cloud resources to improve job completion rates and to meet the deadline of QoS-constrained jobs. In [33], a rescheduling algorithm is used

to deal with grid performance fluctuations. When a job ends after its estimated finish time, a job waiting for execution in the same Grid resource is selected to be executed in a cloud resource. Meanwhile, in [23], task replication is used to reduce the makespan and cost of workflows executed in Grids and Clouds. An unreliable pool of resources is used to execute jobs in first instance, while reliable resources (formed by public Cloud instances and own resources) are used to execute replicated jobs in the tail phase of BoTs (Bag of Tasks). A similar approach is used in [24], where jobs are first scheduled in clusters and Grids, then some jobs are replicated to increase their success probability and finally public cloud resources are used as backup if additional replication is required.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have identified and analysed several availability and reliability problems from the users' point of view in the context of a framework to execute scientific workflows in heterogeneous computing environments. This analysis has allowed us to identify common situations where job fails and users cannot execute any job.

To increase framework availability and reliability, two cloud-based solutions have been proposed: an elastic design of the message bus and a hierarchical fault management. On the one hand, the elastic design of the message bus allows the framework to deal with bursts of requests providing high quality of service at a low cost. On the other hand, managing faults hierarchically results in a better treatment of faults by applying different policies at different levels, faster fault-recovery and less overhead in the framework. Also, using public clouds as reliable computing infrastructures allows the framework to execute jobs even in total unavailability and total unreliability situations, reducing the failure rate experienced by end-users.

As future work, we will study techniques to reduce the cost of the proposed solutions without decreasing the quality of service and job completion rates. Also, we will define reliable scheduling policies to increase the number of jobs successfully completed in their first execution. Finally, we will explore the use of Amazon Simple Queue Service (Amazon SQS) [34] in replacement of the Linda-based message bus to improve performance, availability and reliability of the proposed framework.

ACKNOWLEDGMENT

This work has been supported by the research project TIN2010-17905, granted by the Spanish Ministry of Science and Innovation, and the regional project DGA-FSE, granted by the European Regional Development Fund (ERDF).

REFERENCES

- [1] R. Medeiros, W. Cirne, F. Brasileiro, and J. Sauvé, "Faults in grids: Why are they so bad and what can be done about it?" in *Proceedings of the 4th International Workshop on Grid Computing*, ser. GRID '03, vol. 0, 2003, pp. 18–24.
- [2] J. Hofer and T. Fahringer, "A multi-perspective taxonomy for systematic classification of grid faults," in *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, ser. PDP '08, 2008, pp. 126–130.
- [3] K. Plankensteiner, R. Prodan, T. Fahringer, A. Kertész, and P. Kacsuk, "Fault-tolerant behavior in state-of-the-art grid workflow management systems." CoreGrid, Tech. Rep. TR-0091, 2008.
- [4] C. Dabrowski, "Reliability in grid computing systems," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 8, pp. 927–959, 2009.
- [5] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *J. Grid Comput.*, vol. 3, no. 3-4, pp. 171–200, 2005.
- [6] O. Khalili, J. He, C. Olschanowsky, A. Snaveley, and H. Casanova, "Measuring the performance and reliability of production computational grids," in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (GRID 2006)*, 2006, pp. 293–300.
- [7] P.-O. Östberg and E. Elmroth, "GJMF - a composable service-oriented grid job management framework," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 144–157, 2013.
- [8] P. Kacsuk, G. Dózsa, J. Kovács, R. Lovas, N. Podhorszki, Z. Balaton, and G. Gombás, "P-grade: A grid programming environment," *J. Grid Comput.*, vol. 1, pp. 171–197, 2003.
- [9] Q. Wu, M. Zhu, Y. Gu, P. Brown, X. Lu, W. Lin, and Y. Liu, "A distributed workflow management system with case study of real-life scientific applications on grids," *J. Grid Comput.*, vol. 10, no. 3, pp. 367–393, 2012.
- [10] E. Huedo, R. S. Montero, and I. M. Llorente, "A framework for adaptive execution in grids," *Softw. Pract. Exper.*, vol. 34, no. 7, pp. 631–651, 2004.
- [11] I. Rodero, J. Corbalán, R. M. Badia, and J. Labarta, "eNANOS grid resource broker," in *Proceedings of the 2005 European conference on Advances in Grid Computing*, ser. EGC '05, 2005, pp. 111–121.
- [12] L. Adzigogov, J. Soldatos, and L. Polymenakos, "EMPEROR: An OGSA grid meta-scheduler based on dynamic resource predictions," *J. Grid Comput.*, vol. 3, no. 1-2, pp. 19–37, 2005.
- [13] A. Kertész and P. Kacsuk, "GMBS: A new middleware service for making grids interoperable," *Futur. Gener. Comp. Syst.*, vol. 26, no. 4, pp. 542–553, 2010.
- [14] J. Fabra, S. Hernández, P. Álvarez, and J. Ezpeleta, "A framework for the flexible deployment of scientific workflows in grid environments," in *Proceedings of the Third International Conference on Cloud Computing, GRIDs, and Virtualization*, ser. CLOUD COMPUTING '12, 2012, pp. 1–8.
- [15] N. Carriero and D. Gelernter, "Linda in context," *Commun. ACM*, vol. 32, no. 4, pp. 444–458, 1989.
- [16] Aragón Institute of Engineering Research (I3A). (2013) <http://i3a.unizar.es>. Accessed 15 March 2013.
- [17] HTCCondor Middleware. (2013) <http://research.cs.wisc.edu/htcondor/>. Accessed 15 March 2013.
- [18] gLite Middleware. (2013) <http://glite.cern.ch/>. Accessed 15 March 2013.
- [19] Institute for Biocomputation and Physics of Complex Systems (BIFI). (2013) <http://bifi.es/en/>. Accessed 15 March 2013.
- [20] AraGrid. (2013) <http://www.aragrid.es/>. Accessed 15 March 2013.
- [21] PireGrid. (2013) <http://www.piregrid.eu/?idioma=english>. Accessed 15 March 2013.
- [22] S. Hernández, J. Fabra, P. Álvarez, and J. Ezpeleta, "A Simulation-based Scheduling Strategy for Scientific Workflows," in *Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, ser. SIMULTECH '12, 2012, pp. 61–70.
- [23] O. A. Ben-Yehuda, A. Schuster, A. Sharov, M. Silberstein, and A. Iosup, "EXPERT: Pareto-Efficient Task Replication on Grids and a Cloud." in *Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '12, 2012, pp. 167–178.
- [24] L. Ramakrishnan et al., "VGrADS: enabling e-Science workflows on grids and clouds with fault tolerance," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09, 2009, pp. 47:1–47:12.
- [25] Amazon Elastic Compute Cloud (Amazon EC2). (2013) <http://aws.amazon.com/ec2/>. Accessed 15 March 2013.
- [26] M. Lama, J. C. Vidal, E. Otero-García, A. Bugarín, and S. Barro, "Semantic linking of learning object repositories to DBpedia," *Educ. Technol. Soc.*, vol. 15, no. 4, pp. 47–61, 2012.
- [27] J. Fabra, P. Álvarez, and J. Ezpeleta, "DRLinda: A Distributed Message Broker for Collaborative Interactions Among Business Processes," in *Proceedings of the 8th International Conference E-Commerce and Web Technologies*, ser. EC-Web '07, 2007, pp. 212–221.
- [28] D. Fiedler, K. Walcott, T. Richardson, G. M. Kapfhammer, A. Amer, and P. K. Chrysanthis, "Towards the measurement of tuple space performance," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 3, pp. 51–62, 2005.
- [29] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 45–52, 2011.
- [30] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing*, ser. GRID '10, 2010, pp. 41–48.
- [31] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings of the IEEE 4th International Conference on Cloud Computing*, ser. CLOUD '11, 2011, pp. 500–507.
- [32] J. Yang, J. Qiu, and Y. Li, "A profile-based approach to just-in-time scalability for cloud applications," in *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, ser. CLOUD '09, 2009, pp. 9–16.
- [33] Y. C. Lee and A. Y. Zomaya, "Rescheduling for reliable job completion with the support of clouds," *Future Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1192–1199, 2010.
- [34] Amazon Simple Queue Service (Amazon SQS). (2013) <http://aws.amazon.com/sqs/>. Accessed 15 March 2013.

Eliciting Risk, Quality and Cost Aspects in Multi-cloud Environments

Victor Muntés-Mulero and Peter Matthews

CA Technologies

CA Labs Europe

Email: {Victor.Muntes, Peter.Matthews}@ca.com

Aida Omerovic

SINTEF

Oslo, Norway

Email: aida.omerovic@sintef.no

Alexander Gunka

BOC Information Systems

Austria

Email: alexander.gunka@boc-eu.com

Abstract—With the increasing number of providers offering cloud-based services, new opportunities arise to build applications capable of avoiding vendor lock-in issues. Such applications are developed in multi-cloud environments that allow replacing services with those offered by alternative providers. While this may improve quality and provide independence from a single cloud service provider, it also brings new risks. Being able to assess risks and those quality aspects that are specifically related to multi-cloud environments is essential in order to design reliable applications based on the use of cloud services. Although a lot of work has been done to study risks and quality aspects for cloud services, this is usually focused in single-provider scenarios. In this position paper, we discuss several risks and quality aspects that are specifically related to multi-cloud environments.

Keywords- *Multicloud, Risk assessment, Quality prediction, Cost prediction*

I. INTRODUCTION

Many applications and Cloud Service Providers (CSPs) replicate or combine services from multiple clouds or multi-clouds (also called cloud mashups [9]) to avoid the risk of vendor lock-in. New architectures, technologies, and standards are being proposed to support collaboration among multiple cloud systems [1], [2], [6], [7]. Although direct collaboration among applications hosted by different clouds is still restricted [9], the adoption of these proposals will improve the ease of migration from one provider to another and increase open competition. Nevertheless, the current environment already offers many opportunities for collaboration among services offered by different providers without requiring standards or important changes to the delivery model.

In multi-cloud environments, it is essential to provide tools that guide multi-cloud application architects to choose the services providing the necessary quality and ensuring acceptable level of risk. Previous work has focused on describing quality aspects and metrics to measure the suitability of a cloud service from a multi-dimensional perspective. An example of this is the Service Measurement Index (SMI) [10], a framework designed to allow for quick and reliable comparison of IT business services. SMI establishes the basis for comparing isolated services in regard of several categories such as for instance accountability, agility or assurance. However, they do not explicitly analyze these aspects in a multi-cloud context.

Based on this quality aspects and other factors, model-based decision making system help application designers to choose the cloud components that better fit their needs. Some

of these major factors include functional and non-functional properties, as well as cost and the added value. A trade-off between such factors is the basis for decision making. This trade-off is particularly complex between the non-functional factors, the variable parts of the architecture, and the cost of the selected solutions. The variability, as well as incomplete information or knowledge, are also sources of risk. Since functional requirements are less flexible and specified rather early, and since the added value is strongly related to functional properties, the factors that are tuneable and highly interrelated are risk, quality and cost.

In this paper, we discuss the risks related to cloud services in a multi-cloud environment, the quality aspects that are specific to that environment and make some cost considerations. We analyze three important issues which are essential in multi-cloud environments: interoperability issues between services offered by different providers, the ease of migration from a current service to a new equivalent service, and the security issues that arise from the fact that confidentiality, integrity, availability, etc. does not depend on a single provider.

This paper is organized as it follows. Section II presents related work. Section III briefly describes multi-clouds scenarios and describes the aspects considered in this paper. Section IV presents a summary of quality aspects to be considered. Section V provides a brief description of costs that must be taken into account in this type of environment. In Section VI, we discuss risks that must be considered in a multi-cloud. Finally, Section VII presents the conclusions and draws some future work.

II. RELATED WORK

As a basis for the elicitation of the adequate quality characteristics, the software product quality standard ISO/IEC 9126 defines quality as the totality of features and characteristics of a software product that bear on its ability to satisfy stated and implied needs. The ISO 9126 standard provides an established specification of decomposed quality notions with their qualitative and quantitative definitions. The standard defines a quality model for external and internal quality, and for quality in use. The characteristics of the internal and external quality model are functionality, reliability, usability, efficiency, maintainability and portability. These are in turn decomposed into a total of 34 sub-characteristics.

SMI [10] is a standardization effort from the Cloud Services Measurement Index Consortium (CSMIC) consisting of

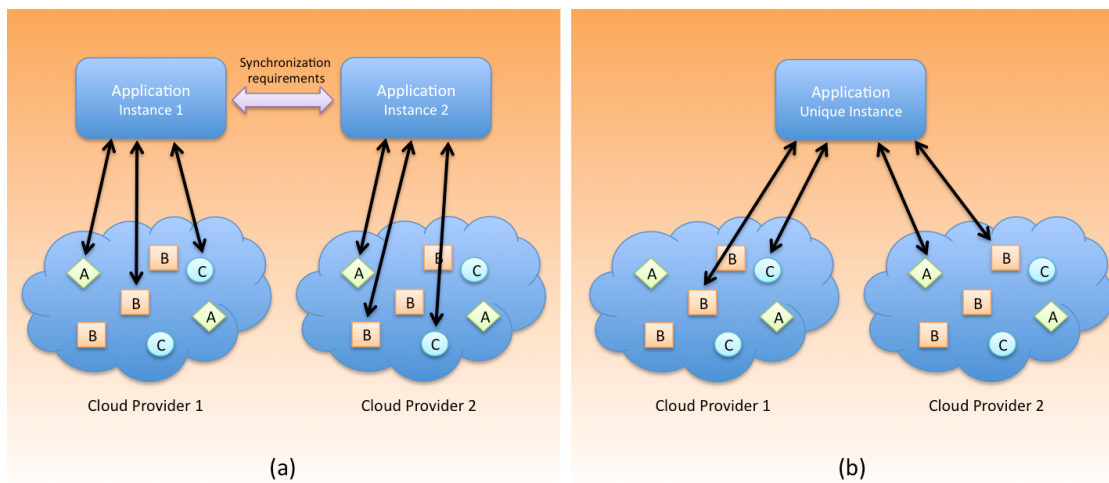


Fig. 1: Examples of two different multi-cloud scenarios

academic and industry organizations. The Service Measurement Index (SMI) uses a series of characteristics and measures to create a common means to compare different services from different suppliers. The characteristics are categorized as Usability, Performance, Agility, Security and Privacy, Financial, Assurance and Usability. Each of these characteristics has a number of measures that can be used to evaluate the risk in using a service. For example in the accountability category one of the measured attributes is Compliance and another is Service-Level Agreements (SLA) verification both of which can be used to create a risk measure for the service and the provider. The work presented in this paper is based both on the ISO standard and SMI conclusions.

In order to enable risk monitoring based on indicators or metrics, there is a need not only to identify the relevant indicators, but also to understand how to relate the indicators to potential risks, and how to aggregate the monitored values into risk levels [5]. In this paper, we identify both risks and quality aspects related to multi-cloud environments. To our knowledge, none of the previous work has been focused on jointly analysing risk, quality and costs in a multcloud environment.

III. MULTI-CLOUD SPECIFIC NEEDS AND CHALLENGES

We define a *multi-cloud application* as any piece of software using several cloud services hosted by two or more different providers. Usually, two different scenarios are considered when referring to multi-cloud environments. Figure 1 depicts these two cases. In the first case (a), an application is replicated to improve resilience, and may also be used to avoid vendor lock-in. This means that the application has two independent instances using the same type of cloud services (A, B, C in the figure) in two different cloud providers. In the second case (b), a single instance of the application runs different cloud services hosted by two or more cloud providers. In this latter case, it is also possible to replicate services to ensure availability. This would also imply synchronization. Because of the need for high interoperability between services offered by different providers, scenario (b) is in general more complex to manage and may potentially involve larger risk compared to (a). In fact,

scenario (a) may be considered a particular case of scenario (b). Because of this, we focus on scenario (b) in this paper.

The use of multiple cloud services from multiple providers adds a new dimension of complexity to an already complex cloud computing scenario. Heterogeneity caused by the existence of independent providers that have created their own business models, protocols, processes and formats generates an increasing number of risks to be taken into account when creating a new application using a multi-cloud strategy. In this paper, we emphasize three essential aspects that must be considered in a multi-cloud environment:

- **Heterogeneity of services offered by different providers results in reduced interoperability:** the lack of standard interfaces for services in different clouds and the creation of independent proprietary systems by each provider, make multi-cloud environments very heterogeneous. Interoperability problems may range from technical issues, such as messaging interfaces or quality of service, to semantic, organizational or legal issues. This heterogeneity is an important risk to consider at design time, since it will influence the capacity of an application architect to decide between one service and another. In terms of quality, a service will be highly interoperable with other systems if it can be combined in collaboration with many other services, from the same or other cloud service providers.
- **Migration between services offered by different CSPs is an essential operation to ensure the compliance with the application requirements:** one of the most common reasons to deploy an application in a multi-cloud environment may include increasing the cloud service catalog and increasing the capacity of users to migrate from one service to another in case the requirements on the application are not fulfilled. We call this capacity *replaceability*, and it represents the ease to migrate from one service to another to replace the first one. It will be essential to decompose migration processes from one cloud service to another into several finer-grained steps, and analyze the quality

aspects to be considered in the process.

- Security threats are increased in multi-cloud computing environments:** increasing the number of services and providers, will increase the complexity of the overall system and the number of potential attacks. Control over customers data decreases, especially because of potential migration between services of different providers. The continuous communication of data between services in different clouds may also result in storing data in intermediary less secure external storage systems, increasing the overall vulnerability and potentially compromising confidential information. In terms of data privacy, multitenancy makes it more difficult to guarantee confidentiality of sensitive information.

These three aspects have been selected and prioritized after several interviews with industrial and academic partners. They have been chosen based on experience and from studying different migration processes. They represent three essential requirements in a multi-cloud environment: coordination between services offered by different providers, capacity to replace a service by another one, and the increase of complexity in the system increasing possible points of failure in terms of security. Note that, we do not claim this to be a comprehensive list of possible aspects to analyze, but we believe they are a good starting point to establish the basis to define risk and quality in multi-clouds.

IV. QUALITY ASPECTS IN MULTI-CLOUD ENVIRONMENTS

In this section, we analyze those quality aspects related to the issues detected in Section III that must be considered in a multi-cloud environment: interoperability, replaceability and security. Figure 2 summarizes the quality aspects considered related to these three issues.

A. Interoperability

The interoperability problems of cloud services in the controlled environment of a single CSP, are exacerbated by mixing services from different providers and may imply incompatibilities in other areas of a mixed service implementation. From the point of view of a developer, it will be important to know the degree of interoperability of a certain service with respect to other services it must interact with. Figure 3 depicts the scenario studied in this case. Figure 2 divides these incompatibilities in four different areas: technical, semantic, organizational and legal. The Technical interoperability quality aspects refer to the capacity of two or more services offered by different providers to communicate through common protocols and to jointly guarantee a certain quality of service. For instance, possible indicators that might be used to evaluate the degree of technical interoperability might be the number of standardized interfaces that can be compared towards the total number of interfaces used by the service, or the average recovery time of the service or other performance aspects. Semantic aspects refer to aspects related to the data syntax consistency and the data quality. These data related aspects are relevant for interoperability since only two or more services offering mechanisms to guarantee global data properties might be combined in the same application. Organizational aspects

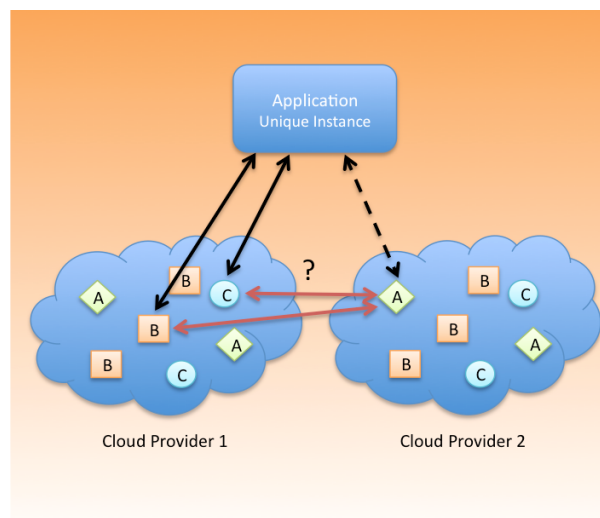


Fig. 3: Interoperability in a multi-cloud environment: services offered by different providers interacting with each other.

indicate how adaptable a service is to several work processes. Since each of these work processes might be established by different providers, it is important that a service in a multi-cloud environment is adaptive to fit the requirements of each work process in each case. Changes in a work process may require changes in a specific cloud service that is already used. In a migration process, choosing a new cloud service candidate to replace an existing service may depend on the capacity of this new service to adapt to the existing work process. Compliance with existing cloud service standards in terms of role and functionality of that specific cloud service will be essential to ensure good organizational interoperability. Regarding legal aspects, we focus on regulatory compliance. Compliance in this case may be understood as a list of laws that are observed by the service provider. Some may be mandated by the customer such as Sarbanes-Oxley [8], some by government, e.g. Data Protection act [3]. It is the presence or absence of compliance that is of interest. A purchasers compliance officer will provide a number of regulations that any service would have to observe and these would be part of the requirements gathering.

Several aspects are likely to be difficult to measure. A good example is the number of standards in the communication capability aspect. Standards for cloud service communications are evolving and several attempts have been made to create an agreed list of them. NIST has a list of recommended standards and the European Commission has created a Cloud Standards Coordination (CSC) that is being administered by ETSI [4]. The requirements of multi-cloud applications may need some or all of the relevant standards to be adhered to.

B. Intercloud Replaceability

Migration is an essential operation linked to multi-cloud environments. The capacity of a software architect to redesign an application and replace existing services by other services with the same or similar functionalities defines in fact the realism of considering cloud mashups. For instance, a cloud database service may integrate application building tools that

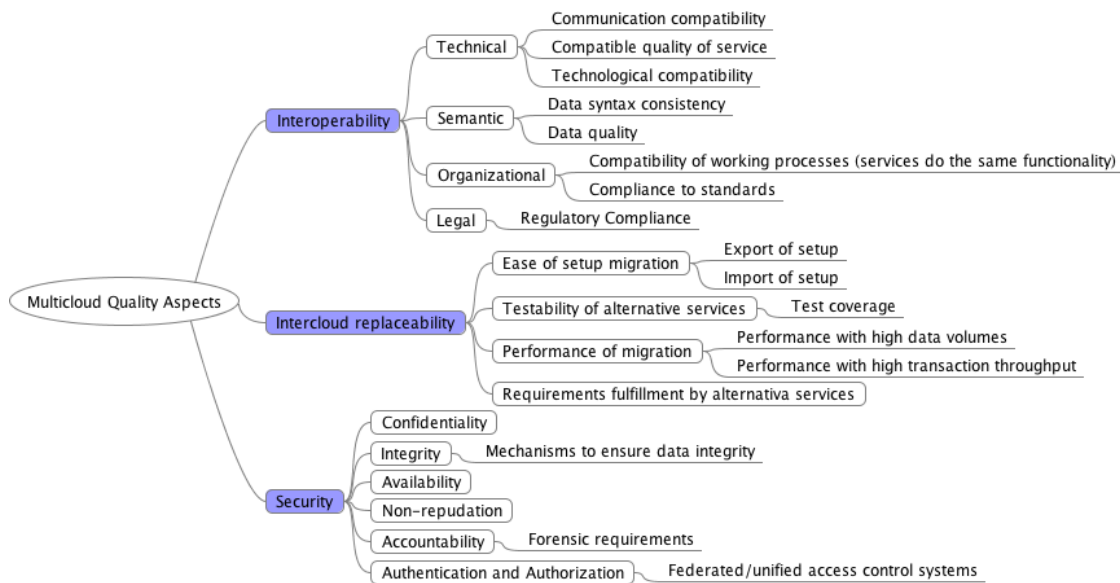


Fig. 2: Quality aspects related to multi-cloud environments

might be used by our system, such as APIs based on web services standards. If the other services interacting with this cloud database service assume that these tools exist, moving to a new cloud database that it does not provide these tools will require reengineering part of our system and it may have an unaffordable cost. In this subsection, we define and analyse the migration process to find the quality aspects that make a service easy to migrate from. We focus on the case where a service is replaced by one or more services offered by a different cloud provider. We consider two situations:

- The current cloud service does not fulfill the requirements of the system: this may happen for instance when the service is updated or modified, when the amount of information handled by the application grows making it impossible to comply with certain pre-established SLAs, etc. Usual examples may range from a variation in the cost that makes the service not competitive compared to other services of the same type, to a change in policies and functionalities that affects security, availability, resilience, or any other important aspect.
- The requirements of the system have changed: one or more cloud services may not fulfill these new requirements and need to be replaced.

Figure 4 depicts a generic process of service-to-service migration. First, a cloud service is selected for migration. Depending on the reason for migration, it may be necessary to review the requirements defined at design-time. After this, one or more new candidate cloud services must be selected. In order to simplify this step, Figure 4 considers a single candidate in the process. Once we have found a candidate target service to migrate to, we can export both data and the configuration from the original service. At this point, it is usually necessary to enter an intentional contract with the new service provider. In some cases, it will be also necessary to inform the old service that we are initiating a process to retire it. In this situation, the old service and the new one

may be active at the same time during the testing and training process. This will depend on the availability requirements of the application migrating one of its cloud-based components. In the next step, it is important to adjust or define a new workflow for the application. This might be necessary if the new service is not perfectly compatible with the old one or if the application was redesigned in a way that the workflow was altered. After this, we can start preparing the testing environment and the new service. Usually, the testing process will be divided in several phases.

In general, it is necessary to carry out functionality and performance testing in a test environment. In this situation, data needs to be kept synchronised. Following successful functionality and performance testing, the service may move to a modification of A/B testing so that the application is tested with the new service in production before switching over completely. In case requirements are not satisfied, we must start the process again. If they are fulfilled, we can start the users training process and eliminate the old service if this is still active. Once this has been done, the application can be deployed again using the new cloud service.

Figure 2 shows several quality aspects related to replaceability. Possible indicators of quality related to intercloud replaceability may include the number of proprietary configurations that can be exported or imported based on a standard format, completeness, precision and relevance of tests, time required to migrate large amounts of data, etc.

C. Security

Preserving security becomes more complex in a multi-cloud environment. Trust among the different cloud service providers is essential. It is difficult to handle the heterogeneity of the different security rules established by each provider, making it complex to monitor security policies in composite services. Besides, an additional challenge involves data and identity privacy preservation when several services from different providers collaborate.

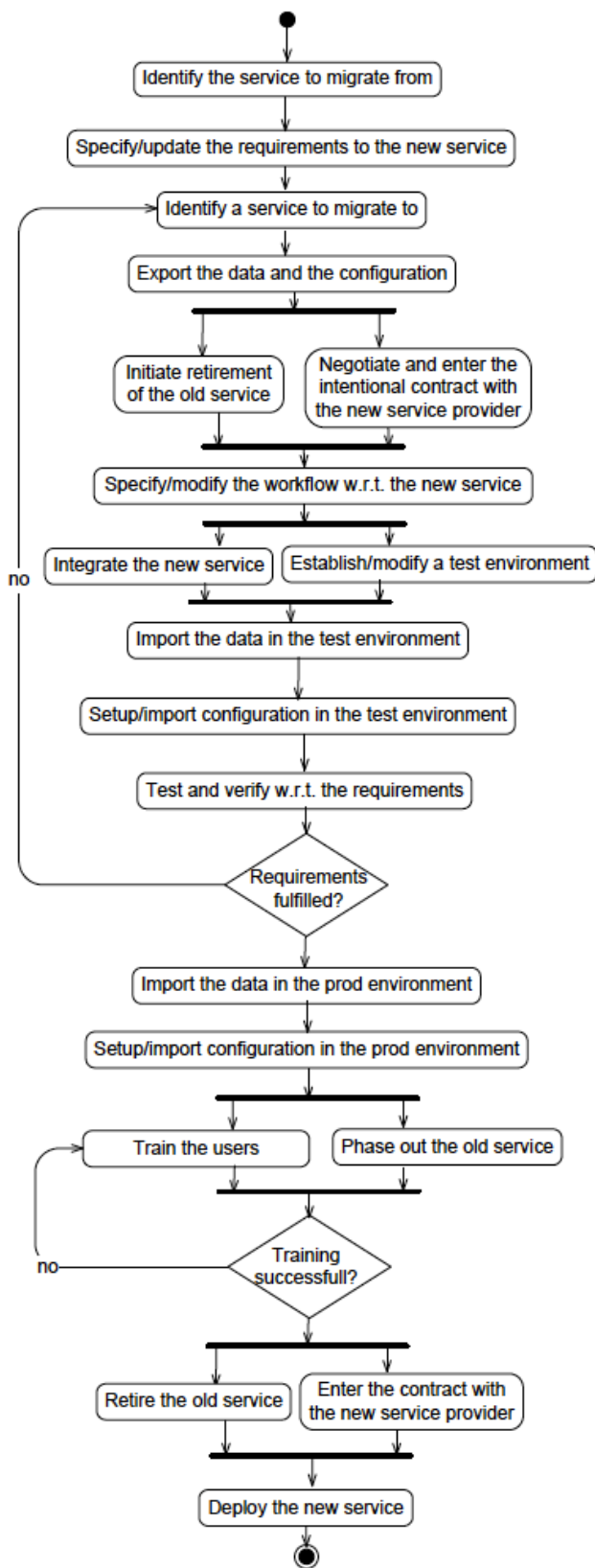


Fig. 4: Description of a generic migration process

In Figure 2, we classify quality aspects related to security in the usual areas: confidentiality, integrity, availability, non-repudiation, accountability and authentication and authorization. In order to preserve data privacy, it is crucial to establish agreements with other providers on the level of privacy of data and identities. Trust in general must be guaranteed by explicit agreements or shared protocols between providers. An alternative solution involves using reliable proxies for communication, but services still need to be able to establish agreements on the fly and secure delegation with these proxies. Finally, it will be important to evaluate services depending on the need to store data in public storage system in order to share this data with other services. In this case, data are exposed to a larger number of threats

V. COST IN MULTI-CLOUD ENVIRONMENTS

Besides risk and quality, we consider another essential dimension: cost. SMI and other previous proposals describe cost-related aspects in cloud computing environments. In a multi-cloud environment, an extra cost appears that may be also considered in the decision-making process: the cost of migration. Migrating from one cloud service to another may involve several economic costs that must be considered at design time. These costs may depend on the personnel involved in the migration process, the cost incurred by keeping the old and the new cloud services running in parallel during the migration process, the cost of the hardware or other resources necessary to perform the migration, or the cost of training the users of the application (note that this cost is also necessary in other situations, but it is usually unavoidable in a migration process).

VI. SPECIFIC RISKS IN MULTI-CLOUD ENVIRONMENTS

In this section, we sketch a list of possible potential risks that may be found in a multi-cloud system. These risks are based on the analysis of the elicited quality aspects that make multi-cloud environments different from clouds provided by a single provider.

1) *Risk of unexpected lack of replacement and consequent vendor lock-in:* a certain cloud service may not fulfill requirements, or requirements may change. In this situation a different service may be needed but it may not be possible to find a new service provided by another vendor which is interoperable with the other services of the system. Two theoretically equivalent services might differ in several relevant aspects. The heterogeneity between different CSPs is usually high as they typically use proprietary interfaces and configurations. Services are also highly integrated with lower-level services offered by the same CSP. Examples of this may be lack of common SLA enforcement systems, use of non-compatible technologies, lack of compatibility in the communications protocol, lack of shared mechanisms to ensure data consistency and quality, the existence of services which are not strictly equivalent and miss some important functionalities, or the lack of services compliant with certain regulations. If this problem appears and the need for migrating from the original service is real, this may even force the migration of other services apart from the service which is not compliant with requirements.

2) *Risk of new security breaches due to the increased complexity of the system and new communications:* data needs to flow from one service to another, hosted by different providers. This creates new points of failure and potential security issues. For instance, this may be caused by the lack of shared security protocols and data integrity mechanisms, lack of forensic mechanisms to be compliant with regulations, the lack of shared authentication systems, etc.

3) *Risk of non-viable migration due to migration costs and complexity:* a developer may not be aware of the cost and complexity of migrating from a certain service chosen to be part of the application to other similar services (see Figure 4). This might become a risk if it is necessary to migrate from that service to another one. As we have discussed, a usual problem in a migration process is the lack of compatible data formats, making it necessary to perform transformations that require time and resources. A related problem might be the lack of information of the new service regarding a certain quality aspect. In this case, uncertainty may also impact a migration process negatively. Note also, that a technical aspect to be considered is whether two services are implemented using the same technology, which might also be a blocking factor for a fast and easy migration. Complexity in the setup migration may also be an important problem. Beyond compatibility in terms of data storage and access, the configuration of a cloud service may also be essential to guarantee the compliance with user requirements. An excessively complex migration of configurations between two services may also result in a time-consuming and expensive migration process. Besides, ease of testing a service and total downtime are two aspects that may largely impact the suitability of a certain migration. Several possible methodologies may be used for developing and support this testing. For instance, modified A/B may be used where only one service is changed and a number of different grades of testing are performed. Finally, depending on the requirements of the application, it might be necessary for the two cloud services, the original one and the replacement, to coexist during a certain period of time, during the testing process of the migration. Complexity to synchronize data between the two services might make the coexistence difficult and using the new service as a hot backup of the first is inefficient.

4) *Risk of costs unpredictability:* by using services from different providers, it may become more and more complex to predict costs.

5) *Risk of lack of provider interest in collaboration:* business agreements are usually required for two CSP to collaborate. For instance, the service delivery model requires customers to register to a service. Because of this, a service in a certain CSP will not allow customers from other CSPs to use it without going through the necessary registration process, unless the right agreements are put in place. Besides, vendors may try to retain customers at any cost to be more competitive. Contracts and other legal issues may be blockers to migrate from one service to an equivalent one. In other words, there is a risk of unfair customer retention and consequent vendor lock-in.

6) *Risk of unavailability of evidences in case of fraudulent actions:* this is a potential risk that may be caused by the lack of forensic tools and global tracking mechanisms.

7) *Risk of lack of negotiation on SLAs:* large organizations using a single supplier can negotiate terms. SMEs or companies using multiple services from multiple vendors are unlikely to have the power or the time to negotiate. This will create an increasingly unstable cost and terms and conditions problem.

Note that a more formal risk analysis might be performed to consider this a final list of risks.

VII. CONCLUSIONS AND FUTURE WORK

In this position paper, we have discussed some essential aspects to establish the necessary baseline for a decision support method aimed at facilitating the selection of cloud services and providers in a multi-cloud environment. In particular, we argue that risk, quality and cost are among the main factors in such a selection process. We believe that a trade-off analysis between risk, cost and quality based on a consolidated view of the three will provide a useful basis for a decision maker in assessing the possible choices through a cost-benefit analysis. For this, we have reported the results of an elicitation of the risk, cost and quality aspects that are specific to multi-cloud environments. We argue that security, interoperability and ease of migration are among the main quality aspects in a multi-cloud environment.

Beyond this initial analysis, we plan to develop a comprehensive study on risk and quality aspects to be considered in a multi-cloud. With this, we aim at creating a decision support tool able to help multi-cloud applications architects to design their systems. This tool will be implemented based on a new methodology that integrates risk, quality and cost dimensions.

ACKNOWLEDGMENT

This work has been conducted as a part of the MODA-Clouds project (Grant Agreement FP7-318484) funded by the European Commission within the 7th Framework Programme.

REFERENCES

- [1] D. Bernstein and D. Vij, Intercloud Security Considerations, Proc. 2nd Intl Conf. Cloud Computing (CloudCom 10), IEEE Press, 2010, pp. 537-544.
- [2] R. Buyya et al., Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility, Proc. 9th IEEE/ACM Intl Symp. Cluster Computing and the Grid (CCGRID 09), IEEE CS, 2009, pp. 599-616.
- [3] Data Protection Act 1998.
<http://www.legislation.gov.uk/ukpga/1998/29/contents>
- [4] European Telecommunications Standards Institute (ETSI).
<http://www.etsi.org>
- [5] Ligaarden, O. S.: A Framework for Analyzing and Monitoring the Impact of Dependencies on Quality. PhD thesis, University of Oslo (2013)
- [6] M.P. Papazoglou and W. van den Heuvel, Blueprinting the Cloud, IEEE Internet Computing, Nov./Dec 2011, pp. 74-79.
- [7] B. Rochwerger et al., ReservoirWhen One Cloud Is Not Enough, Computer, Mar. 2011, pp. 44-51.
- [8] Sarbanes-Oxley Act of 2002 (Pub.L. 107204, 116 Stat. 745, enacted July 30, 2002). <http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/html/PLAW-107publ204.htm>
- [9] Singhal, M.; Chandrasekhar, S.; Tingjian Ge; Sandhu, R.; Krishnan, R.; Gail-Joon Ahn; Bertino, E., "Collaboration in multicloud computing environments: Framework and security issues," Computer, vol.46, no.2, pp.76-84, Feb. 2013
- [10] Cloud Services Measurement Index Consortium: CSMIC Website <http://csmic.org/>. Accessed, March 2013

Towards a Method for Decision Support in Multi-cloud Environments

Aida Omerovic
SINTEF ICT
Norway

Email: aida.omerovic@sintef.no

Victor Muntés-Mulero and Peter Matthews
CA Technologies
CA Labs Europe

Email: {victor.muntes,peter.matthews}@ca.com

Alexander Gunka
BOC Information Systems
Austria

Email: alexander.gunka@boc-eu.com

Abstract—Providers of cloud services as well as the cloud services themselves differ in the business models, functionality, quality of service, cost, value, etc. which makes the choice of a provider and a service difficult. Beyond that the complexity and lack of transparency with respect to cost and quality render the run-time adaptation and replacement of services almost impossible. This position paper presents main results of our recent efforts towards development of a decision support method (DSM) in multi-clouds. The DSM aims at taking into account risk, quality and cost aspects in order to assist a decision maker in choosing providers and services in a multi-cloud environment. We characterize the needs for the DSM in the multi-cloud context and propose an initial version of the process for the DSM. Based on the method proposed and the needs identified, we elaborate to what degree the current state of the art can be leveraged and what further multi-clouds-specific extensions are needed.

Keywords—multi-cloud; decision support; risk assessment; quality prediction; cost prediction; architectural design; trade-off analysis; cloud service selection; cloud provider selection.

I. INTRODUCTION

The rapidly increasing number of cloud services and cloud service providers opens for new opportunities [1] in designing application and enterprise architectures. It also enables new business models and investments [2] [3] [4], new quality levels [5], as well as new capabilities. The services can be orchestrated and their compositions adapted even more dynamically than earlier. Availability of similar services from several providers opens for replaceability between services, or redundancy of services. As a result, the quality may improve and the risk of vendor lock-in will normally be reduced. However, there are also significant challenges [6] involved in realizing collaborations between clouds. One of the major challenges regarding cloud services and their providers is that they differ in the business models, functionality, quality of service, cost, value, etc. Another challenge is complexity and lack of transparency with respect to cost and quality. This makes the choice of a provider and a service difficult and the run-time adaptation and replacement of services almost impossible. When selecting the cloud services and the cloud providers, systematic support for identifying the candidate services and understanding the implications of choosing the different alternatives, is needed.

Decision support [7] for multi-cloud environments imposes several challenges compared to the traditional model-based decision support. Most notably, the dynamics of multi-cloud require light-weight processes and tools, the decision makers

depend on easy-to-understand representations of the impacts of the decisions, the notion of cost is to a lower degree established in the existing approaches supporting the trade-off analysis of enterprise and software architectures, and a merge of the aspects of risk, cost and quality in a consolidated view imposes a new complexity as well as methodological challenges.

The specific objective of this paper is to establish the necessary baseline for a tool-supported decision support method (DSM) aimed at facilitating selection of cloud services and providers in a multi-cloud environment. In particular, we argue that risk, quality and cost are among the main three factors in such a selection process. To that end, we aim at providing a decision support which analyses the impacts of the possible decision alternatives in a multi-cloud environment with respect to those three factors. We believe that a trade-off analysis between risk, cost and quality based on a consolidated view of the three will provide a useful basis for a decision maker in assessing the possible choices through a cost-benefit analysis.

This position paper presents the main results of the recent efforts towards development of a DSM for multi-cloud environments. We characterize the needs for the DSM in the multi-cloud context and propose an initial version of the process for the DSM. Based on the method proposed, we elaborate on the suitability of both the method proposed and the state of the art for analyzing risks as well as for predicting quality and cost in the multi-cloud context.

The paper is organized as follows. Section 2 summarizes the state of the art regarding risk analysis, quality prediction, and cost analysis. Section 3 characterizes the needs for the DSM in the multi-cloud context. Section 4 proposes an initial process for the DSM. Section 5 discusses to what degree the state of the art can be leveraged within the DSM process proposed. Main conclusions are provided in Section 6.

II. STATE OF THE ART

The ISO 31000 standard for risk management comes with no specific techniques, modeling languages or recommended tools for how to conduct risk assessment in practice. However, most established risk management methods [8] [9] [10] [11] follow the ISO 31000 process, and provide such additional support. Common for these approaches is that they are designed to support risk management and risk documentation from the perspective of an organization and its policies. There is lack of support in the state of the art for extracting the risk picture that is relevant for specific external stakeholders, such

as services consumers, and to present this picture in an intuitive and easily understandable way. There is also lack of an approach which combines cloud modeling and risk modeling. There exist many different approaches to service modeling [12] [13] [14] [15], focusing on expressing relevant elements and aspects of services, such as actors and components, roles, activities, interfaces and contracts. However, none of these have a risk-oriented view where stakeholders are represented as risk owners, and where the assets at stake are made explicit.

In a model-based decision making, the decisions are made based on a number of factors. The major ones include functional and non-functional properties, as well as cost and the added value. A trade-off between such factors is the basis for decision making. This trade-off is particularly complex between the non-functional factors, the variable parts of the architecture, and the cost of the selected solutions. The variability, as well as incomplete information or knowledge, are also sources of risk. Since functional requirements normally are less flexible and specified rather early, and since the added value is strongly related to the functional properties, the factors that are tunable and highly interrelated are risk, quality and cost. Therefore, in a model-based decision making, the decisions are based on a trade-off assessment between risk, quality and cost. The risk assessment, in turn, is based on information that is gathered about assets, entities, actors, etc. that are involved in the service event or action in question.

As a basis for the elicitation of the adequate quality characteristics, we may use the software product quality standard ISO/IEC 9126 [5]. The ISO 9126 defines quality as “the totality of features and characteristics of a software product that bear on its ability to satisfy stated and implied needs”. The ISO 9126 standard provides an established specification of decomposed quality notions with their qualitative and quantitative definitions. The standard defines a quality model for external and internal quality, and for quality in use. External quality is the totality of the characteristics of the software product from an external view when the software is executed. Internal quality is the totality of characteristics from an internal view and is used to specify properties of interim products. The characteristics of the internal and external quality model are functionality, reliability, usability, efficiency, maintainability and portability. These are in turn decomposed into a total of 34 sub-characteristics. Quality in use is the user’s view of the quality of the software product when it is used in a specific environment and a specific context of use. The quality in use characteristics are effectiveness, productivity, safety and satisfaction. There is also a further decomposition of all characteristics into the related metrics.

SMI [16] is a standardization effort from the Cloud Services Measurement Index Consortium (CSMIC) consisting of academic and industry organizations. The Service Measurement Index (SMI) uses a series of characteristics and measures to create a common means to compare different services from different suppliers. The characteristics are categorized as Usability, Performance, Agility, Security and Privacy, Financial, Assurance and Usability. Each of these characteristics has a number of measures that can be used to evaluate the risk in using a service. For example in the accountability category one of the measured attributes is Compliance and another is SLA verification both of which can be used to create a risk measure

for the service and the provider. CSMIC is in negotiation with a number of large standardization organizations to develop a joint working group and specification.

According to Fenton and Neil [17], most prediction models use size and complexity metrics to predict defects. Others are based on testing data, the quality of the development process, or take a multivariate approach. The goal/question/metric paradigm [18] [19] is a significant contribution to quality control and can be used for development of quality models and for the design of a measurement plan [20] [21]. To enable explicit risk and quality assessment, we make use of monitoring and measurement. Risk monitoring is a means to facilitate continuous risk assessment by the monitoring of relevant key indicators or metrics. An indicator can be defined as “something that provides a clue to a matter of larger significance or makes perceptible a trend or phenomenon that is not immediately detectable” [22]. To enable explicit risk and quality assessment, we make use of monitoring and measurement.

PREDIQT [23] is a tool supported method for model-based prediction of impacts of architectural design changes on system quality characteristics (performance, scalability, security, etc.). PREDIQT facilitates specification of quality characteristics and their indicators, aggregation of the indicators into functions for overall quality characteristic levels, and dependency analysis. The main objective of a PREDIQT-based analysis is prediction of system quality by identifying different quality aspects, evaluating each of these, and composing the results into an overall quality evaluation. This is useful, for example, for elicitation of quality requirements, evaluation of the quality characteristics of a system, run-time monitoring of quality relevant indicators, as well as verification of the overall quality characteristic fulfillment levels. PREDIQT makes use of models that capture the system design, the system quality notions, as well as the relations between them. An important aim of PREDIQT is to enable the right balance between practical usability of the models and the soundness of the predictions. The method is compatible with the ISO/IEC 9126 software quality standard, and has been successfully applied in real-life industrial settings [24] [25].

CORAS [8] is a tool-supported and model-driven approach to risk analysis that is based on the ISO 31000 risk management standard. Whereas alternative state-of-the-art approaches such as CRAMM [26] and OCTAVE [27] rely on text and tables, CORAS uses diagrams as an important means for communication, evaluation and assessment. Risk modeling is a technique for risk identification and assessment, and the state-of-the-art offers several tree-based and graph-based notations. Fault tree analysis [28] (FTA), event tree analysis [29] (ETA) and attack trees [30] are examples of the former and provide support for reasoning about the sources and consequences of unwanted incidents, as well as their likelihoods. Cause-consequence analysis [31] (CCA), Bayesian network [32] and Markov analysis [33] are examples of graph-based notations. CCA employs diagrams that combine the features of both fault trees and event trees, whereas the latter two serve as mathematical models for probabilistic and statistical calculations, respectively.

Approaches to quality assessment, risk analysis and security management provide support for decision making so as to

ensure a required quality level while managing risks. However, while identifying and suggesting options and solutions, such as security mechanisms, the methods often lack techniques and tools for analyzing the associated cost and the return of investment in the identified solutions. Franqueira et al. [2] address this problem by proposing a method for handling security investment decisions achieved by so-called Real Option thinking. The method is partly based on Real Option Analysis [3] (ROA), which is a decision support technique in the area of capital investment by means of mathematical models to evaluate financial options. The method is supported by a security trade-off tool called SecInvest, which is implemented as a Bayesian network topology and supports decision makers in evaluating investment options and identifying the most suitable and cost-efficient ones. Other approaches to cost estimation in the setting of security investments are Net Present Value (NPV) [4], Return on Security Investment (ROSI) [34], Architecture Trade-Off Analysis Method (ATAM) [35], the Cost Benefit Analysis Method (CBAM) [7] and the Security Solution Design Trade-Off Analysis [36]. These and similar approaches can be understood as methods and techniques to facilitate so-called security economics.

III. CHARACTERIZATION OF NEEDS

As a part of context establishment, we elicited quality aspects and risks which are specific to a multi-cloud environments. The elicitation was based on a comprehensive model of migration process. The model was used as a baseline and a checklist for understanding and decomposing the risk, quality and cost aspects. The exercise resulted in a high-level overview of main risks, as well as a model of decomposed quality characteristics which are specific to multi-clouds. The three overall characteristics identified are: interoperability, intercloud replaceability and security. In addition, cost of migration between multi-clouds was classified into cost of personnel, cost of time with two coexisting services, cost of compensation for uncertainty, and cost of hardware and other resources. Through these models, a common understanding of the main risk, quality and cost aspects in our context, was established. The initial experiences and results of the quality, cost and risk classification indicate that:

- Before eliciting the quality characteristics and risks of a multi-cloud based architecture, the context has to be thoroughly defined. Moreover, the architecture models of the target need to be established. This provides a common understanding of the scope and objectives, as well as the necessary frames for further modeling and decision making. For example, during the context establishment, a process model for migration was used as the foundation for eliciting the aspects and indicators related to quality, cost and risk.
- The decision support models should, once available, be able to take the proposed alternatives for architecture design (measures and treatments considered) and, based on each alternative, provide the resulting risk picture, predicted levels of fulfillment of the relevant quality characteristics, as well as the estimated costs. Thus, risk, quality characteristics and cost should be treated as separate concerns.

- Ideally, in order to accommodate for a cost-benefit analysis, the method should consider added value (or profit) in addition to cost. Minimizing cost and risks and maximizing quality levels is not necessarily a realistic goal. In fact, the benefits may arise from e.g. process improvement through the new architecture, improved or extended functionality, or similar. Thus the trade-offs between quality, risk and cost may vary significantly depending on the utility function and the risk attitude of the decision maker. In addition, the trade-off (or “selection criteria”) should take into account the need for balancing the cost with the added value beyond achieving the quality and risk relevant objectives.
- The method should be tool supported, and the tool should at least provide a diagram editor as well as an easy-to-understand presentation of the impacts of the decision alternatives on quality, risk and cost. The tool should also offer the interfaces needed for acquisition of the data needed for evaluation of the indicators, as well as the interfaces for the needed trace-link information.

IV. METHOD FOR DECISION SUPPORT FOR MULTI-CLOUD ENVIRONMENTS – A PRELIMINARY SPECIFICATION

The DSM for multi-cloud applications is a model-driven method consisting of three main artifacts: a process, a language and a tool. This section provides the initial specification of the DSM process and the actors involved. The DSM process consists of three overall phases, and each phase is decomposed into a set of sub-phases. The DSM process is undergone while developing, verifying and applying the comprehensive decision support models which include the aspects of architecture, risk, quality and cost. We assume the following four types of actors involved in the DSM process:

- Analyst: the analyst is an expert in the DSM and has the responsibility for leading and facilitating a DSM-based analysis. That is, the analyst coordinates the overall actors, collects the input for developing the decision support models, interacts with the overall actors during the model development and usage, makes sure that the necessary steps have been conducted within the resources allocated, and validates that the models have the needed quality and contents.
- Decision maker: the decision maker defines the scope and the objective of a DSM-based analysis. He/she will provide the instructions as to what parts of the architecture should be encompassed in the models, the expected validity of the models, the scope and kinds of the perspective changes/revisions of the architecture, etc. The decision maker will also be the main user of the decision models once they have been developed. He will therefore specify the decision alternatives in the decision models, and use the resulting impact estimates with respect to risk, cost and quality as an aid in the decision making. This actor is aware of the business model and strategy of the company. Hence, a decision maker may be a business expert as well, capable of making decisions based on his knowledge

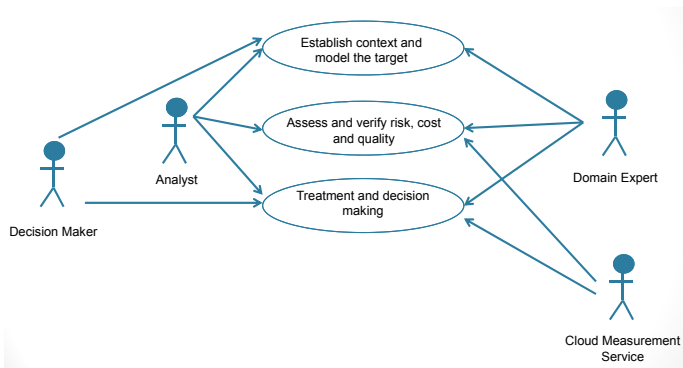


Fig. 1. The top level three phases and the actors involved in the DSM process

of the project budgets, allowable risks and the business processes being supported by the applications. Larger organizations may distinguish between a business expert who builds the requirements specification and a decision maker who selects services based on the specification. For simplicity, these two roles are in our case represented by the decision maker who has all the knowledge sufficient to take decisions.

- Domain expert: normally, a group of domain experts will be involved in a DSM-based analysis in relation to the development, validation and revision of the decision models. The domain experts will contribute by providing the thorough input regarding the current architecture, quality levels, dependencies and processes. The analyst will actively interact with the domain experts during all the three phases of the DSM process.
- Cloud measurement service: this is a (partially) automatized service for retrieval of the empirical data needed for estimating the parameters of the decision models. We assume that the parameters are estimated either based on the feeds from the cloud measurement service or based on expert judgments. A parameter may be estimated or measured either directly, or through estimation of a measurable indicator which then is aggregated and mapped to the decision model through a function. The dynamics of the indicators and the parameters as well as their relevance and uncertainty will be among the factors for determining whether the data acquisition should be automatic (e.g. real-time retrieval based on a monitoring environment) or manual, and how frequent it should be.

Figure 1 shows the overall three phases of the DSM process, as well as the actors involved. In the first phase, the context of the analysis is established. As a part of this, the scope is defined, the relevant risk, cost and quality notions are defined, and the architecture is modeled. In addition, the expected validity as well as perspective business models and architecture alternatives should be anticipated in order to cover the needed scope and level of detail in the target models. During the second phase, the decision models covering the risk, quality and cost aspects are instantiated with respect to target. As a part of this, the dependencies are modeled and the parameters (with the related indicators) are estimated.

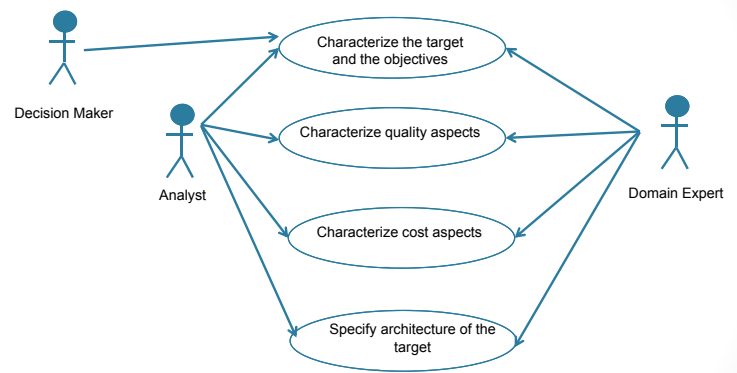


Fig. 2. Establish context and model the target phase decomposed

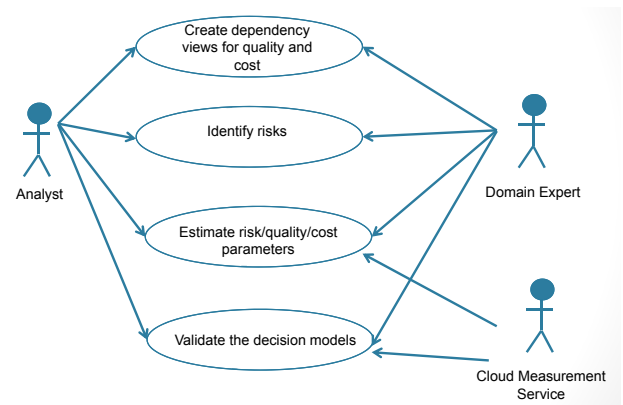


Fig. 3. Assess and verify risk, cost and quality phase decomposed

In addition, the models are validated through various kinds of triangulation, mainly based on the empirical input, logs, domain expert judgments, experience factories, etc. In the last phase, the decision models are applied by first specifying the decision alternatives, applying the alternatives on the models, and finally obtaining the resulting impact of the respective decisions on quality, risk and cost. The result is a consolidated view of the quality, risk and cost picture, provided each decision alternative.

Figure 2 shows the stages of the “establish context and model the target” - phase. First, the target and the objectives are characterized. Based on the initial input, the stakeholders involved deduce a high level characterization of the target architecture, its scope and the objectives of the DSM-based analysis, by formulating the system boundaries, system context (including the usage profile), system lifetime and the extent (nature and rate) of design changes expected. In the second stage, the quality aspects are characterized by specifying which quality characteristics are relevant for the target, and thereafter decomposing them down to indicators. A quantitative and a qualitative definition should be provided for all elements. Thirdly, a corresponding decomposition should be done for the cost aspects. In the last stage, the architecture is modeled with the detail level and within the frames specified during the characterization stage.

Figure 3 shows the stages of the “assess and verify risk, cost and quality” - phase. Firstly, the dependency views for

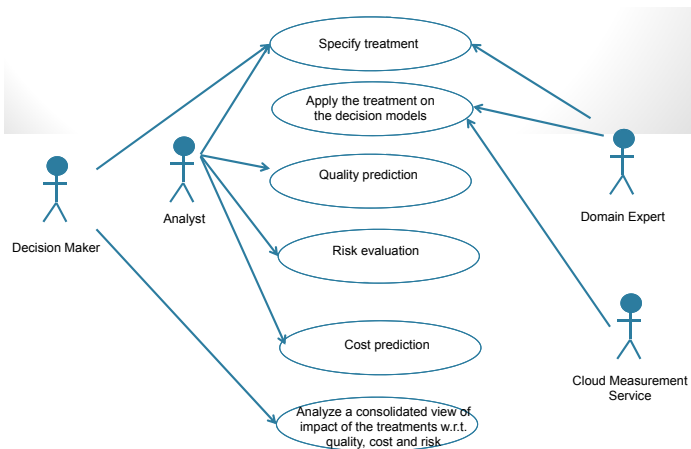


Fig. 4. Treatment and decision making phase decomposed

respectively quality and cost are developed. Secondly, assets and risks are identified in separate decision models (“threat diagrams”). The three types of the decision models (i.e. quality dependency views, cost dependency views and threat diagrams) are then annotated by the parameter values through evaluation of indicators or direct expert judgments on the prior parameters. Finally, triangulation is performed in order to validate the decision models. The models are approved once an acceptable level of uncertainty has been reached.

Figure 4 shows the stages of the “treatment and decision making” - phase. First, the respective decision alternatives are specified separately. Then, each alternative is applied on the decision models. The models and the respective calculus is used to propagate the impacts of each decision alternative on risk, quality and cost. Finally, a consolidated view of the impacts of the decision alternatives is presented to the decision maker.

Figure 5 shows an activity diagram with the entire DSM process, including the feedback loops. The right hand side of the figure indicates the phases presented in Figure 1. The activities are equivalent to the ones presented in relation to Figure 2, Figure 3 and Figure 4.

V. DISCUSSION

This section elaborates to what degree the existing PREDIQT and CORAS methods for for quality prediction and risk analysis, respectively, can serve as a baseline for our DSM in multi-clouds. The objective is to leverage the state of the art decision support, while extending it and adjusting to the special needs of the multi-clouds. Thus, the established methods, languages and tools can be reused with the well known properties and resources, while the efforts can be concentrated on the multi-cloud-specific extensions.

PREDIQT is a method (process, language, and tool support) for model-based prediction of system quality. The PREDIQT method produces and applies a multi-layer model structure, called prediction models, which represent system relevant quality concepts (through “Quality Model”), architectural design (through “Design Model”), and the dependencies between architectural design and quality (through “Dependency

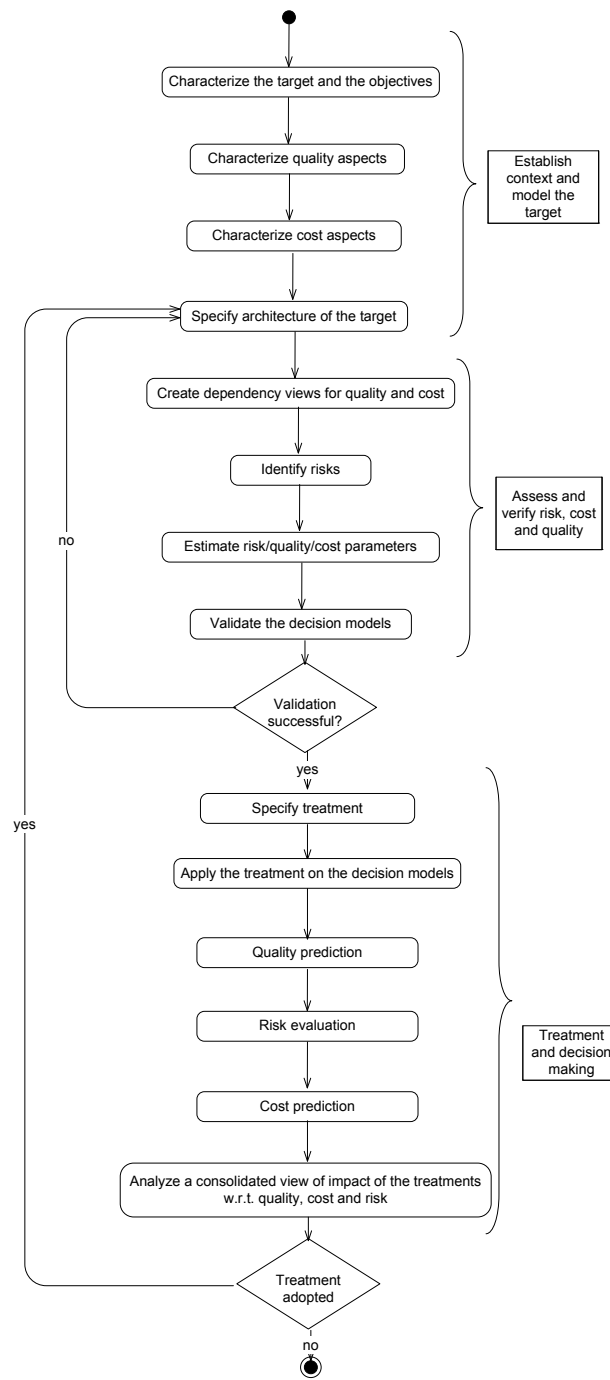


Fig. 5. The DSM process diagram with feedback loops

Views”). The Design Model diagrams are used to specify the architectural design of the target system and the changes whose effects on quality are to be predicted. The Quality Model diagrams are used to formalize the quality notions and define their interpretations. The values and the dependencies modeled through the Dependency Views (DVs) are based on the definitions provided by the Quality Model. The DVs express the interplay between the system architectural design and the quality characteristics. Once a change is specified on the Design Model diagrams, the affected parts of the DVs are

identified, and the effects of the change on the quality values are automatically propagated at the appropriate parts of the DV.

CORAS is a method (process, language, and tool support) for conducting model-based security risk analysis. CORAS provides a customized language for threat and risk modeling, and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis. The Unified Modeling Language (UML) is typically used to model the target of the analysis. For documenting intermediate results, and for presenting the overall conclusions we use special CORAS diagrams which are inspired by UML. The CORAS tool supports documenting, maintaining and reporting analysis results through risk modeling.

The DSM process is based on an attempt to merge the processes of CORAS and PREDIQT for a consolidated analysis of risk, quality and cost. Most of the stages of the DSM process can be found in CORAS and PREDIQT. The actors/stakeholders defined in the DSM are fully compliant with the ones defined by CORAS and PREDIQT. The types of the decision models proposed in the DSM are heavily based on the modeling notations, languages and tools of PREDIQT and CORAS, respectively. The approach to modeling of quality and cost aspects based on the DVs is a part of the PREDIQT method, while a language for risk modeling is provided by CORAS. The respective approaches to modeling in PREDIQT and CORAS are based on graphical modeling languages with defined propagation models. Both modeling approaches are developed with special focus on comprehensibility and expressiveness. In that manner, the models are accommodated for fulfilling real-life needs in terms of covering the representations needed while being rather intuitive so that non-experts should be able to relate to them in an industrial setting. The characterization of quality proposed in DSM is by PREDIQT addressed through the so called Quality Model. Both the Quality Model and the intended quality characterization in DSM are similar to the elicitation we have performed, which is briefly presented in Section 3.

The DSM process is to a high degree a superset of the processes of PREDIQT and CORAS. Moreover, the modeling approaches of PREDIQT and CORAS cover the concerns of quality and risk, as well as partially the concern of cost. Furthermore, the existing tools of CORAS and PREDIQT may be useful in the DSM context. Provided this baseline, we believe that utilization of the CORAS and PREDIQT methods including the processes, the languages and the tools, is worth a further evaluation in the DSM context. In particular, this means that case studies in multi-cloud environments should be performed in order to evaluate the feasibility of DSM, as well as the suitability of the relevant parts of PREDIQT and CORAS in a multi-cloud context.

VI. CONCLUSION AND FUTURE WORK

This position paper aims at establishing the necessary baseline for a DSM. The intended purpose of the DSM is to facilitate the selection of cloud services and providers in a multi-cloud environment. In particular, we argue that risk, quality and cost are among the main factors in such a selection

process. We believe that a trade-off analysis between risk, cost and quality based on a consolidated view of the three will provide a useful basis for a decision maker in assessing the possible choices through a cost-benefit analysis.

Decision support for multi-cloud environments imposes however several challenges compared to the traditional model-based decision support. Most notably, the dynamics of multi-cloud require light-weight processes and tools, the decision makers depend on easy-to-understand representations of the impacts of the decisions, the notion of cost is to a lower degree established in the trade-off analysis of enterprise and software architectures, and a merge of the aspects of risk, cost and quality in a consolidated view imposes a new complexity as well as methodological challenges.

This paper presents the main results of our recent efforts towards the development of a DSM for multi-cloud environments. We characterize the needs for the DSM in the multi-cloud context and propose an initial version of the process for the DSM. Based on the experiences from CORAS and PREDIQT based analyses, and relying on the existing process descriptions and modeling approaches from CORAS and PREDIQT, we propose a comprehensive process for a DSM-based analysis, and present the roles of the actors/stakeholders involved. The DSM process consolidates the steps necessary towards development, verification and application of the decision support models. Based on the method proposed, we elaborate on the suitability of both the method proposed and the state of the art for analyzing risks as well as for predicting quality and cost in the multi-cloud context. We argue that many aspects of CORAS and PREDIQT, including the approaches to modeling (the modeling languages), the processes, and the respective tool support, should be well suited in the DSM context, i.e. in an analysis which merges the aspects of risk, quality and cost. However, in order to evaluate the feasibility of both the proposed DSM in general as well as the CORAS and PREDIQT methods in particular, in the multi-cloud context, realistic case studies should be performed and the proposed method adapted based on the experiences obtained.

Hence, the next steps in the development of decision support for multi-clouds should include case studies, evaluation and development of approaches to modeling (the modeling languages) for a consolidated model-based risk analysis, quality prediction and cost analysis. Moreover, the method should offer an easy-to-understand visualization of the impacts of the decision alternatives on quality, cost and risk. We also aim at refining the method and the tool requirements for DSM, as well as providing a prototype tool which will facilitate a DSM-based analysis.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 318484 (MODAClouds).

REFERENCES

- [1] R. Buyya, "Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009.

- [2] V. N. L. Franqueira, S. H. Houmb, and M. Daneva, "Using Real Option Thinking to Improve Decision Making in Security Investment," in *5th International Symposium on Information Security, LNCS 6426*. Springer, 2010, pp. 619–638.
- [3] M. Amram and N. Kulatilaka, *Real Options: Managing Strategic Investment in an Uncertain World*. Harvard Business School Press, Cambridge, Massachusetts, 1999.
- [4] M. Daneva, "Applying Real Options Thinking to Information Security in Networked Organizations. CTIT Report TR-CTIT-06-11," University of Twente, Tech. Rep., 2006.
- [5] *ISO/IEC 9126 – Software engineering – Product quality – Part 1-4*, International Organization for Standardization/International Electrotechnical Commission, 2001-2004.
- [6] M. Singhal, S. Chandrasekhar, G. Tingjian, R. Sandhu, R. Krishnan, A. Gail-Joon, and E. Bertino, "Collaboration in Multicloud computing Environments: Framework and Security Issues," *Computer*, vol. 46, no. 2, pp. 76–84, 2013.
- [7] R. Kazman, J. Asundi, and M. Klein, "Making Architecture Design Decisions: An Economic Approach. Technical report CMU/SEI-2002-TR-035," Carnegie Mellon, Tech. Rep., 2002.
- [8] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis - The CORAS Approach*. Springer, 2011.
- [9] Siemens, "CRAMM - The Total Information Security Toolkit," March 2004, accessed: January 30, 2013. [Online]. Available: <http://www.cramm.com>
- [10] C. J. Alberts and A. J. Dorofee, "OCTAVE Criteria. Technical Report CMU/SEI-2001-TR-016," CERT, Tech. Rep., 2001.
- [11] T. Peltier, *Information Security Risk Analysis, 3rd edn*. Auerbach Publications, 2010.
- [12] R. Chinnici, J. J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation," June 2007, accessed: January, 2013. [Online]. Available: <http://www.w3.org/TR/wsdl20>
- [13] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema. W3C Recommendation," August 2007, accessed: January, 2013. [Online]. Available: <http://www.w3.org/TR/sawSDL>
- [14] "Service Oriented Architecture Modeling Language (SoAML) Specification, Version 1.0," Object Management Group, Tech. Rep., 2012.
- [15] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "Reference Model for Service Oriented Architecture 1.0.," OASIS, Tech. Rep., 2006.
- [16] Cloud Services Measurement Index Consortium, "CSMIC," accessed: January 2013. [Online]. Available: <http://csmic.org>
- [17] N. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Transactions on Software Engineering*, vol. 25, pp. 675–689, 1999.
- [18] V. R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm, Technical Report TR-92-96," University of Maryland, Tech. Rep., 1992.
- [19] V. Basili, G. Caldiera, and H. Rombach, *The Goal Question Metric Approach*. Encyclopedia of Software Engineering, 1994.
- [20] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., 1998.
- [21] C. Ebert, R. Dumke, M. Bundschuh, A. Schmietendorf, and R. Dumke, *Best Practices in Software Measurement*. Springer Verlag, 2004.
- [22] A. Hammond, A. Adriaanse, E. Rodenburg, D. Bryant, and R. Woodward, "Environmental Indicators: A Systematic Approach to Measuring and Reporting on Environmental Policy Performance in the Context of Sustainable Development," World Resources Institute, Tech. Rep., 1995.
- [23] A. Omerovic, *PREDIQT: A Method for Model-based Prediction of Impacts of Architectural Design Changes on System Quality. PhD thesis*. University of Oslo, 2012.
- [24] A. Omerovic, A. Andresen, H. Grindheim, P. Myrseth, A. Refsdal, K. Stølen, and J. Ølnes, "A Feasibility Study in Model-based Prediction of Changes on System Quality. Technical report A13339," SINTEF ICT, Tech. Rep., 2010.
- [25] A. Omerovic, B. Solhaug, and K. Stølen, "Assessing Practical Usefulness and Performance of the PREDIQT Method: An industrial case study," *Information and Software Technology*, vol. 54, no. 12, pp. 1377–1395, 2012.
- [26] B. Barber and J. Davey, "The Use of the CCTA Risk Analysis and Management Methodology CRAMM in Health Information Systems," in *7th International Congress on Medical Informatics*, 1992, pp. 1589–1593.
- [27] C. J. Alberts and J. Davey, "OCTAVE Criteria Version 2.0. Technical report CMU/SEI-2001-TR-016," Carnegie Mellon University, Tech. Rep., 2004.
- [28] "IEC 61025 Fault Tree Analysis (FTA)," International Electrotechnical Commission, Tech. Rep., 1997.
- [29] "IEC 60300-3-9 Dependability Management - Part 3: Application guide - Section 9: Risk analysis of technological systems - Event Tree Analysis (ETA)," International Electrotechnical Commission, Tech. Rep., 1995.
- [30] B. Schneier, "Attack Trees: Modeling security threats," *Dr. Dobbs Journal*, vol. 24, no. 12, pp. 21–29, 1999.
- [31] D. S. Nielsen, "The Cause/Consequence Diagram Method as Basis for Quantitative Accident Analysis. Technical report RISO-M-1374," Danish Atomic Energy Commission, Tech. Rep., 1971.
- [32] I. Ben-Gal, *Bayesian Networks*. In F. Ruggeri, R. S. Kenett, F. W. Faltin (eds.): *Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, 2007.
- [33] R. A. Howard, *Dynamic Probabilistic Systems. Volume I: Markov Models*. John Wiley & Sons, 1971.
- [34] W. Sonnenreich, J. Albanese, and B. Stout, "Return on Security Investment (ROSI)-A Practical Quantitative Model," *Journal of Research and Practice in Information Technology*, vol. 38, no. 1, pp. 45–56, 2006.
- [35] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for Architecture Evaluation. Technical report CMU/SEI-2000-TR-004," Carnegie Mellon, Tech. Rep., 2000.
- [36] S. H. Houmb, G. Georg, R. France, J. Bieman, and J. Jürjens, "Cost-benefit Trade-off Analysis Using BBN for Aspect-oriented Risk-driven Development," in *10th International Conference on Engineering of Complex Computer Systems*. IEEE Computer Society, 2005, pp. 195–204.