CLOUD COMPUTING 2011

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

ISBN: 978-1-61208-153-3

September 25-30, 2011
Rome, Italy

CLOUD COMPUTING 2011 Editors

Massimo Villari, University of Messina, Italy

Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland

Yong Woo Lee, University of Seoul, Korea

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

CLOUD COMPUTING 2011

Foreword

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.

The Second International Conference on Cloud Computing, GRIDs, and Virtualization [CLOUD COMPUTING 2011], held between September 25 and 30, 2011, in Rome, Italy, 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 be fixed, especially on security, privacy, and inter- and intra-clouds protocols.

We welcomed technical papers presenting research and practical results, position papers addressing the pros and cons of specific proposals, such as those being discussed in the standard fora or in industry consortia, survey papers addressing the key problems and solutions on any of the above topics short papers on work in progress, and panel proposals.

We take here the opportunity to warmly thank all the members of the CLOUD COMPUTING 2011 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 2011. 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 2011 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 2011 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the area of cloud computing.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the charm of Rome, Italy.

CLOUD COMPUTING 2011 Chairs:

Arden Agopyan, IBM Central & Eastern Europe, Russia, Middle East & Africa (CEE & MEA), Turkey
Javier Diaz, Indiana University, USA
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Wolfgang Gentzsch, Senior HPC Consultant, Germany
Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA

CLOUD COMPUTING 2011

Committee

CLOUD COMPUTING Advisory Chairs

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany
Daniel S. Katz, University of Chicago & Argonne National Laboratory, USA
Yong Woo Lee, University of Seoul, Korea

CLOUD COMPUTING 2011 Industry/Research Chairs

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

CLOUD COMPUTING 2011 Research Institutes Chairs

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

COULD COMPUTING 2011 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 2011 Technical Program Committee

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 or 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 is 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

A Workflow Engine for Computing Clouds

Daniel Franz, Jie Tao, Holger Marten, and Achim Streit
Steinbuch Center for Computing
Karlsruhe Institute of Technology, Germany
daniel2712@gmx.de, {jie.tao, holger.marten, achim.streit}@kit.edu

***Abstract*—This work developed a workflow engine that enables the execution of workflows on existing Cloud platforms. The workflow engine automatically delivers the computation of each individual task to the selected Cloud and transfers the input/output data across different platforms. Additionally, it predicts the execution time and payment of the tasks, helping users select the best Cloud services with respect to the performance vs. cost tradeoff.**

***Keywords*-Cloud Computing, Workflow Management System, Grid Computing**

I. INTRODUCTION

Since Amazon published its Elastic Compute Cloud (EC2) [1] and Simple Storage Service (S3) [2] in 2008, Cloud Computing became a hot topic in both industrial and academic areas. There exist different definitions of Cloud Computing, including our earlier contribution [3]. Recently, the National Institute of Standards and Technology (NIST) provides a specific definition: Cloud computing is a model for enabling 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 [4].

Cloud computing distinguishes itself from other computing paradigms in the following aspects:

- Utility computing model: Users obtain and employ computing platforms in computing Clouds as easily as they access a traditional public utility (such as electricity, water, natural gas, or telephone network).
- On-demand service provisioning: Computing Clouds provide resources and services for users on demand. Users can customize and personalize their computing environments later on, for example, software installation, network configuration, as users usually own administrative privileges.
- QoS guaranteed offer: The computing environments provided by computing Clouds can guarantee QoS for users, e.g., hardware performance. The computing Cloud renders QoS in general by processing Service Level Agreement (SLA) with users.

As a result of these advantages, Cloud Computing is gaining more and more customers. Currently established Cloud infrastructures mainly deliver three kinds of services: Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service. IaaS targets on an on-demand provision of the computational resources. The commercial computing Cloud Amazon EC2 and its non-commercial implementation Eucalyptus [5] are well-known examples of IaaS-featured Cloud platforms. SaaS allows the consumers to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface [4]. An example of SaaS is Web-based email. PaaS targets on an entire platform including the hardware and the application development environment. Google App Engine [6] and Microsoft Azure [7] are examples of PaaS-featured Clouds.

The goal of this work is to combine different Clouds to run a user-defined service workflow. A workflow is a methodology that splits the computation of a complex problem into several tasks. A well-known scenario is to run scientific experiments on the Grid [8], where an entire computation is partitioned and distributed over several computing nodes with a result of being able to process large data sets. This scenario can also occur on the Cloud when scientific applications move to them. Furthermore, there are other scenarios on the Cloud, where users require the workflow support. For example, users may compose the services provided by different Clouds for an overall goal.

We developed an execution engine for workflow management on Clouds. In difference to Grid workflow implementations that target on a unified interface [9], a Cloud workflow system has to cope with different interfaces and features of individual Clouds. In order to enable the combination of single workflow tasks running on various Clouds, we implemented a Cloud abstraction and designed mechanisms for inter-Cloud data transfer. We also established a prediction model to estimate the execution time and cost of the individual tasks on different Cloud nodes, therefore helping users achieve maximum performance at lowest payment.

The remainder of the paper is organized as following. Section II describes the related work. Section III analyzes the requirement on a Cloud workflow framework and presents the designed software architecture. Section IV gives the details of an initial prototypical implementation, followed by the evaluation results in Section V. The paper concludes in Section VI with a brief summary and several future directions.

 ISBN: 978-1-61208-153-3

II. Related Work

The concept of resource sharing in Cloud Computing is similar to Grid Computing. Cloud Computing allows on-demand resource creation and easy access to resources, while Grid Computing developed standards and provides various utilities. A detailed comparison of these two computing paradigms can be found in [10]. One utility implemented on the Grid is the workflow management system. Production Grids, such as WLCG [11], TeraGrid [12], and EGEE [13], commonly support the execution of scientific workflows on the underlying resources. There are also various implementations of workflow engines on the Grid. Examples are ASKALON [14], Unicore [15], Kepler [16], GridAnt [17], Pegasus [18], and GridFlow [19]. An overview of these workflow systems is presented in [20].

The research work on workflow management systems on the Cloud has been started. A well-known project is the Cloudbus Toolkit [21] that defines a complete architecture for creating market-oriented Clouds. A workflow engine is also mentioned in the designed architecture and described in detail in [22]. The authors analyzed the requirement and changes needed to be incorporated when moving scientific workflows to Clouds. They also described the visions and inherent difficulties when a workflow involves various Cloud services. The work presented in this paper aims at a prototypical implementation of a workflow engine that executes a workflow composed of different Cloud services, because such a tool is currently still not available. The goal is to simply provide a new functionality rather than to investigate a comprehensive solution.

III. Architecture Design

Grid Computing has been investigated for more than a dozen of years and established standards. Cloud Computing, in contrast, is a novel technology and has not been standardized. The specific feature of each Cloud brings additional challenges to implementing a workflow engine on Clouds.

A. Design Challenges

Grid workflows may be executed in several resource centers but the involved resources are contained in a single Grid infrastructure and hence can be accessed with the same interface. Cloud workflows, however, run usually on different Clouds.

Figure 1 shows a sample scenario of running workflows on Clouds. While some tasks may be executed on the same Cloud, e.g., Cloud C1, some others may run on different Cloud platforms. The data are transferred from one Cloud to another in order to deliver the output of one task to other tasks. Unfortunately, different Clouds use also different data format. Furthermore, existing Clouds have their own access interfaces. A standard, called Open Cloud Computing Interface (OCCI) [23], has been proposed but no implementation is currently available. To link the services of different

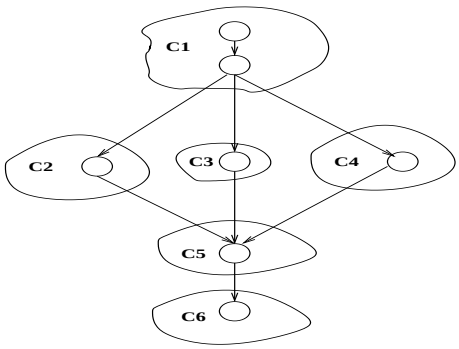


Figure 1. A sample execution scenario of Cloud workflows.

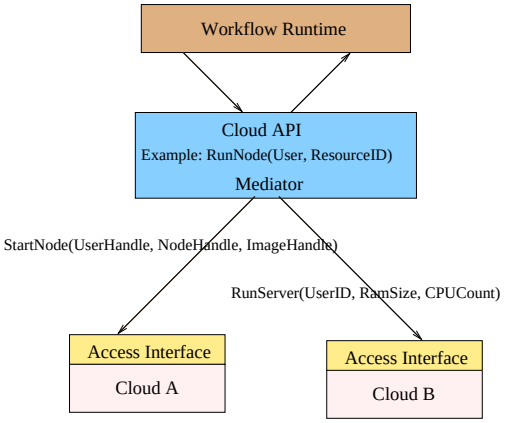


Figure 2. Software architecture of the workflow engine.

Clouds, an abstraction layer is required for providing an identical view with the data and interfaces of the target Cloud infrastructures.

Additionally, the service price varies across Cloud providers. Cloud users usually expect an optimal performance vs. cost tradeoff: i.e., acquiring the best service with the lowest payment. While increasing Cloud infrastructures are emerging, there may be several choices to run a workflow task. A prediction model, which is capable of estimating the performance and cost of an execution on a specific Cloud, can help users select the best Cloud for their tasks.

Based on the aforementioned observations, we designed a software architecture for the proposed Cloud workflow engine and defined a performance-cost model. The following two subsections give some details.

B. Software Architecture

Figure 2 demonstrates the software architecture of the proposed workflow engine for Cloud Computing. An important component in the architecture is the Cloud abstraction layer, shown in the middle of the figure. The task of this layer is to implement a unified API for accessing different Clouds. The runtime environment of the workflow engine uses this

 ISBN: 978-1-61208-153-3

API to run the tasks in a workflow.

The abstraction layer defines common functions for Cloud activities. It also contains a mediator that translates the functions in the unified API to concrete calls to the underlying Cloud platforms. For example, the function RunNode() is provided for running a virtual machine instance on any IaaS-featured Cloud. During the runtime the mediator replaces the function by a Cloud specific one, in this example, either StartNode for Cloud A or RunServer for Cloud B. It also maps the function parameters in the functions of the unified API to the functions of the APIs of individual Clouds. Furthermore, the mediator handles the authentication/security issues.

C. Prediction Model

Cloud users not only take care of the execution performance but pay more attention to the payment for using resources on the Clouds. As an initial design, we bring the two most important metrics, application execution time and the cost, into the prediction model. Workflows in this work are defined as: A workflow is comprised of several tasks, each is combined with an application/software that is either executed on an IaaS-Cloud or hosted as a Web service on a SaaS/PaaS-Cloud.

The execution time of a workflow (EoW in short) can be calculated with the following mathematical form:

$$EoW = EoT_1 + DT_1 + EoT_2 + DT_2 + ... + EoT_n$$

where EoT_i is the execution time of task i and DT_i is the time for transferring data from T_i to T_{i+1}. Note that we ignore the time to start a service on the Cloud as well as data transfers from and back to the customer environment.

The execution time of a single task depends on the features of the host machine on which the task is running. Roughly, it can be presented with:

$$EoT = f(S_{comp}, F_{cpu}, S_{mem}, S_{I/O})$$

where the parameters are size of the computation, frequency of CPU, size of memory and cache, and size of input/output data. For parallel applications, an additional parameter, the communication speed, has to be considered.

The price of a service on a Cloud is usually determined by the node type and the location of the resource. Each Cloud provider maintains a price table, where concrete payment (in US$ per hour) is depicted. Based on this table, we calculate the cost of a workflow task with:

$$CoT = f(EoT, \$/h)$$

The cost of executing a workflow is then calculated with:

$$CoW = CoT_1 + CoT_2 + ... + CoT_n$$

The functions for computing the execution time of a task can be designed differently with a tradeoff between complexity and accuracy. We implemented a simple model, which is detailed in the following section.

IV. Prototypical Implementation

Our initial implementation of a Cloud workflow management system focused on the following components:

- Cloud abstraction
- Runtime execution environment
- Prediction model

A. Cloud Abstraction

To run a workflow on diverse Clouds, an abstraction layer is required for the purpose of hiding the different access interface each Cloud presents to the users. We use jClouds [24] as the base of this work. jClouds provides a framework for transferring programs and their data to an IaaS-Cloud and then starting an instance to execute the program on the Cloud. The current release of jCloud can connect several IaaS-Clouds including Amazon EC2.

jClouds defines an API for accessing the underlying IaaS platforms. For SaaS/PaaS-featured Clouds, however, there exists currently no implementation for an abstraction layer. Our main task in extending jClouds is to develop an S+P abstraction that interacts with SaaS-featured and PaaS-featured Clouds.

The S+P abstraction contains two kinds of functions, GET and POST, for transferring data and service requests. Their input and output are defined in XML documents. This is identical to all Clouds. Each Cloud, however, requires specific input and output formats as well as different parameters for service requests. Our solution is to use XSL Transformation (XSLT) [25] to map the input and output of the service functions to the required data format and service parameters.

XSLT is a part of the Extensible Stylesheet Language (XSL) family and often adopted to transform XML documents. An XSLT file describes templates for matching the source document. In the transformation process, XSLT uses XPath, an interface for accessing XML, to navigate through the source document and thereby to extract information or to combine/reorganize the separate information elements. For this work an XSLT document is introduced for some data formats, like SOAP. For others, such as binary and JSON (JavaScript Object Notation), a data transformation is not needed.

The process of invoking a SaaS or PaaS service with the developed S+P abstract contains the following steps:

- Processing the input data of the service request.
- Constructing a URL for the service. Information about Cookies, SOAP actions and other parameters, is contained in the head of the protocol (HTTP), while the content of the protocol defines the request.
- A service request is sent to the aforementioned URL, together with the data.
- The results of the service are downloaded as raw data. For the data formats like SOAP, where the results are

 ISBN: 978-1-61208-153-3

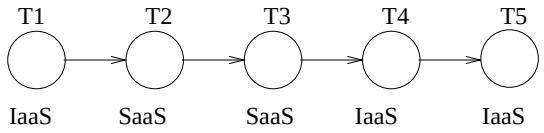


Figure 3. A simple Cloud workflow.

coded, an XSLT document is defined to extract the useful information.

B. Workflow Execution

In order to allow an easier understanding of the tasks for a Cloud workflow execution engine, we take a simple workflow as an example. Figure 3 demonstrates the sample workflow consisting of five tasks, T1 to T5, which are combined through a respective data flow. A task can be a program or an available Web service on a SaaS or PaaS Cloud. For the former, the program is executed on an IaaS Cloud, while for the latter the Cloud provides resources for running the software. The workflow and its tasks are defined by the user in an XML file. .

The workflow execution engine is responsible for running each task on the selected Cloud, transferring the result of one task to its successor, and downloading the final results to the user. The first job is performed within a single Cloud and contains the following steps, which are all covered by the Cloud abstraction described above:

- Transferring data (Program or service parameters) to the target Cloud.
- Executing the program on an IaaS Cloud or invoking the Web service on the SaaS or PaaS Cloud. In the case of IaaS, a virtual machine instance has to be started and some scripts are executed for configuration and program installation.
- Extracting the results out of the Cloud.

Another task of the workflow runtime engine is to deliver the output of one task to the next task as input. This involves an inter-Cloud communication. We implemented mechanisms for the following data transfer:

- IaaS to SaaS/PaaS: We use SSH to transfer data from the IaaS node to the local host and then use HTTP to deliver the data further to the SaaS/PaaS request;
- SaaS/PaaS to SaaS/PaaS: Data are extracted from the HTTP stream, stored temporally on the host, and then applied to the next HTTP request;
- SaaS/PaaS to IaaS: Locally storing the data, which are again extracted from an HTTP stream, and then transferring them to the IaaS node via SSH;
- IaaS to IaaS: We transfer the data directly from one IaaS node to the other that is potentially located on a different Cloud. This is an optimization for removing the overhead caused by an intermediate storage.

Finally, the result of the entire execution is downloaded to the user or stored on the last Cloud.

C. Performance & Cost Prediction

The proposed prediction model, as described in the previous section, involves several hardware parameters that can be only acquired at the runtime by accessing the Cloud resources. For the prototypical implementation, we developed a simple model without using the runtime resource information of the underlying infrastructures.

Our model is based on the execution history of similar tasks, which are tasks executing the same program. The execution history is stored in a user database, which contains the following main data structures:

- node_class: describes a computing node with node ID, node name, Cloud name, payment cycle, and startup time.
- execution: describes an execution of a task on a node with several attributes including program name, node_class, size of I/O, and execution time.
- node_price: gives the per-cycle-price of the computing nodes.
- node_location: gives the country and continent the node is located.

For each task in a new user-defined workflow, the potential execution time is calculated for all registered Clouds and their associated computing nodes. The payment is then calculated according to the price published by the Cloud providers. The first five {Cloud, node} pairs with the best performance vs. cost tradeoff are shown to the users to help them select the optimal target platforms.

We use the following algorithm to predict the execution time of a new task presented with $t_{(p,s)}$, where the first attribute is the program to be executed and s is the size of the input data.

First, the average execution time of the program on a node n_s is calculated with

$$t_{(p,n_s)} = \frac{\sum_{i=1}^{n} t_{i(p,n_s,s_i)}}{n}$$

where $t_{i(p,n_s,s_i)}$ is the time measured with the recorded ith execution of program p on node n_s with a data size of s_i. Here, $t_{(p,n_s)}$ is associated with the average data size $s_{(p,n_s)}$, which is calculated in a similar way. The execution time of the new task $t_{(p,s)}$ can then be estimated with

$$t_{(p,s)} = \frac{s}{s_{(p,n_s)}} \cdot t_{(p,n_s)} \cdot W_{data}$$

We introduce a weight variable W_{data} to represent the influence degree of the input size on the execution time.

Table I
EXPERIMENTAL RESULTS WITH THE 3D RENDER WORKFLOW (85 CAMERA POSITIONS).

Task	Node	Execution time			Performance vs. Cost
		Measured	Predicted	Difference (%)	
3dscenetopictures	m1.small	145	138	-4.8	12.4
	c1.medium	56	52	-7.1	19.03
	m1.large	48	42	-12.5	17,97
picturetovideo	m1.small	59	48	-18.6	5.01
	c1.medium	47	37	-21.2	15.97
	m1.large	44	36	-18.2	14.96

Table II
EXPERIMENTAL RESULTS WITH THE WORKFLOW OF SYNCHRONIZING A FOUR MINUTES VIDEO.

Task	Node	Execution time			Performance vs. Cost
		Measured	Predicted	Difference (%)	
videototext	m1.small	665	688	3.4	168.04
	c1.medium	341	355	4.1	116.3
	m1.large	257	271	5.4	87.2
translatejatoen		45	40	-11.1	0
texttospeech	m1.small	26	22	-15.4	1.87
	c1.medium	22	20	-9.1	7.47
	m1.large	19	17	-10.5	6.46
jointovideo	m1.small	89	104	16.8	7.6
	c1.medium	87	94	8.04	29.6
	m1.large	97	75	-12.4	33.02

V. EVALUATION RESULTS

To evaluate the developed framework, several workflows were tested. In this section, we present the results with two examples. The first workflow processes 3D scenes with a result of creating a video. The second workflow performs film synchronization whereby to translate the spoken text from Japanese to English.

The first workflow contains two main tasks, *3dscenetopictures* (the raytracer) and *picturetovideo*. The raytracer acquires a scene file and a camera file as input and splits the scene into single pictures based on the position defined in the camera file. The single pictures are then processed by the second task to produce a continuous video. We apply the Tachyon [26] raytracer for the first task, which needs an MPI cluster on an IaaS Cloud because the software is parallelized with MPI. To combine the pictures to a video, the program FFmpeg [27] is applied. We run this task on a single IaaS node. Hence, the first workflow involves only IaaSs.

The second workflow is comprised of four components: the language identifier (task *videototext*), a translator (task *translatejatoen*), the text synthesizer (task *texttospeech*), and the task *jointovideo*. The language identifier acquires a video file as input and outputs its text in Japanese. The output is then delivered to the language translator, where an English text is produced. In the following, the text synthesizer converts the text to speech, which is combined with the video via the last task of the workflow. We apply the language identifier Julius [28] to process the audio that is extracted from the video by FFmpeg. In order to speed up the process, an audio is first partitioned and the partitions are then processed in parallel. Hence, an MPI cluster is required for this task. For language translation, the translation service of Google is applied. In order to model a SaaS/PaaS to SaaS/PaaS data transfer and to verify our Cloud abstraction, the Japanese text is first translated to German and then to English. The task *texttospeech* is implemented using the speech synthesizer eSpeak [29]. Finally, the aforementioned FFmpeg program combines the audio with the video.

For the experiments we requested an account on EC2. The test results are shown in Table I and Table II for each workflow. The tables show the execution time of tasks of a single workflow on different nodes of EC2. In the case of Google, the Web service is executed on a Google machine, which cannot be specified by the user.

The execution time of a task is presented with the measured time and the predicted one, where the former was acquired at runtime and the latter was calculated using the developed prediction model. It can be seen that the accuracy of our model varies between the tasks, where the value with the second workflow is relative better. For the 3D render, the model underestimates the execution time in most cases, while an alternating behavior can be seen with the second workflow. Altogether, we achieved the best case with a difference of 3.4% between the real execution time and the predicted one, while the worse case shows a value of -21.2%. The difference is caused by the fact that the time for executing a program can vary significantly from one execution to the other, even though the executions are performed successively. This indicates that a more accurate model is required for a better prediction, which shall be our future work.

 ISBN: 978-1-61208-153-3

The values in the last column of the tables are calculated by multiplying the real execution time by the payment. It is expected that both the execution time and the payment are low. Hence, we use the values in the last column to represent the performance vs. cost tradeoff, where a lower value indicates a better behavior. Observing Table I it can be seen that the nodes m1.small have a better behavior. This may be associated with the concrete tasks, which do not demand a high computation capacity. With larger programs, e.g., the task *videototext* in the second workflow, a node with higher capacity, m1.large in this case, behaves better. However, the best choice is to use the free services provided by some Clouds, such as the translation service on Google.

VI. CONCLUSIONS

This paper described a workflow engine, which are designed and implemented for Cloud Computing. To enable the execution of a service workflow we developed a Cloud abstraction that mediates between different Cloud platforms. We implemented a runtime engine to execute the single tasks in the workflow and transfer data among them. Additionally, a prediction model was designed to estimate the execution time of the tasks on different Cloud nodes. Currently we implemented a simple model that will be improved in the next step of this work. Furthermore, we plan to develop a search engine that automatically detects Cloud services for a user-specified task. A graphical interface is also planned to allow the user to define the workflows in a more intuitive way. In addition, the workflow engine will be extended to handle the exception/errors of the Cloud services.

REFERENCES

[1] "Amazon Elastic Compute Cloud," [Online], June 2011, http://aws.amazon.com/ec2/.

[2] "Simple Storage Service," [Online], June 2011, http://aws.amazon.com/s3/.

[3] L. Wang, M. Kunze, and J. Tao, "Performance evaluation of virtual machine-based Grid workflow system," *Concurrency and Computation: Practice & Experience*, vol. 20, pp. 1759–1771, October 2008.

[4] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," [Online], January 2011, http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

[5] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," in *Proceedings of Cloud Computing and Its Applications*, October 2008. [Online]. Available: http://eucalyptus.cs.ucsb.edu/wiki/Presentations

[6] "Google App Engine," [Online], June 2011, http://code.google.com/appengine/.

[7] "Windows Azure Platform," [Online], June 2011, http://www.microsoft.com/windowsazure/.

[8] B. Asvija, K. V. Shamjith, R. Sridharan, and S. Chattopadhyay, "Provisioning the MM5 Meteorological Model as Grid Scientific Workflow," in *Proceedings of the International Conference on Intelligent Networking and Collaborative Systems*, 2010, pp. 310–314.

[9] G. Fox and D. Gannon, "Special Issue: Workflow in Grid Systems," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1009–1019, 2006.

[10] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, 2008, pp. 1–10.

[11] WLCG, "Worldwide LHC Computing Grid," [Online], June 2001, http://lcg.web.cern.ch/lcg/.

[12] P. H. Beckman, "Building the TeraGrid," *Philosophical transactions - Royal Society. Mathematical, physical and engineering sciences*, vol. 363, no. 1833, pp. 1715–1728, 2005.

[13] EGEE, "Enabling Grids for E-sciencE," [Online], June 2011, project homepage: http://www.eu-egee.org/.

[14] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. S. Jr, and H. L. Truong, "ASKALON: a tool set for cluster and Grid computing," *Concurrency and Computation: Practice & Experience*, vol. 17, pp. 143–169, February 2005.

[15] M. Riedel, D. Mallmann, and A. Streit, "Enhancing Scientific Workflows with Secure Shell Functionality in UNICORE Grids," in *Proceedings of the IEEE International Conference on e-Science and Grid Computing*. IEEE Computer Society Press, December 2005, pp. 132–139.

[16] D. Barseghian, I. Altintas, M. B. Jones, D. Crawl, N. Potter, J. Gallagher, P. Cornillon, M. Schildhauer, E. T. Borer, E. W. Seabloom, and P. R. Hosseini, "Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis," *Ecological Informatics*, vol. 5, pp. 42–50, 2010.

[17] G. von Laszewski, K. Amin, M. Hategan, N. J. Z. S. Hampton, and A. Rossi, "GridAnt: A Client-Controllable Grid Workflow System," in *37th Hawaii International Conference on System Science*. IEEE CS Press, January 2004.

[18] S., D. Karastoyanova, and E. Deelman, "Bridging the Gap between Business and Scientific Workflows: Humans in the Loop of Scientific Workflows," in *IEEE International Conference on eScience*, 2010, pp. 206–213.

[19] J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd, "GridFlow:Workflow Management for Grid Computing," in *Proceedings of the International Symposium on Cluster Computing and the Grid*, May 2003, pp. 198–205.

[20] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," *Journal of Grid Computing*, vol. 3, no. 3-4, pp. 171–200, September 2005.

[21] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus Toolkit for Market-Oriented Cloud Computing," in *Proceeding of the 1st International Conference on Cloud Computing*, December 2009, pp. 978–642.

[22] S. Pandey, D. Karunamoorthy, and R. Buyya, *Cloud Computing: Principles and Paradigms*. Wiley Press, February 2011, ch. 12, pp. 321–344.

[23] "Open Cloud Computing Interface," [Online], June 2001, http://occi-wg.org/.

[24] "jclouds," [Online], June 2001, http://www.jclouds.org/.

[25] M. Kay, *XSLT 2.0 Programmer's Reference*. Wrox, 3 edition, August 2004.

[26] J. Stone, "An Efficient Library for Parallel Ray Tracing and Animation," In Intel Supercomputer Users Group Proceedings, Tech. Rep., 1995.

[27] "FFmpeg," [Online], June 2001, http://www.ffmpeg.org/.

[28] "Open-Source Large Vocabulary CSR Engine Julius," [Online], June 2001, http://julius.sourceforge.jp/en_index.php.

[29] "eSpeak text to speech," [Online], June 2001, http://espeak.sourceforge.net/.

 ISBN: 978-1-61208-153-3

FCM: an Architecture for Integrating IaaS Cloud Systems

Attila Csaba Marosi, Gabor Kecskemeti, Attila Kertesz, Peter Kacsuk
MTA SZTAKI
Computer and Automation Research Institute
of the Hungarian Academy of Sciences
H-1528 Budapest, P.O.Box 63, Hungary
Email: {atisu, kecskemeti, keratt, kacsuk}@sztaki.hu

***Abstract*—Cloud Computing builds on the latest achievements of diverse research areas, such as Grid Computing, Service-oriented computing, business processes and virtualization. In this paper, we reveal open research issues by envisaging a federated cloud that aggregates capabilities of various IaaS cloud providers. We propose a Federated Cloud Management architecture that acts as an entry point to cloud federations and incorporates the concepts of meta-brokering, cloud brokering and on-demand service deployment. The meta-brokering component provides transparent service execution for the users by allowing the system to interconnect the various cloud broker solutions available in the system. Cloud brokers manage the number and the location of the utilized virtual machines for the received service requests. In order to fast track the virtual machine instantiation, our architecture uses the automatic service deployment component that is capable of optimizing service delivery by encapsulating services as virtual appliances in order to allow their decomposition and replication among the various IaaS cloud infrastructures. Our solution is able to cope with highly dynamic service executions by federating heterogeneous cloud infrastructures in a transparent and autonomous manner.**

***Keywords*—cloud federation; cloud brokering; IaaS; virtual appliance.**

I. INTRODUCTION

Highly dynamic service environments [1] require a novel infrastructure that can handle the on demand deployment and decommission of service instances. Cloud Computing [2] offers simple and cost effective outsourcing in dynamic service environments and allows the construction of service-based applications extensible with the latest achievements of diverse research areas, such as Grid Computing, Service-oriented computing, business processes and virtualization. Virtual appliances (VA) encapsulate metadata (e.g., network requirements) with a complete software system (e.g., operating system, software libraries and applications) prepared for execution in virtual machines (VM). Infrastructure as a Service (IaaS) cloud systems provide access to remote computing infrastructures by allowing their users to instantiate virtual appliances on their virtualized resources as virtual machines.

Nowadays, several public and private IaaS systems co-exist and to accomplish dynamic service environments users frequently envisage a federated cloud that aggregates capabilities of various IaaS cloud providers. These IaaS systems are either offered by public service providers (like Amazon [3] or RackSpace [4]) or by smaller scale privately managed infrastructures. We propose an autonomic resource management solution that serves as an entry point to this cloud federation by providing transparent service execution for users. The following challenges are of great importance for such a mediator solution: varying load of user requests, enabling virtualized management of applications, establishing interoperability, minimizing Cloud usage costs and enhancing provider selection.

This paper proposes a layered architecture that incorporates the concepts of meta-brokering, cloud brokers and automated, on-demand service deployment. The meta-brokering component allows the system to interconnect the various cloud brokers available in the system. The cloud broker component is responsible for managing the virtual machine instances of the particular virtual appliances hosted on a specific infrastructure as a service provider. Our architecture organizes the virtual appliance distribution with the automatic service deployment component that can decompose virtual appliances to smaller parts. With the help of the minimal manageable virtual appliances the Virtual Machine Handler rebuilds these decomposed parts in the IaaS system chosen by the meta-broker. As a result, the cloud broker component uses the VM Handler to maintain the number of virtual machines according to the demand.

Related works have identified several shortcomings in the current cloud infrastructures [5]: e.g., federated clouds will face the issue of scalability and self-

 ISBN: 978-1-61208-153-3

management similarly to Grid systems, or users of the cloud systems should be in control of their computing costs. We propose an architecture that aims at both of these problems by allowing users to utilize meta-brokering between public and private cloud systems as a result lowering their operation costs. Our architecture also handles the issue of scalability by offering the cloud brokers that manage the virtual machines according to the actual demands of the user applications.

This paper is organized as follows: first, we introduce the related research results in Section II. Then, we discuss an advanced use case in Section III that involves our proposed architecture and discusses its advantages in contrast to previous research results. Next, we detail the operational roles of the brokering components in our architecture in Section III-A and Section III-B. Afterwards, in Section IV, we discuss an optimization approach to rebuild virtual appliances within the virtual machine that is used to execute them. Finally, we conclude our research in Section V.

II. Related Work

Matthias Schmidt et al. [6] investigate different strategies for distributing virtual machine images within a data center: unicast, multicast, binary tree distribution and peer-to-peer distribution based on BitTorrent. They found the multicast method the most efficient, but in order to be able to distribute images over network boundaries ("cross-cloud") they choose BitTorrent. They also propose to use layered virtual machine images for virtual appliances consisting of three layers: user, vendor and base. By using the layers and a copy-on-write method they were able to avoid the retransmission of images already present at the destination and thus decrease instantiation time and network utilization. The authors only investigated distribution methods within the boundaries of a single data center, going beyond that remained future work.

There are several related works focusing on providing dynamic pool of resources. Paul Marshall et al. [7] describe an approach for developing an "elastic site" model where batch schedulers, storage and web services can utilize such resources. They introduce different basic policies for allocating resources, that can be "on-demand" meaning resources are allocated when a service call or task arrives, "steady stream" assumes steady utilization, thus leaves some elastic resources continuously running, regardless of the (temporary) shortage of tasks, or "bursts" for fluctuating load. They concentrate on dynamically increasing and decreasing the number of resources, but rely on third party logic for balancing load among the allocated resources. Constantino Vázquez et al. [8] are building complex grid infrastructures on top of IaaS cloud systems, that allow them to adjust the number of grid resources dynamically. They focus on the capability of using resources from different cloud providers and on the capability of providing resources for different grid middleware, but meta-scheduling between the utilized infrastructures and developing a model, that considers the different cloud provider characteristics is not addressed.

In 2009, Amazon Web Services launched Amazon CloudWatch [9], that is a supplementary service for Amazon EC2 instances that provides monitoring services for running virtual machine instances. It allows to gather information about the different characteristics (traffic shape, load, disk utilization, etc.) of resources, and based on that users and services are able to dynamically start or release instances to match demand as utilization goes over or below predefined thresholds. The main shortcoming is that this solution is tied to a specific IaaS cloud system and introduces a monetary overhead, since the service charges a fixed hourly rate for each monitored instance.

Mohsen Amini et al. [10] are focusing on so called marketing-oriented scheduling policies, that can provision extra resources when the local cluster resources are not sufficient to meet the user requirements. Former scheduling policies used in grids are not working effectively in cloud environments, mainly because Infrastructure as a Service providers are charging users in a pay-as-you-go manner in an hourly basis for computational resources. To find the trade-off between to buy acquired additional resources from IaaS and reuse existing local infrastructure resources he proposes two scheduling policies (cost and time optimization scheduling policies) for mixed (commercial and non-commercial) resource environments. Basically two different approaches were identified on provisioning commercial resources. The first approach is offered by the IaaS providers at resource provisioning level (user/application constraints are neglected: deadline, budget, etc.), the other approach deploys resources focusing at user level (time and/or cost minimization, estimating the workload in advance, etc.).

III. Federated Cloud Management Architecture

Figure 1 shows the Federated Cloud Management (FCM) architecture and its connections to the corresponding components that together represent an interoperable solution for establishing a federated cloud environment. The FCM targets the problem area outlined in the Introduction, and provides solutions for most of the listed open issues. In the following, we exemplify

 ISBN: 978-1-61208-153-3

the interaction of the main components of this solution through a low level use case.

In this scenario we restrict our solution to support standard stateless web services described with WSDL [11]. Using this solution, users are able to execute services deployed on cloud infrastructures transparently, in an automated way. Virtual appliances for all services should be stored in a generic repository called FCM Repository, from that they are automatically replicated to the native repositories of the different Infrastructure as a Service cloud providers.

When a user sends a service call to the system, he/she submits a request to the "*Generic Meta-Broker Service*" (GMBS) specifying the requested service with a WSDL, the operation to be called, and its possible input parameters. The GMBS checks if the service has an uploaded VA in the generic repository, then it selects a suitable CloudBroker for further submission. The matchmaking is based on static data gathered from the "*FCM Repository*" (e.g., service operations, WSDL), and on dynamic information of special deployment metrics gathered by the CloudBrokers. Currently we use the average VA deployment time and the average service execution time for each VA. VA deployment time assumes that the native repository already has the requested VA, thus includes only the service provision time on a specific IaaS cloud. The role of GMBS is to manage autonomously the interconnected cloud infrastructures with the help of the CloudBrokers by forming a federation.

Each "*CloudBroker*" has an own queue for storing the incoming service calls (called Q_1 and Q_2 in Figure 1), and manages one virtual machine queue for each VA ($VA_x \rightarrow VMQ_x$). Virtual machine queues represent the resources that currently can serve a virtual appliance specific service call. The main goal of the CloudBroker is to manage the virtual machine queues according to their respective service demand. The default virtual machine scheduling is based on the currently available requests in the queue, their historical execution times, and the number (n, m, o, p) of running VMs. The secondary task of the CloudBroker involves the dynamic creation and destruction of the various VMQs.

Virtual Machine Handler ("*VM Handler*") components are assigned to each virtual machine queue. These components process the virtual machine creation and destruction requests placed in the queue. The requests are translated and forwarded to the corresponding IaaS system ($Cloud_a$). This component is a cloud infrastructure-specific one, that uses the public interface of the managed infrastructure.

Independently from the virtual machine scheduling process the CloudBroker also handles the queue of the incoming service calls. As a result, these calls are dispatched to the available VMs created in the previously discussed manner.

In order to optimize service executions in highly dynamic service environments, our architecture organizes the virtual appliance distribution as a background process with the automatic service deployment component that can decompose virtual appliances to smaller parts. With the help of the minimal manageable virtual appliances (MMVA – further discussed in Section IV) the Virtual Machine Handler is able to rebuild these decomposed parts in the IaaS system on demand, that results in faster VA deployment and in a reduced storage requirement in the native repositories.

In the following, subsections we detail how resource management is carried out in this architecture. At the top-level, a meta-broker is used to select from the available cloud providers based on performance metrics, while at the bottom-level, IaaS-specific CloudBrokers are used to schedule VA instantiation and deliver the service calls to the clouds.

A. Top-level Brokering in FCM

As we already mentioned in the scenario discussed in the previous section, brokering takes place at two levels in the FCM architecture: the service call is first submitted to the Generic Meta-Broker Service (GMBS – that is a revised and extended version of the Grid Meta-Broker Service described in [12]), where a top-level decision is made to that cloud infrastructure the call should be forwarded. Then the service call is placed in the queue of the selected CloudBroker, where the bottom-level brokering is carried out to select the VM that performs the actual service execution. This bottom-level brokering and the detailed introduction of the architecture of the CloudBroker is discussed later in Section III-B.

Now, let us turn our attention to the role of GMBS. An overview of its architecture is shown in Figure 2. This meta-brokering service has five major components. The Meta-Broker Core is responsible for managing the interaction with the other components and handling user interactions.

The MatchMaker component performs the scheduling of the calls by selecting a suitable broker. This decision making is based on aggregated static and dynamic data stored by the Information Collector (IC) component in a local database. The Information System (IS) Agent is implemented as a listener service of GMBS, and it is responsible for regularly updating static information from the FCM Repository on service availability, and aggregated dynamic information collected from the Cloud-

 ISBN: 978-1-61208-153-3

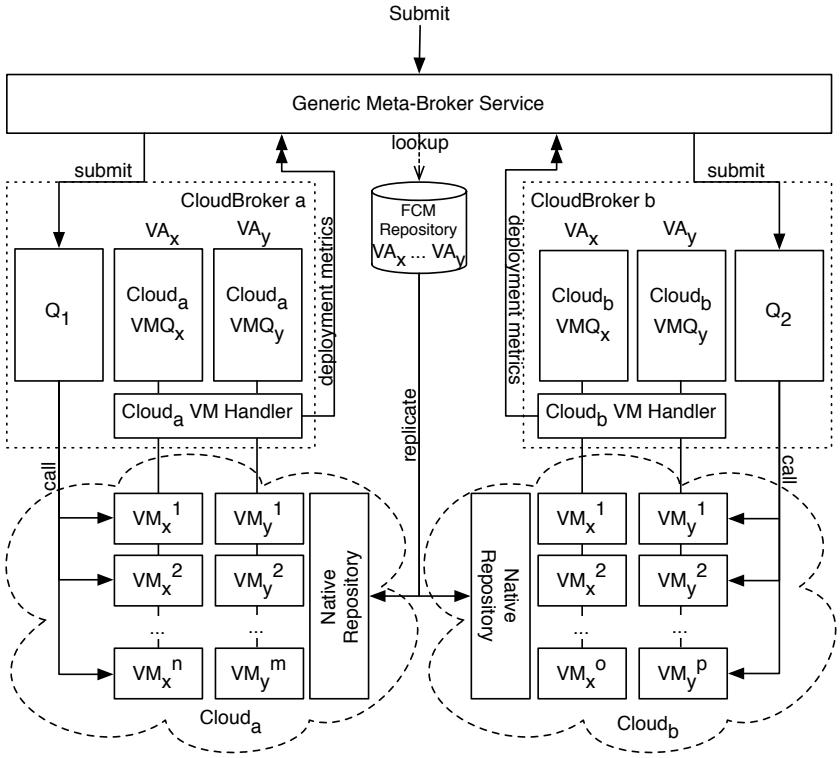


Fig. 1. The Federated Cloud Management architecture

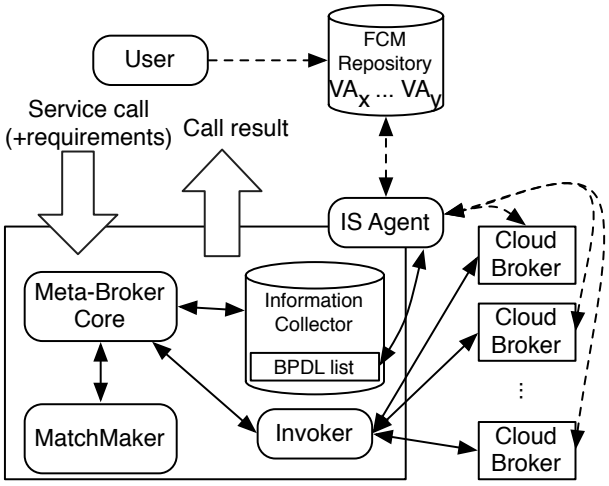


Fig. 2. The architecture of the Generic Meta-Broker Service

Brokers including average VA deployment and service execution times. The Invoker component forwards the service call to the selected CloudBroker and receives the service response.

Each CloudBroker is described by an XML-based Broker Property Description Language (BPDL) document containing basic broker properties (e.g., name), and the gathered aggregated dynamic properties. The scheduling-related attributes are typically stored in the PerformanceMetrics field of BPDL. More information on this document format can be read in [12]. Namely, the following data are stored in the BPDL of each CloudBroker:

- Estimated availability time for a specific virtual appliance in a native repository – collected from the FCM Repository;
- average VA deployment time and average service execution time for each VA – queried from the CloudBroker;

The scheduling process first filters the CloudBrokers by checking VA availability in the native cloud repository, then a rank is calculated for each broker based on the collected static and dynamic data. Finally, the CloudBroker with the highest rank is selected for forwarding the service request.

B. CloudBroker

The CloudBroker handles and dispatches service calls to resources and performs resource management within a single IaaS system, it is an extended version of the

system described in [13].

The architecture of the CloudBroker is shown in Figure 1. Its first task is to dynamically create or destroy virtual machines (VM_x^i) and VM queues (VMQ_x) for the different used virtual appliances. To do that, first, the VA has to be replicated to the native repository of the IaaS system from the FCM Repository (an alternative method is discussed in Section IV). Alongside the appliance, the FCM Repository also stores additional static requirements about its future instances, like its minimum resource demands (disk, CPU and memory), that are needed by the CloudBroker. This data is not replicated to the native repository, rather the FCM Repository is queried.

A VM queue stores references to resources capable of handling a specific service call, thus instances of a specific VA. New resource requests are new entries inserted into the queue of the appropriate VA, while resource destruction requests are modification of entries representing an already running resource. The entries are managed by the VM Handler, that is a cloud fabric specific component designed to interact with the public interface of a single IaaS system. It simply translates and forwards requests to the public interface of the IaaS system ($Cloud_a$). Each VA contains a monitoring component deployed, that allows the CloudBroker to monitor the basic status (CPU, disk and memory usage) of the running resources along the average deployment time for each VA and average service execution times. These data can be queried by the IS Agent of the GMBS.

The service call queue (Q_1 and Q_2) stores incoming service requests and, for each request, reference to a VA in the FCM Repository. There is a single service call queue in each CloudBroker, while there are many VM queues. If the native repository does not contain the requested VA it is replicated first. Dynamic requirements for the VA may be specified with the service call:

- Additional resources (CPU, memory and disk);
- an UUID, that allows to identify service calls originating from the same entity.

The UUID will allow to meet SLA constraints later, e.g., to enforce a total cost limit on public clouds for service calls of the entity, or to be in compliance with deadlines. If any dynamic requirements are present the CloudBroker treats the VA as a new VA type, thus creating a new VM queue and starts a VM. The service calls may now be dispatched to the appropriate VMs. Most IaaS systems offer predefined classes of resources (CPU, memory and disk capacity) not adjustable by the user, in this case the CloudBroker will select the resource class that has at least the requested resources available. This may lead to allocating excess resources in some cases (e.g., the resource class has twice the memory requested to meet the CPU number requirement).

The CloudBroker also performs the scheduling of service call requests to VA's and the life-cycle management of resources. Scheduling decision is made based on the monitoring information gathered from the resources. If the service request cannot be scheduled to any resource the CloudBroker may decide to start a new VM capable of serving the request. The decision is based on the following:

- The number of running VM's available to handle the service call;
- the number of waiting service calls for the VA in the service call queue;
- the average execution time of service calls;
- the average deployment time of VA's;
- and SLA constraints (e.g., total budget, deadline);

VM decommission is also based on the above, but the CloudBroker takes into account the billing period of the IaaS system, shutdown is performed only shortly before the end of the period with regard to the average decommission time for the system.

IV. Virtual Appliance Delivery Optimization

IaaS systems require virtual appliances to be stored in their native repositories. Only those virtual appliances, that were previously stored in these repositories, can be used to instantiate virtual machines. Our architecture allows users to upload their virtual appliances to the FCM Repository that behaves as an active repository and handles the distribution of the appliances to the native repositories according to [14]. As an active repository, the FCM repository identifies the common parts of the appliances and decomposes them into smaller packages that allow appliance delivery and rebuilding from multiple repositories.

Central virtual appliance storage would require the VM Handler to first download the entire appliance from the FCM repository to a native one, then instantiate the appliance with the IaaS system. To avoid the first transfer, but keep the convenience for the users of our architecture, we have investigated options to rebuild virtual appliances in already running virtual machines. We have identified two distinct approaches for rebuilding: (i) native appliance reuse, (ii) minimal manageable virtual appliances. The first approach utilizes already available virtual appliances in the native repositories and extends them towards the required virtual appliance. In this article, we do not aim at this approach because it requires the investigation of the publicly available

 ISBN: 978-1-61208-153-3

appliances in order to find the appliance most suitable for extension.

The second approach proposes the minimal manageable virtual appliance that we define as basic appliance with the following three properties:

- Offers content management interfaces to add, configure and remove new appliance parts.
- Offers monitoring interfaces to analyze the current state of its instances (e.g., provide access to their CPU load, free disk space and network usage).
- Optimally sized: only those files present in the appliance that are required to offer their extensibility with the previously mentioned interfaces.

As a result, our architecture only needs to replicate the MMVAs to every native repository. If the FCM repository identifies high demands for specific virtual appliance parts, then the active repository functionality automatically replicates the appliance to those IaaS systems where most requests were originated from.

Our VM Handler is prepared to control virtual appliance rebuilding using minimal manageable virtual appliances. Consequently, the VM Handler applies a new strategy when it receives a virtual appliance instantiation request for a specific appliance that is not available in the native repository. This strategy starts with the instantiation of the MMVA. Next, the Handler waits until the virtual machine of the MMVA has started up. Then, it requests the content management interfaces to add the parts of the specific appliance that were identified as unique during the decomposition of the MMVA and the specific appliance. As a result, the specific appliance is rebuilt and ready to serve the scheduled service requests in the virtual machine instantiated for the MMVA.

V. Conclusion and Future Works

In this paper, we proposed a Federated Cloud Management solution that acts as an entry point to cloud federations. Its architecture incorporates the concepts of meta-brokering, cloud brokering and on-demand service deployment – their interaction is exemplified through a low-level use case. The meta-brokering component provides transparent service execution for the users by allowing the system to interconnect the various cloud broker solutions managed by aggregating capabilities of these IaaS cloud providers. We have shown how CloudBrokers manage the number and the location of the utilized virtual machines for the various service requests they receive. In order to fast track the virtual machine instantiation, our architecture uses the automatic service deployment component that is capable of optimizing its delivery by decomposing and replicating it among the various IaaS cloud infrastructures. Regarding future works, we plan to investigate various scenarios that arise during handling federated cloud infrastructures using the FCM architecture (e.g., the interactions and interoperation of public and private IaaS systems).

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube) and 261556 (EDGI).

References

[1] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engg.*, vol. 15, pp. 313–341, December 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1459074.1459084

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, June 2009.

[3] Amazon Web Services LLC, "Amazon elastic compute cloud," http://aws.amazon.com/ec2/, 2009.

[4] Rackspace Cloud, http://www.rackspace.com/cloud/.

[5] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 50–55, December 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[6] M. Schmidt, N. Fallenbeck, M. Smith, and B. Freisleben, "Efficient distribution of virtual machines for cloud computing," in *Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, ser. PDP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 567–574. [Online]. Available: http://dx.doi.org/10.1109/PDP.2010.39

[7] P. Marshall, K. Keahey, and T. Freeman, "Elastic site: Using clouds to elastically extend site resources," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGRID '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 43–52. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2010.80

[8] C. Vázquez, E. Huedo, R. S. Montero, and I. M. Llorente, "On the use of clouds for grid resource provisioning," *Future Gener. Comput. Syst.*, vol. 27, pp. 600–605, May 2011. [Online]. Available: http://dx.doi.org/10.1016/j.future.2010.10.003

[9] Amazon CloudWatch, http://aws.amazon.com/cloudwatch/.

[10] M. A. Salehi and R. Buyya, "Adapting market-oriented scheduling policies for cloud computing."

[11] The World Wide Web Consortium, http://www.w3.org/TR/wsdl.

[12] A. Kertész and P. Kacsuk, "Gmbs: A new middleware service for making grids interoperable," *Future Gener. Comput. Syst.*, vol. 26, pp. 542–553, April 2010. [Online]. Available: http://dx.doi.org/10.1016/j.future.2009.10.007

[13] A. C. Marosi and P. Kacsuk, "Workers in the clouds," in *PDP*, Y. Cotronis, M. Danelutto, and G. A. Papadopoulos, Eds. IEEE Computer Society, 2011, pp. 519–526.

[14] G. Kecskeméti, G. Terstyánszky, P. Kacsuk, and Z. Németh, "An approach for virtual appliance distribution for service deployment," *Future Gener. Comput. Syst.*, vol. 27, pp. 280–289, March 2011. [Online]. Available: http://dx.doi.org/10.1016/j.future.2010.09.009

 ISBN: 978-1-61208-153-3

One Click to Build An On Demand Virtual Cluster in Cloud Web-based Operating System with Dynamic Loading Prediction Scheduling Algorithm

Chang-Hsing Wu, Yi-Lun Pan, Hsi-En Yu, Hui-Shan Chen, and Ching-Wen Yu
Software Technology Division
National Center for High-Performance Computing
Hsinchu, Taiwan
E-mail：hsing@nchc.narl.org.tw, serenapan@nchc.narl.org.tw, yun@nchc.narl.org.tw, chwhs@nchc.narl.org.tw, cwyu@nchc.narl.org.tw

***Abstract*—As virtualization technologies become more prevalent, Cloud users usually encounter the problem of how to build his/her own virtual cluster with a friendly user interface for virtual resource management. To help resolving this problem, an On Demand Virtual Cluster system in Cloud Web-Based OS has been developed by the Pervasive Computing Team at the National Center for High-Performance Computing (NCHC). Through the On Demand Virtual Cluster system, with a click, Cloud users can customize and configure the specified virtual environment. We embedded the On Demand Virtual Cluster system into the Cloud WebOS, an extremely lightweight approach helping users to access virtual computing resources. The Cloud WebOS leverages virtualization techniques and cluster scheduling policy, which is the proposed dynamic loading prediction scheduling algorithm.**

Keywords - Virtualization Techniques; WebOS; Virtual Cluster; Cluster Scheduling Policy.

I. INTRODUCTION

In Cloud computing environment, there are various important issues, including information security, virtual computing resource management, routing, fault tolerance, and so on. Among these issues, the virtual computing resource management has emerged as one of the most important ones in the past few years. Currently, Cloud users have to manually build specified virtual cluster with the commend line mode in order to manage or generate virtual resources. To improve this condition, an On Demand Virtual Cluster system in Cloud WebOS (Web-Based Operating System) platform has been developed by the Pervasive Computing (PerComp) Team at the National Center for High-Performance Computing (NCHC). On this platform, Cloud users can build on demand virtual clusters with one click.

The Cloud WebOS platform provides a new service paradigm [1]. The WebOS infrastructure offers a seamless and unified access to geographical distributed resources connected via Internet, and it can supply most basic operating system services [2]. The proposed Cloud WebOS platform adopts the Asynchronous JavaScript and XML (AJAX) as a base. The major feature of this Cloud WebOS platform embedded with the On Demand Virtual Cluster system is that users can easily customize and configure their virtual environment according to their needs. It also can seek, diagnose, and monitor Cloud computing resources automatically. Meanwhile, the PerComp Team developed several Cloud widgets on the Cloud WebOS platform to control virtual clusters and virtual machines.

An efficient scheduling policy is indispensable, especially for distributed computing and Cloud computing. We designed an efficient scheduling policy, a dynamic loading prediction scheduling (DLPS) algorithm. It predicts loading of computing resources and makes the most adaptive resources allocation. The PerComp Team not only built the Cloud WebOS platform with the eyeOS [3] framework, but also incorporated the mechanism of scheduling algorithm.

In conclusion, the ultimate target of this research is to find a solution for scientists/researchers to painlessly run their jobs on Clouds. This research focuses on the development of friendly user interface, automatically dynamic allocation technique, integrated heterogeneous computing resources, and computing results visualization.

The rest of the paper is organized as follows. Section II presents related works. In Sections III, we proposed the On Demand Virtual Cluster system in Cloud WebOS and the dynamic loading prediction scheduling algorithm (DLPS). In Section IV, Cloud Widgets and experimental results are presented. Finally, the conclusion and future research directions are presented in Section V.

II. RELATED WORKS

A. *Existing Web-based Operating System (Web OS) Projects*

Recently, a famous WebOS - Chrome OS, developed based on AJAX technique [4]. It can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. The developments of Cloud WebOS platform via AJAX technique become practicable. However, Chrome OS does not provide on-demand applications and computing services to users in Clouds.

A Web-based Operating System (WebOS) project started at the University of California, Berkeley in 1996 as part of Network of Workstations [4]. So far, there are several typical commercial projects of WebOS, such as FlyakiteOSX [6],

 ISBN: 978-1-61208-153-3

Glide OS [7], XIN [8], and so on. All of these systems are online OS with Ajax and PHP techniques. However, these projects are not open source and lack of the management of distributed computing resources. To meet the demand of distributed computing resource management, the Cloud WebOS platform is developed. This development follows the spirit of open source, open standard, and GNU/GPL license.

B. *Virtualization Technologies*

Our research enhances the efficiency of job scheduling and retains the execution of parallel computing jobs via virtual technique. To implement virtualization technology, an additional software layer, called a virtual machine monitor or a hypervisor, has to be inserted between the existing operating system and hardware to manage the resources and virtual machines. The characteristics of virtualization technology are described as the following:

- Utilization – better utilization means various services run on one physical machine with multiple virtual machines (VMs);
- Isolation - better isolation means a VM can halt and catch fire without affecting the real host or other running VMs;
- Flexibility - the ability of the virtualization technologies to run platforms and operating systems that are different from the host, good flexibility means more choices for VM platforms and the ability to run VMs with minimal modification;
- Manageability - availability of tools and APIs for starting, stopping and moving VMs.

Generally, modern hypervisor implementations are divided into two categories, including Host-based and Bare-metal approaches. The host-based approach uses modified operating systems to provide virtual machine monitoring, such as Linux-VServer [9], Solaris Zones [10], and Kernel-based Virtual Machine (KVM) [11]. On the other hand, the bare-metal approach employs small-dedicated hypervisors to directly run on physical machines. The VMware ESX server [12], and the XenServer [13] are the famous examples of the bare-metal approach.

With success of the virtual technologies, we integrate virtualization technology – KVM and WebOS. This research comes up with a new and lightweight approach to access virtual computing services via the On Demand Virtual Cluster system.

III. PROPOSED ON DEMAND VIRTUAL CLUSTER SYSTEM IN CLOUD WEBOS

A. *Research Objective*

The key idea of Cloud Computing lies in its component-based nature, which are reusability, substitutability and user friendly. By integrating virtualization technologies and WebOS, we provided a web environment to access Cloud services via Cloud Widgets in the Cloud WebOS. This progress helps to lower the barrier for using Clouds. In order to develop an autonomic virtual computing resources management system based on decentralized resource discovery architecture, we implemented the On Demand Virtual Cluster system in Cloud WebOS. At the same time, an efficient scheduling policy is also important. Therefore, the dynamic loading prediction scheduling (DLPS) algorithm is used for the scheduling of virtual cluster and physical cluster. It predicts loading of computing resources and makes the most adaptive resources allocation.

As the Figure 1, it shows a high level overview of the Cloud WebOS. In the middle of this figure, when the Cloud WebOS receives a Cloud job request from the users via the web browser, and then the job will be sent to the fittest virtual cluster in the backend to process via the On Demand Virtual Cluster system. The system will help users to generate the fittest virtual cluster and choose/allocate the most adaptive physical resources with a graphical interface.

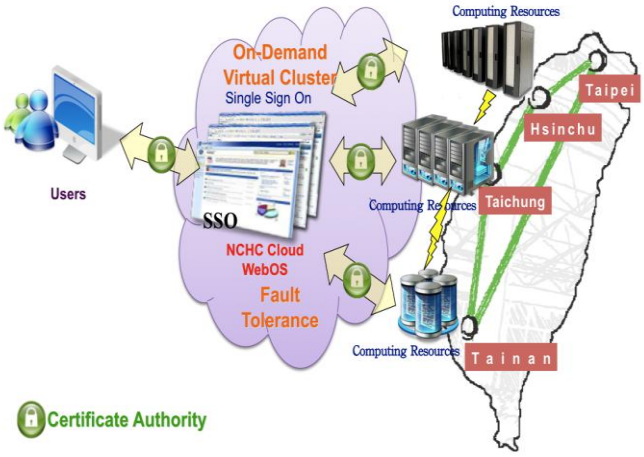


Figure 1. The Overview of Cloud WebOS Platform

B. *Implementation and System Architecture*

In this project, we combine the WebOS platform with Cloud computing resources to offer users a friendlier Cloud environment. The system architecture of the On Demand Virtual Cluster system in Cloud WebOS is sketched in the Figure 2. In the Cloud WebOS, upon receiving a Cloud job request from the end users via Web Browser, the system acquires Cloud Services via Cloud Widgets, which in turn connect the Image Creator Widget, Virtual Machine (VM) Creator Widget, VM Monitor Widget, and VM Control Widget, within On Demand Virtual Cluster system. Each of Cloud Widgets is described in Section IV. The system helps selecting the most adaptive computing resources to create virtual clusters automatically based on the demands from the end users. These Widgets of On Demand Virtual Cluster system in Cloud WebOS also drive the Cloud middleware to operate physical computing resources and storages.

 ISBN: 978-1-61208-153-3

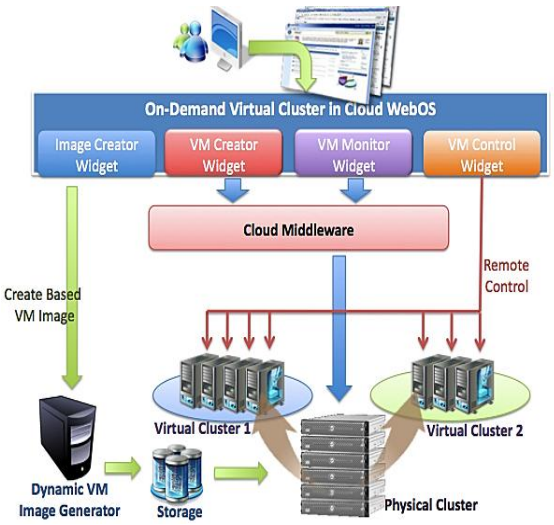


Figure 2. The System Architecture of On Demand Virtual Cluster System in Cloud WebOS

Meanwhile, as the Figure 3 is shown, the end users connect the Application Pool to get the software services, such as information security and Bio simulation via Cloud WebOS easily. After connecting Application Pool, the Cloud WebOS also can integrate the public service provider, such as Amazon EC2 [14] and so on. Upon receiving Cloud job request via Cloud WebOS, the On Demand Virtual Cluster system makes communication with Cloud Middlewares, which are Data Broker, Monitoring & Reporting, and Dynamic Provisioning. The Data Broker collects data from the distributed physical sensors. The Monitoring & Reporting takes responsible for monitoring the status of physical machines and virtual machines. Finally, the Dynamic Provisioning provides the capability of resource allocation automatically, and the feature of DLPS algorithm is activated at the same time. The DLPS algorithm can improve the performance of the dynamic scheduling over conventional scheduling policies.

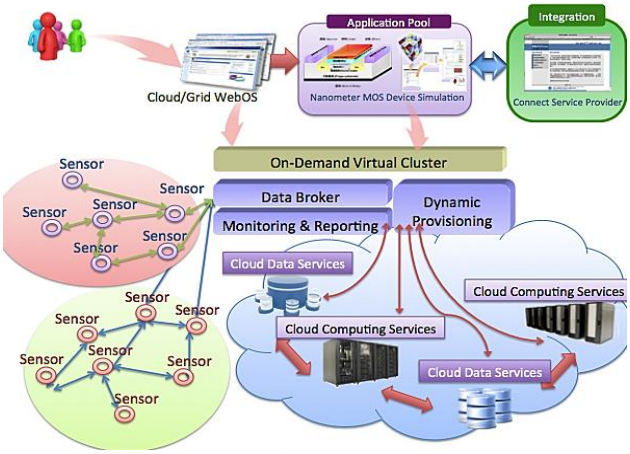


Figure 3. The Application of On Demand Virtual Cluster System

With On Demand Virtual Cluster system in Cloud WebOS, users can create a dynamic HPC cluster consisting of VMs. The scale of each virtual cluster can be determined by user's criteria. When user needs virtual cluster no longer, the virtual cluster can be destroyed. Then the computing resources are released. The whole operation can be manipulated via web browser, because we use XML-RPC based Application Programming Interface (API). Moreover, there are two middlewares embedded into the proposed Cloud WebOS, as the following shown: 1) *Integration with OpenNebula* – OpenNebula is used as central cloud management [15]. It is responsible for finding available computing resources, creating VMs based on a selected image, and deploying the image into the physical computing resources. It also manages unique MAC address, IP address, and virtual network (vNet) ID. Therefore, each user's cluster lives on its own vNet, in order to isolate the various virtual clusters. 2) *Batch System with Torque* – It is used for the scheduling of virtual cluster and physical cluster. The DLPS algorithm is embedded with this resource manager, Torque [16] in the Figure 4. This development not only makes users submit job as usual via PBS_SERVER, but also makes resource manager have additional capabilities of loading prediction and job scheduling with virtual technology. The detail explanation is in Section III-C.

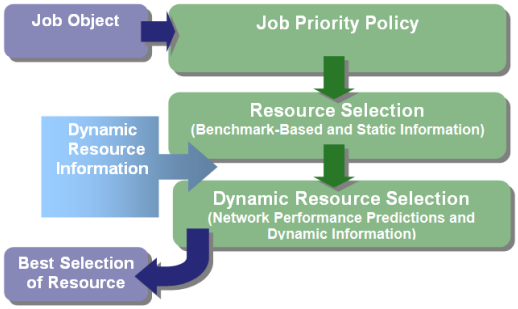


Figure 4. The Scheduling Policy - DLPS in Cloud WebOS

C. *Scheduling Policy - Dynamic Loading Prediction Scheduling (DLPS) Algorithm in Cloud WebOS*

The presented scheduling policy is called Dynamic Loading Prediction Scheduling (DLPS) algorithm. It can schedule the computing resources in Clouds and even multiple clusters. The objective function of DLPS is achieving the minimized makespan (defined in Definition 1). Thus, we designed the following equation to describe the objective function, as in (1).

$$M^* = Min[\max(d_k) - \min(s_k)] \quad (1)$$

The above equation (1) is defined in ***Definition 2***.

Definition 1: The completion time is defined as the time from the job being assigned to one machine until the time the job is finished. The complete time is also called makespan time.

Definition 2: M^* means the minimized makespan. In order to predict precisely, there are two parameters - d_k and S_k. d_k is the maximum *job ending time* of the *k*th job, which means the end time of job completed. And S_k is the minimum *job submitting time* of the *k*th job, which means the time stamp when users submit the *k*th job.

The each step of DLPS algorithm is descried as the following pseudo code in Figure 5.

Input: *Cloud WebOS form-specification(s) of jobs requirements and users' criteria*

Objective Function: $M^* = Min[\max(d_k) - \min(s_k)]$

It *means the time of total deliver or response time is minimum.*

Output: *The most appropriate resource candidate and available resources list*

1: ***Process users' criteria and job requirements from Cloud WebOS, High-Performance Linpack Benchmark, data set, execution programs, and Queue type, etc.***
2: ***Make communicate with each Cloud resource for getting the static and dynamic resource in formation.***
3: ***For each job*** {
   i. Store the features and status of each cluster and Cloud resource into Database
   ii. Call the adaptive resource allocation function to compare the free nodes with requirement nodes, and then filter out unsuitable resources.

4: Y = (Tem_value); initial value
5: ***if*** (free nodes ≥ requirement nodes){
6: With Definition 3, it calculates higher weight;}
7: ***else*** (free nodes < requirement nodes){
8: call dynamic loading prediction function with predict EstBacklog (Definition 4) and Job Expansion Factor (Definition 5) methods to calculate final weight through Definition 6;}
9: Calculate the *Min(Y)* = *M**;
10: }

Figure. 5 The Pseudo of DLPS

The logical flow chart of the DLPS is illustrated as in the Figure 6. First, the DLPS retrieves the information of computing resources from the local queuing system, and then filters out unsuitable resources with the adaptive resource allocation function. After using adaptive resource allocation function, DLPS compares free nodes with required nodes. If current free nodes are enough, DLPS will give higher weight (defined in Definition 3). Otherwise, the following step enters the dynamic loading prediction function with EstBacklog and minimum Job Expansion Factor (defined in Definition 4 and Definition 5) methods to predict which computing resources respond and execute job quickly, and then calculate the weight (defined in Definition 6). Finally, the DLPS ranks all available resources and selects the most appropriate resources to dispatch job and generate virtual machines.

Definition 3: When free nodes fulfill required nodes, the weight of *k*th job is designed as following:

$$Weight_k = (R_{nodes} / f_{nodes}) \times M_{capability}$$

Where R_{nodes} means the number of required nodes, f_{nodes} means the number of free nodes, and $M_{capability}$ means the capability of each computing resources. The capability is based on static information, such as High-Performance Linpack Benchmark results, and HPC Challenge Benchmark.

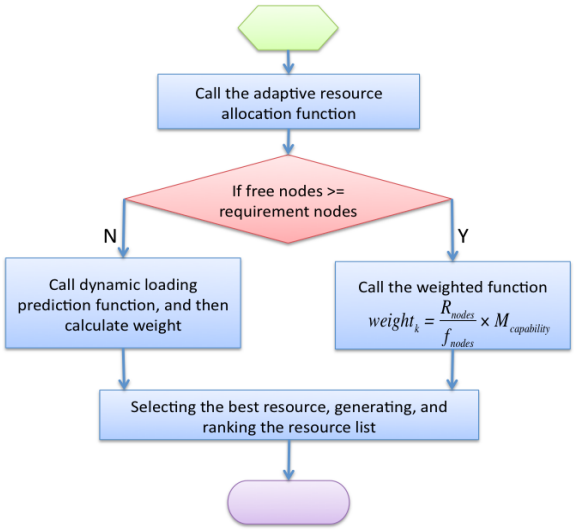


Figure. 6 The Logical Flow Chart of DLPS

Definition 4: The *EstBacklog* means estimated backlog of queued work in hours. The general *EstBacklog* form is shown as the equation (2):

$$EBL = (\frac{QueuePS \times CPUAccuracy}{TotalJobsCompleted}) \times \qquad (2)$$

$$(\frac{TotalProcHours \times 3600 \times AvailableProcHours}{Dedicated\,ProcHours})$$

QueuePS is the idle time of queued jobs. *CPUAccuracy* is the actual run time of job. *TotalJobsCompleted* is the number of jobs completed. The *Toatl ProcHours* is the total number of proc-hours required to complete running jobs. The *Available ProcHours* is the total proc-hours available to the scheduler. The last variable, *Dedicated ProcHours*, is the total proc-hours made available to jobs.

Some of above values are from the system historical statistic values of queuing system loading and the others are from real-time queuing situation. The output is divided into two categories, running and completed. The Running

statistics include information about the currently running jobs. The completed statistics are compiled using historical information from both running and completed jobs. Therefore, the *EBL* can forecast the backlog of each computing site with above information.

Definition 5: The *job expansion factor* subcomponent has an effect similar to the queue time factor but favors shorter jobs based on the requested wallclock run time. The job expansion factor metric is calculated by the information from local queuing system as described in the equation (3):

$$JEF = \frac{QueuedTime + RunTime}{WallClockLimit} \quad (3)$$

Definition 6: After getting *EstBacklog* and *job expansion factor*, the metric is calculated by the following equation (4):

$$Weight_k = \lambda \times \frac{JEF_k}{TotalJEF} + (1-\lambda) \times \frac{EBL_k}{TotalEBL} \quad (4)$$

The $Weight_k$ means the weight of the *k*th job. $\frac{JEF_k}{TotalJEF}$ means the *JEF* of *k*th job divided by the Total *JEF*. $\frac{EBL_k}{TotalEBL}$ means the *EBL* of *k*th job divided by the Total *EBL*. Where λ is the modulated parameter of system, which can be obtained from numerous trials. The EstBacklog can be respected the dynamic situation of queuing system generally. Therefore, it always uses the higher λ value. Consequently, we can use above parameters to calculate the minimum time of total deliver and response time.

IV. CLOUD WIDGETS/EXPERIMENTAL RESULTS

A. *Cloud Widgets*

In addition to the basic Widgets, more advanced Cloud Computing Widgets are attempted as well. We have developed many Cloud Widgets with friendly graphical user interface in WebOS. The kernel of the On Demand Virtual Cluster system architecture consists of four Widgets, including Image Creator Widget, VM Creator Widget, VM Monitor Widget, and VM Control Widget. Users without much learning effort can easily manage all of these widgets.

The Image Creator Widget, in the Figure 7, is to generate the customized base image and the on-demand/specified applications from the end users' requirements. This Widget provides a complete and integrated HPC software stack that consists of operating system, management tools, resource monitor, and even commercial package, such as the Matlab. The VM Creator Widget - with the profile of virtual cluster demanded by the user provided, it will generate a specification, shown in the Figure 8, which in turn is parsed by the VM Creator engine to create specified virtual cluster on the physical computing resources. Thus, after completing the profile of virtual cluster, with a click, Cloud users can customize and configure the specified virtual environment in real time.

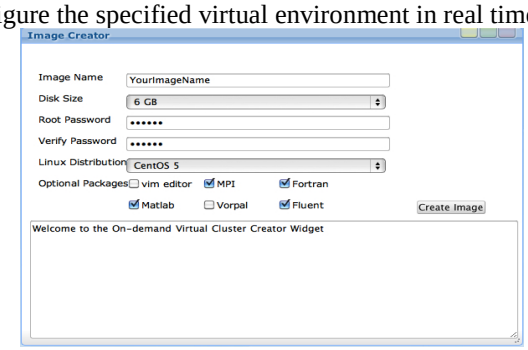


Figure 7. Image Creator Widget

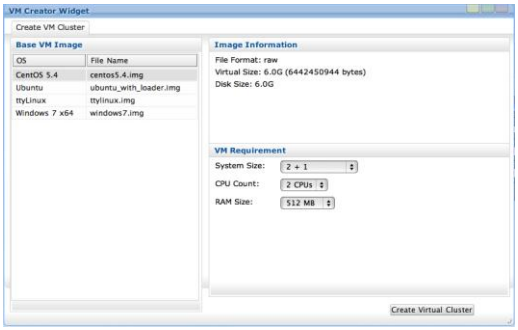


Figure 8. VM Creator Widget

In the Figure 9, the main task of the VM Monitor Widget is to monitor the all the status of virtual machines, Networks, and the physical hardware. The VM Monitor also can show the current loading of physical machines, in the Figure 10. The VM Control Widget provides users to access, ssh, or operate virtual machines through Cloud visualizer, as shown in the Figure 11 and Figure 12. We used above Cloud Widgets to implement the following two customized applications for biological simulation and information security simulation. The F-motif Simulation Widget provides specialized Cloud services to search and analyze the sequence of gene in real time, in Figure 13. The other customized Cloud Widget about information security is called ICAS (IDS-log Cloud Analysis System) Widget. As long as user selects the ICAS base image, the Hadoop DB and virtual cluster are constructed automatically, and then users can analyze the IDS-log as the Figure 14 shown.

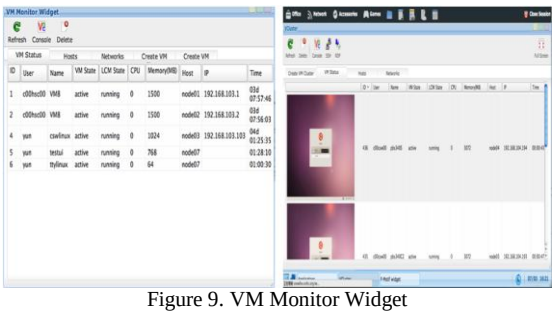


Figure 9. VM Monitor Widget

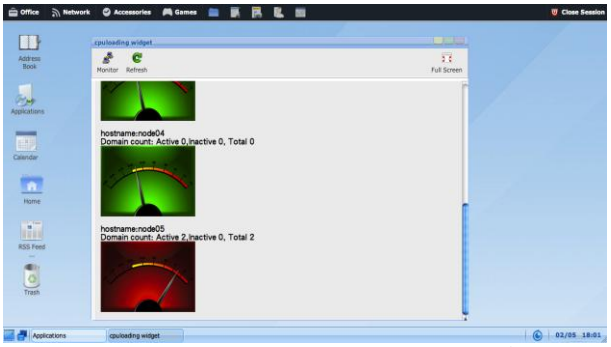


Figure 10. VM Monitor Widget – The Current Loading of Physical Machines

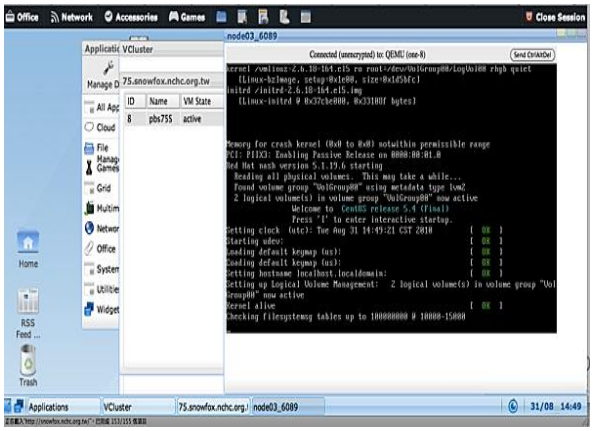


Figure 11. VM Control Widget Cloud Visualizer – Linux Booting Status

Figure12. VM Control Widget Cloud Visualizer – Windows 7 Booting Status

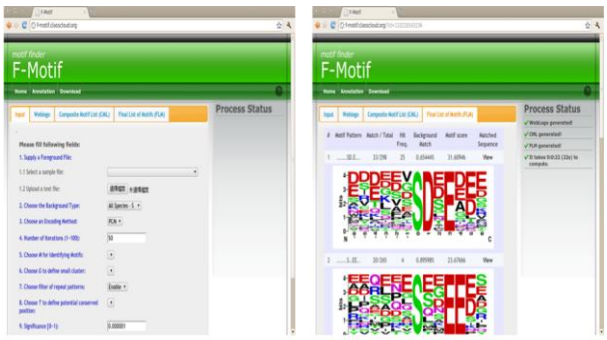


Figure 13. F-motif Widget

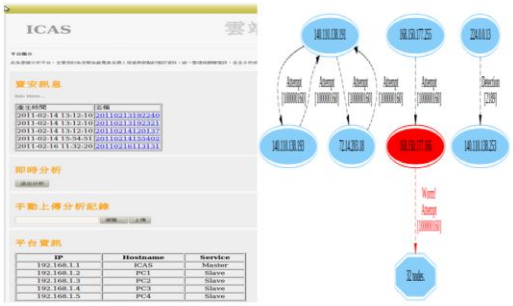


Figure 14. ICAS Widget

B. *Experimental Results*

The preliminaries of experiment are needed to set up, including the multi-sites physical computing environment, the virtual machine – KVM, Network Speed Test [17], and Disk I/O test tool - bonnie++ [18]. As shown in Table I, we list the summary environment characteristics of NCHC computing resources.

TABLE I. SUMMARY ENVIRONMENT CHARACTERISTICS OF NCHC COMPUTING RESOURCES

Resource	CPU Model	Memory (GB)	CPU Speed (MHz)	#Cores	Nodes	Job Manager
Snowfox	Intel(R)Xeon(R) CPU 2.5GHz, E5420	16	2500	112	14	Torque
Capri	Intel(R) Xeon(R) CPU E5620 , 2.40GHz	32	2400	232	29	Torque

There are three scenarios, including the performance of Network I/O, Disk I/O, and the DLPS algorithm. Moreover, in order to improve the performance, we use the Virtio driver [19] in the virtual machines. Virtio driver provides paravirtualized functions for network virtualization and disk I/O virtualization. In Figure 15, we found the Network speed is tackled about 166 Mb/s without Virtio, because the I/O bottleneck is between virtual machine and hypervisor. Therefore, the Virtio is activated in the On Demand Virtual Cluster system. The performance of Network I/O is nearly the same with native machine. In the performance of Disk I/O scenario, we compared with Virtio and without Virtio. With Virtio, it can be improved write performance about 120% and read performance about 20%, as the following Figure 16 is shown.

The performance of DLPS algorithm is compared with several algorithms, such as Round Robin, Short-Job-First (SJF), Big-Job-First (BJF), and First-Come-First-Serve (FCFS). We submitted testing jobs, which were generated randomly with the synthetic models as the Figure 17 is shown. The vertical axis is the value of min makespan (seconds), and the horizontal axis is the number of jobs. The makespan of DLPS algorithm is notably less than other algorithms, especially when a huge number of jobs are submitted. Therefore, the objective function of DLPS approaches the minimized makespan. The dynamic loading prediction characteristic of presented system is proved to be better in this experiment.

 ISBN: 978-1-61208-153-3

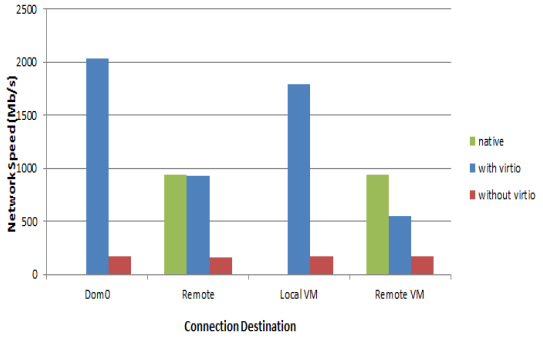


Figure 15. The Performance of Network I/O

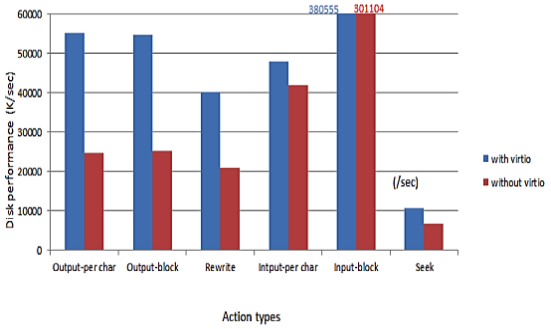


Figure 16. The Performance of Disk I/O

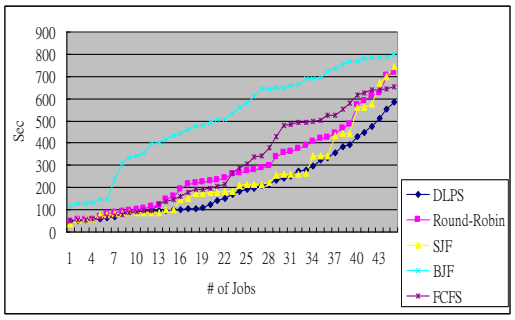


Figure 17. Compare Makespan of DLPS with Other Algorithms

When a small number of jobs are submitted, the efficiency of DLPS may be worse than other algorithms, especially for SJF and FCFS. This situation is reasonable, because small jobs are easy consumed by SJF and FCFS. When the number of jobs is increasing, the developed DLPS is absolutely better than SJF and FCFS, because the notable drawback of SJF and FCFS happens, which the large numbers of jobs are queued inefficiently in the local scheduler of cluster. Comprehensively the above efficiency figure, the best efficiency of DLPS occurs at full usage of each cluster.

V. CONCLUSION AND FUTURE WORK

The research – On-Demand Virtual Cluster system in Cloud WebOS, provides Cloud users with an interface that is user-friendly, more straightforward, and more efficiency. The On Demand Virtual Cluster System in Cloud WebOS not only helps user to build virtual cluster easily and automatically, but also provides different varieties of computing environment such as Linux, Win7, and so on. The Virtio driver is activated in the On Demand Virtual Cluster system. Thus, the performance of Network I/O is nearly the same with native machine. The performance of Disk I/O can be improved write performance about 120% and read performance about 20%.

Furthermore, the research leverages virtualization techniques combined with cluster queuing system and job scheduling mechanism. According to the pervious experiment, the DLPS has better efficiency than other scheduling algorithms; especially the huge numbers of job are submitted into the computing cluster. Finally, we obtain an important property that the algorithm is appropriate to deal with large amount of jobs in real Clouds or distributed environment.

VI. REFERENCES

[1] W. Kim, “Cloud computing: Today and Tomorrow,” Journal of Object Technology, 8, 2009.

[2] G. Gu and X. Lu, “Simple Web OS System Based on Ext Framework and Cloud Computing,“ International Forum on Information Technology and Applications, pp. 448-450, IEEE, 2010.

[3] http://www.eyeos.org/, [accessed: July, 2011].

[4] http://www.chromium.org/chromium-os/, [accessed: July, 2011].

[5] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A, Patterson, “Searching for the Sorting Record: Experiences in Tuning NOW-Sort,” *The 1998 Symposium on Parallel and Distributed Tools (SPDT '98)*, Welches, Oregon, August 3-4, 1998.

[6] http://osx.portraitofakite.com/logon.htm, [accessed: July, 2011].

[7] http://www.glidedigital.com/, [accessed: July, 2011].

[8] http://www.xindesk.com/, [accessed: July, 2011].

[9] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors, " Proceedings of ACM SIGOPS/Eurosys European Conf. on Computer Systems, pp. 275-287, Mar. 2007.

[10] D. Price and A. Tucker, "Solaris Zones: Operating Systems Support for Consolidating Commercial Workloads, " Proceedings of 18th Large Installation System Administration Conf., pp. 241-254, Nov. 2004.

[11] B. Zhang, X. Wang, R. Lai, Y. Liang, Z. Wang, Y. Luo, and X. Li, "Evaluating and Optimizing I/O Virtualization in Kernel-based Virtual Machine (KVM)," International Conference on Network and Parallel Computing, pp. 220-231, Zhengzhou, China, September 13-15, 2010.

[12] John Paul, "VMWare ESX Server Workload Analysis: How to Determine Good Candidates for Virtualization," 33rd International Computer Measurement Group Conference, pp. 483-484, San Diego, CA, USA, December 2-7, 2007.

[13] X. Ge, H. Jin; S. Wu, X. Shi, W. Gao, "A method of multi-VM automatic network configuration," Intelligent Computing and Intelligent Systems, pp. 309-313, 2009.

[14] http://aws.amazon.com/ec2, [accessed: July, 2011].

[15] Ignacio M. Llorente and Rubén S. Montero, “OpenNebula: A Cloud Management Tool,” Internet Computing, IEEE, pp. 11-14, March-April 2011.

[16] http://www.clusterresources.com/products/torque-resource-manager .php, [accessed: July, 2011].

[17] http://vmstudy.blogspot.com/2010_04_01_archive.html, [accessed: July, 2011].

[18] http://www.coker.com.au/bonnie++/, [accessed: July, 2011].

[19] http://www.linux-kvm.org/page/Virtio, [accessed: July, 2011].

 ISBN: 978-1-61208-153-3

Understanding Cloud Requirements - A Supply Chain Lifecycle Approach

Maik A. Lindner
SAP Research
SAP Labs, LLC
Palo Alto, CA
m.lindner@sap.com

Fiona McDonald
SAP Research
SAP (UK) Ltd.
Belfast, Northern Ireland
fiona.mcdonald@sap.com

Gerard Conway
Innovation Value Institute
National University of Ireland,
Maynooth, Ireland
gerard.conway@nuim.ie

Edward Curry
Digital Enterprise Research Institute
National University of Ireland,
Galway, Ireland
ed.curry@deri.org

***Abstract*—Cloud Computing is offering competitive advantages to companies through flexible and, scalable access to computing resources. More and more companies are moving to cloud environments; therefore understanding the requirements for this process is both important and beneficial. The requirements for migrating from a traditional computing environment to a cloud hosting environment are discussed in this paper, considering this migration from a supply chain lifecycle perspective. The cloud supply chain is examined from a lifecycle perspective for the management of the migration project. This paper illustrates the requirements that need to be considered when adopting a cloud migration strategy and the steps to take in order to manage this process.**

***Index Terms*—cloud computing; supply chain; cloud sourcing; cloud lifecycle.**

The cloud provides scalable, on-demand network access to virtualised computing resources [1]. This is a very attractive concept for enterprise Information Technology (IT) landscapes to adapt. However as with any new concept or emerging technology, IT departments face challenges with the opportunities being offered by the cloud. Some of the challenges include security, privacy and lack of control. The physical location of hardware in addition to who can access the data is not always known which, can lead to security and privacy issues. As the cloud is run by a cloud service provider, users have limited control of factors such as maintenance or resource usage as these are the responsibility of the cloud service provider. Although cloud computing reduces capital expenditure by using a pay-per-use model, there can be hidden costs in order to ensure adequate backups and disaster recovery processes are in place. Despite these drawbacks many companies still strive for cloud adoption as the advantages more than compensate for these drawbacks, e.g., the cost benefits including, scalability and flexibility. Cloud computing resources can be ”right-sized” to meet real-time requirements. When high capacity is needed at peak times the cloud can provide additional resources on-demand, these can be instantly adjusted when less capacity is needed. The functional benefits of cloud computing consist of increased response times as well as instant software updates that are automatically provided. Other benefits of the cloud include resource benefits, as employees can access information anywhere and can focus on high priority tasks rather than the routine maintenance tasks. These are a selection of the reasons enterprises want to move to the cloud.

Businesses in the cloud computing area are interconnected by what is known as the cloud supply chain [2]. This can be defined as two or more parties linked by the provision of cloud services, related information and funds [2]. These businesses are involved in the end-to-end provision of products and services from the cloud service provider for end cloud customers. Within the cloud supply chain, there are several components and actors. There always exists a product/service at the beginning of the supply chain and a consumer at the end who is requesting the product/service. E.g., on-demand software could be the product/service that is provided by the cloud service provider to the customer (cloud consumer) who wants to use the software.

A well-defined cloud supply chain is needed to encourage the adoption of cloud computing. Not only are products and services passed through the supply chain but also information and funds. It is important to be aware of what and who is involved in the cloud supply chain to understand the potential of this new technology chain and how it is used to identify the requirements of moving to the cloud. The cloud supply chain clarifies the process involved with both providing and consuming cloud services. Supply chains generally serve two functions, a physical function which, is the production of the product and transportation of all components to the right place and a market mediation function which, ensures the product meets market needs. A supply chain must be classified according to its components and the end-product it supplies.

For a business to successfully utilise the cloud, it needs to migrate some or all of its IT services to the cloud, and then manage the new environment. The research undertaken has shown that by using a cloud lifecycle [3] both the migration and the on-going management of the cloud can be planned and controlled to ensure success. The lifecycle provides a mechanism of breaking down all of the activities required for a move to the cloud, into discreet manageable steps which, allows an organisation to seamlessly migrate its services, whilst minimising the risk to the business.

The following sections of the paper are structured as follows: In Section 1, cloud computing is discussed on a company level from a supply chain approach, and the different cloud types and service models are considered as well as an analysis of the cloud supply chain by identifying the actors within the

 ISBN: 978-1-61208-153-3

chain. In Section 2, the structural setup of the supply chain is also considered based on the identified needs. Section 3 looks at the requirements from a lifecycle approach for cloud migration and on-going management. Section 4 includes the conclusions and future work.

I. Introducing Cloud Computing on a Company Level - A Supply Chain Approach

There are various models and ways in which, cloud computing can benefit a company. It is important to understand, that there are multiple cloud types and service models that can be adopted and these need to be considered as requirements. Each company's individual setup will determine the model and the benefits of these services. Based on the identified type of cloud to use and the services needed, a supply chain setup needs to be established and relevant partners and distribution channels need to be chosen.

A. Service Models and Cloud Types

In order to understand the concept of the cloud supply chain, it is important to be aware of what it is composed of. There are different service models that need to be considered in the cloud such as:

- Infrastructure as a Service- this focuses on providing the resources for the service such as, network, memory and storage capacity which, is essentially the primary stage of the process.
- Platform as a Service - this is the second stage that presents the user with an additional abstraction level for software to run on or for the user to build on [4].
- Software as a Service - this provides complete turnkey software applications that may be of interest to the users and allows these to be fully-utilised using the cloud.

Each of these service models can be used more than once in the cloud supply chain. As these not only provide single services, they can also be combined to provide value-adding services that act as single objects in the cloud supply chain. These aggregated services can be made up of two or more services, e.g., Infrastructure and Platform can be combined as a service for software developers.

The different types of clouds that can be used to consume cloud services are public, private, hybrid and community clouds, and the decision to use a particular cloud can depend on the individual business needs and requirements. However, the current position of the company also needs to be considered. Firstly, the choice of the appropriate cloud to use depends on the prerequisites within a company; if they have an existing data center, they may be more likely to choose a private cloud structure as their current data center can be reconstructed. On the other hand, a company with no data center what so ever may go for public cloud with zero upfront spending. In practice, consumers use different cloud instances to fulfill different requirements, e.g., they may use a private cloud for data storage and a public cloud for everyday processes such as e-mail. This leads to using a hybrid model which, allows the consumer to source from multiple clouds, therefore resulting in better value and a customised service.

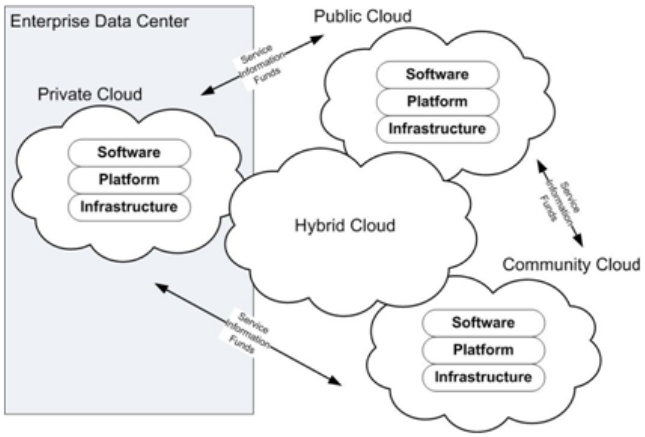


Figure 1. Composite Service Cloud

Hybrid landscapes can be defined as an IT environment that uses both public and private clouds. This type of cloud allows users to take advantage of the scalability and reduction in Capital Expenditure (CAPEX) yet still have the security of a private cloud. Effective and efficient management of hybrid landscapes will allow for users to receive better benefits and an optimal service from the hybrid cloud. It is important to consider the cloud types when determining the requirements for migrating to the cloud.

Figure 1 illustrates the transfer of the service, information and funds through the supply chain of each cloud and shows how a hybrid cloud is made up of a combination of two or more of these clouds. Looking at the area of cloud computing from different perspectives raises the issue of conflicting aims between the provider and consumer of the cloud services. We will look at who is involved in each of these models and the relationship between these.

B. The Cloud Supply Chain

Once a decision on a specific cloud type and service setup has been made, the comprehensive supply chain can be determined and built up. For this, relevant partners have to be identified and a clear product structure has to be established. The cloud supply chain is illustrated in Figure 2 showing the components and actors within the chain. The product is passed along the supply chain to the end-customer. The service provider can provide the end-customer with just one service (software, platform or infrastructure) or they can act as a service aggregator and combine these services to provide a composite service for the customer.

Accounting, billing and monitoring should also be considered throughout the cloud supply chain when understanding requirements, as information and funds are passed through the chain. As a continuous example of a public cloud provider, we will use Amazon who provide an IT company with storage using Amazon Simple Storage Service (Amazon S3). Various

 ISBN: 978-1-61208-153-3

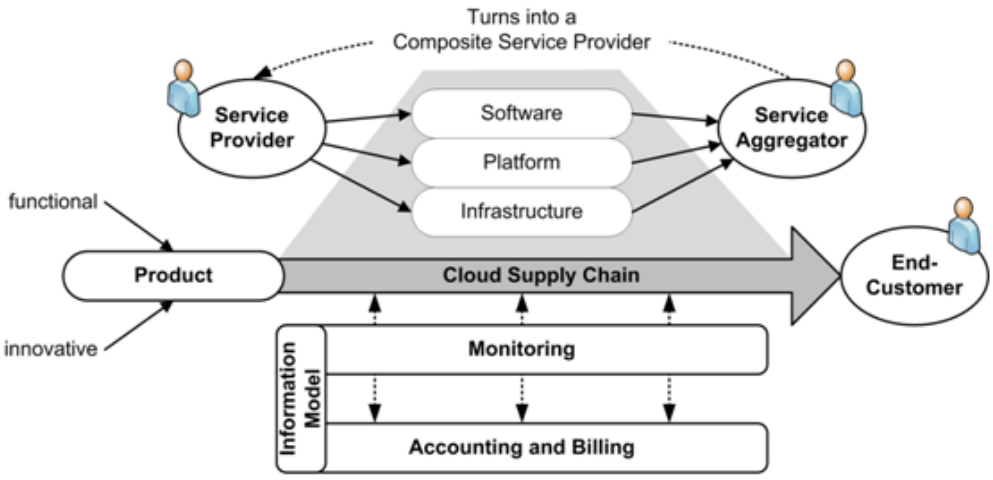


Figure 2. Cloud Supply Chain

actors and goods have to be considered to determine and define a supply chain. The cloud service provider is an actor who provides the service to the end-customer and they can take various roles depending on whether they provide infrastructure, platform or software as a service. In this case the service provider is Amazon who is providing the IT company with storage for their data. Amazon can have direct contact with the customer or they can act as a broker who uses the service and combines it with other services in order to enrich it. This depends on the needs of the IT company, so if they know specifically the service they require, they can directly communicate with Amazon for that service. If they are unsure and only have an idea of what they need, they could use a cloud broker, who will find the best suited package from a selection of service providers for the consumer. However, sometimes if a broker is involved, the service provider is in contact with the broker who then deals with the end-customer.

A broker combines and enriches the services provided by Amazon with and by others to provide a composite service for consumers (the IT company). Therefore, the product for the end-customer is an enhanced service provided in a flexible manner. The broker communicates with the service provider and the end-customer, therefore the IT company receives the cloud service through the broker. It is important to maintain visibility and transparency of all processes and data within a supply network to ensure the end-product remains clear and defined. The end-customers usually consume a product that is a single or composite service which, is provided by a service provider over the cloud supply chain [2]. It is important to examine the cloud supply chain and to be aware of the requirements in order to decide whether a broker is preferred to receive a cloud service. With many actors involved, it is important to maintain clarity and visibility within the supply chain especially when it can become quite complex.

As well as the actors in the supply chain, the components and structural setup needs to be considered when determining the requirements of a cloud project.

II. Structural Setup of the Supply Chain Based on Identified Needs

The setup of the supply chain depends on the needs and requirements of the organisation. This section discusses the components of the supply chain and considers the possible complexities involved.

A. Components Within the Supply Chain

Components within the supply chain that lead to costs include the management and restructuring of services, information and funds. The typical payment model for cloud is pay-per-use, however providers such as Salesforce and Microsoft use a subscription based model for payment of their services. The pay-per-use model is one of the key benefits that outweigh the traditional method of fixed-rate exploitation. These funds flow from the service provider to the cloud infrastructure provider who provides the IT infrastructure. However it can be considered that this can flow the opposite way in some circumstances if there has been a violation in the Service Level Agreement (SLA), which, would result in a compensation penalty from the supplier.

B. Implications of a Complex Supply Chain

In order for users to receive the best possible service to fulfill their requirements, it is important to consider that more than one type of cloud can be used in the supply chain. Using a number of various components such as services or types of clouds can cause the supply chain to be complex and this needs to be considered. Depending on user requirements or company requirements, one cloud may not be able to offer the complete service they have requested. For example, a user/company could request a service in a public cloud but require some of the data to be in a local cloud, i.e., within a certain region. Therefore the cloud could outsource this portion of the service temporarily to a cloud within their desired region in order to

 ISBN: 978-1-61208-153-3

be able to offer the consumer the full package. In-sourcing of previously outsourced solutions to the cloud can also be considered, so the data in the local cloud could be moved back to the original when required. The different types of clouds can work as a synergy to provide the best service for the consumer.

From a consumer perspective, the consumers are receiving the optimal service to meet their needs as a result of the hybrid model as well as a more efficient supply chain. By choosing to outsource non-core service capabilities to the public cloud, it will allow them to develop a dynamic service supply chain [5]. Most consumers would not choose to outsource their core service capabilities as they are more secure within their private cloud.

On the other hand, from a provider perspective running e.g., a private cloud within a business, at seasonal or event-based peak of traffic, they can move their data or applications to an external cloud to cope with the surge of work. When the work calms down to the normal pace, they can in-source their data or applications back into their private cloud. This eliminates the need to purchase additional hardware and software for those peak times only, saving costs in the short-term and long-term.

This process of leasing compute capacity from an external cloud in peak times is called cloud bursting. It is useful if additional compute capacity is required in a short period of time, as this can be leased from a cloud service provider for the required time. The resources acquired from the cloud service provider are, secured, provisioned, and made available to load balancers so they then have the ability to manage the additional requests. This can happen on an approved, scheduled, or as-needed basis [6]. From an internal IT provider perspective, with a setup of an internal cloud, cloud sourcing and cloud bursting offer numerous cost and value benefits to their business. From an end-consumer perspective, all of the advantages of using cloud are relevant, as well as the additional benefit of receiving the best possible solution to meet their needs through the use of the hybrid model as long as all of the initial requirements are fulfilled.

By analysing the cloud supply chain, the technical requirements for migrating to the cloud can be identified. As discussed, the type of clouds need to be considered and whether more than one cloud will be used, as well as the number of services required, what pricing model suits best and whether to use a broker or directly contact a cloud service provider.

III. The Cloud Lifecycle

Management of this process can be carried out by using the cloud lifecycle. This represents the process of moving from a traditional to a cloud infrastructure. This section describes each of the steps in the lifecycle and how the supply chain plays an active role in this process.

The lifecycle has four phases and eight steps that have been proposed to follow in order to manage the process of migrating to the cloud. It is an improvement cycle, therefore allowing the process to be evaluated and improved continually. Each step is explained in the following section.

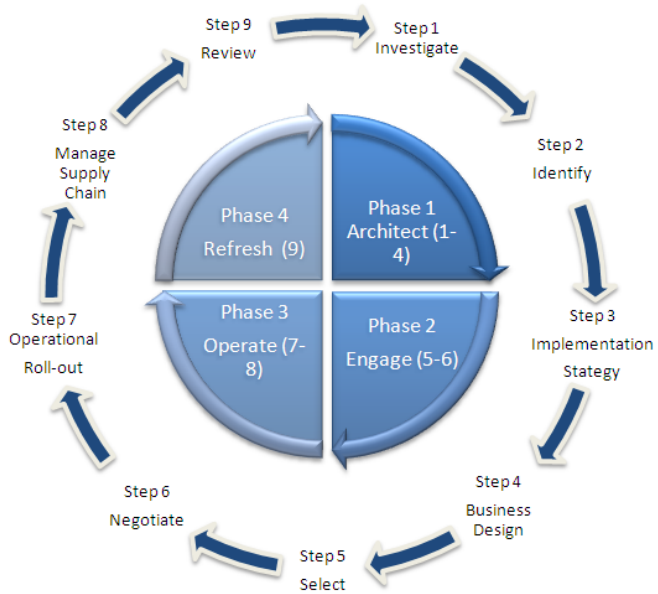


Figure 3. Cloud Lifecycle

A. Steps of the Lifecycle

Phase 1: Architecture

1) Investigate
This step provides an insight and understanding of what the organisation wants to achieve by moving to the cloud and what goals and expectations are to be met. This will be based on an analysis of the appropriate industrial segment, with insights from experts and experiences from peer organisations, together with knowledge of potential suppliers.

Outputs:

- IT strategy for cloud computing
- Strategic intent of moving to the cloud and how it progresses the business objectives
- Intelligence document on service offerings and providers
- A document describing what will be achieved by comparing the strategic requirements with the available services and providers

2) Identify
The purpose of this step is to objectively assess what areas of the business are appropriate to outsource to the cloud and what impact this will have on the current delivery model. This will require an understanding of the current state, so that it can be compared to the desired future state. At least, the impact on the service, people, cost, infrastructure, stakeholders and how it will

be managed should be considered.

Outputs:
- List of services to be outsourced to the cloud
- Outsourcing delivery model
- The current and future states of the IT structure

3) Implementation Strategy
The aim of this step is to define at a strategic level how the services that are to be outsourced will be rolled out. This will document how key decisions will be made later on, by defining strategies on staffing, communication, program roll-out and risk assessment.

Outputs:
- Program Roll-out strategy
- Communication strategy
- Strategy to manage staff impacted by the migration to cloud

4) Business Design
This step involves designing what is to be outsourced to the cloud and what the future state will look like. It will detail the new service, how it will be managed, how it interfaces to the existing/remaining systems, and how it will be monitored and reported. It exists to provide requirements with sufficient detail to have a meaningful conversation with suppliers so that they can be objectively compared, based on cost and quality of service.

Outputs:
External
- Contact template
- Service Level Agreement (SLA)
- Pricing model

Internal
- The future Enterprise Architecture with support and technical interfaces
- How the contract negotiations will be managed
- How the supplier will be managed

Phase 2: Engagement

5) Selection
Based on the requirements and the other criteria defined by the Architect phase, this step will select the best supplier based on value, sustainability and quality.

Outputs:
- Tender process
- Evaluation criteria
- Short-list of suitable suppliers with caveats
- Due diligence report

6) Negotiation and Sign-off
The purpose of this step is to pick the preferred supplier(s), complete the final negotiation, get internal approval and then sign the contract.

Outputs:
- Negotiation strategy
- Results of the negotiation
- Signed final documents: Contract, SLA and Pricing document

Phase 3: Operate

7) Operational Roll-out
This step involves putting together a transition project team that will manage the transition of the agreed services to the new cloud environment. This will require the transition of the service itself, the management of staff impacted, communication to all stakeholders, knowledge retention / transition and acceptance sign-off.

Outputs:
- Roll-out plan
- Progress updates
- Signed acceptance document

8) Management of the Supply Chain
The aim of this step is to manage the new environment as efficiently and effectively as possible. The organisation will need to adapt to the new setup particularly at management level, rather than directly managing internal resources. The requirement will be to manage the supplier and in particular the supplier relationship. This will require effective monitoring and control so that issues, variations and disputes can be resolved to both parties satisfaction.

Outputs:
- Day to day performance metrics
- Status on issues, problems, variations and disputes
- Supplier meeting minutes
- Change management report
- Audit reports

Phase 4: Regenerate

9) Review
This step is important to review the service based on requirements of the service itself, other changes within the business, changes within the supplier organisation or the need to change supplier.

 ISBN: 978-1-61208-153-3

Outputs:

- Intelligence report for next generation options
- Supplier audit results
- Business case for any proposed changes

B. Influence of the Supply Chain

The lifecycle is intended to provide an organisation with a management structure to assess the following:

1) The readiness/maturity of an organisation to move to the cloud.
2) Once they are migrated, assess how the organisation is managing the new environment on a day-to-day basis.
3) Assess what new services can be moved to a cloud environment.

The lifecycle interfaces with the supply chain in a number of ways as follows:

- In the Architect phase when deciding what services to move to the cloud and what suppliers can provide, this will set the scene on what is technically viable to move to the cloud.
- The Engage phase will determine what supplier will be used and if they can deliver the requested services to the required levels of reliability and quality.
- The Operate phase is the most critical as the chosen service(s) will be migrated and then become the responsibility of the chosen supplier. If the lifecycle is used correctly this phase should run smoothly, otherwise either the migration will fail or once migrated the service will not be at the required levels to support the business.
- The final Regenerate phase will assess the current supplier and cater for the migration of new services.

In summary the lifecycle and the supply chain are intrinsically linked and for the lifecycle to be successful there is a dependency on a fully functioning, flexible and robust supply chain.

IV. Conclusion and Future Work

In conclusion, the paper analyzed the requirements that need to be considered for migrating from a traditional IT environment to the cloud from a supply chain approach. The paper looked at the area of cloud computing in relation to organisations and the various benefits and problems associated with this. The supply chain was explained as well as the different types of cloud and different service models within cloud computing. The cloud supply chain was illustrated through the diagram as well as an explanation of the various actors and components within it and how these interact. The paper focused on the structural setup of the supply chain and how it is composed and considered the possible complexity of the supply chain taking into consideration the number of clouds and services used at once within it. This introduced the area of cloud bursting and showed the benefits from both a consumer and provider perspective of this process. The management of the migration process is described through the use of a cloud lifecycle. Each of the steps within the lifecycle were identified and the steps that were influenced by the cloud supply chain were discussed. Overall, by using the cloud supply chain, the technical requirements for a move to cloud can be identified and the cloud lifecycle can be used to manage the migration and the ongoing improvement of the cloud environment.

Future work includes the assessment of the cloud lifecycle process to measure how effective it is in helping organisations move to the cloud. This would allow for further improvements to the cycle and possibly lead to a more efficient migration process. In relation to the supply chain, cloud supply chain management and controlling would be a future topic of interest. This would consist of the management from the cloud service provider to the end-cloud consumer including all of the components across the supply chain such as, cloud services, information and funds.

V. Acknowledgements

The work presented in this paper has been funded in part by Enterprise Ireland under Grant CC/2009/0801 and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

[1] G. Conway "Introduction to Cloud Computing", White Paper, Innovation Value Institute, Jan 2011 http://ivi.nuim.ie/publications/IVI-Exec-Briefing-Intro-to-Cloud-Computing.pdf, 28.06.2011

[2] M. A. Lindner et al., "Cloud supply chain: A framework for information, monitoring, accounting and billing," CloudComp 2010, Springer Verlag, 2010

[3] S. Cullen, P. Seddon, and L. Willcocks, "Managing outsourcing: The life cycle imperative," MIS Quarterly Executive Vol. 4, No.1, Mar 2005

[4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition, ACM SIGCOMM, Vol 39 No.1 Jan 2009

[5] Ca technologies, "Can you harness clouds without creating storms," Nov 2010, http://www.ithound.com/abstract/harness-clouds-creating-storms-5720, 28.06.2011

[6] S. Somashekar, "Opportunities for the cloud in the enterprise,"' Jan 2010, http://www.ca.com/es/whitepapers/collateral.aspx?cid=230175, 28.06.2011

 ISBN: 978-1-61208-153-3

A Service-Level Agreement Approach Towards Termination Analysis of Service-Oriented Systems

Mandy Weißbach and Wolf Zimmermann
Institute of Computer Science, University of Halle
06120 Halle (Saale), Germany
Email: {weissbach, zimmermann}@informatik.uni-halle.de

***Abstract*—Classical approaches for program analysis as, e.g., termination analysis usually do not take into account modern software approaches such as service-oriented systems or cloud computing. Instead, they have a monolithic view on the software system as a single completely available program. As first step to enable such analyses also in a service-oriented or cloud computing context, respectively, this paper considers termination. Since termination is a service quality attribute, we consider a service-level agreement approach that allows dynamic bindings to software services. In contrast to many other service-level agreements, termination is a binary attribute that cannot be measured quantitatively (as, e.g., reliability or response time). The proposed approach shows how clients of services can verify the information provided by the services.**

Keywords-Termination; Software Services; Service Level Agreement; Verification.

I. Introduction and Motivation

The vision of cloud computing is (among others) that there are numerous software services in the cloud that can be used by clients to fullfil their functionality. These services are functionally equivalent in the sense of the context of the client. However, they might differ in their non-functional properties. Thus clients may negotiate service level agreements on non-functional properties such as, e.g., reliability, availability, response times, etc. The literature on service-oriented computing and cloud computing offers already numerous techniques that clients may monitor these quality attributes, see, e.g., [1] for an overview. However, there are other service quality attributes such as, e.g., termination of the services and/or the client, robustness (i.e., neither the service nor the client aborts due to uncaught exceptions), or absence of deadlocks. In contrast to the above mentioned service quality attributes, these attributes have a binary character: either the services or clients satisfy the quality attribute or not. In this work, we consider in particular the termination of software services and the clients using software services. Petri-Net based approaches towards deadlock analysis are usually based on termination of the services [2]–[4].

Remark: At first glance, it seems that there is no need for a termination analysis in service-oriented systems (except possibly for deadlock analysis) because one might think that after a certain time the client might switch to another, functionally equivalent, service in the cloud. However, there are situations where this approach cannot be applied. First, the approach doesn't work if none of the functionally equivalent services has (for the client) satisfactory quality attributes. In this case the chosen service becomes the single candidate and its termination is an important property for the client. A second reason is the choice of the time period: If the period is fixed to just a few seconds or minutes, this might be a reasonable approach. It might work well in business applications. However, in scientific computing or bioinformatics there are computation intensive applications and if these are offered as services they might run for hours or days. Fixing the time period to a few seconds or minutes implies that any functionally equivalent service fails to be finished within this time. On the other hand, it doesn't make sense to switch after a few hours or days to another service that possibly requires even more execution time than the originally chosen service. Thus, in these situations it is better to know that the service terminates and will deliver an answer. Third, a termination proof for clients may require information on the effect on results of services being called, e.g., their size. This size change information must also be included in the analysis and has a rather different character than simple termination. □

The techniques that enable the clients to check whether service-level agreements are obeyed cannot be applied in the context of binary quality attributes. Consider for example termination: if a client has not yet a response from a service, the client cannot conclude that the service doesn't terminate. The service might respond within the next second. On the other hand, the client cannot reason on its own termination behaviour without provision of adequate information from the services. This information must be correct. Thus, the challenge is how the client can verify that the requested information is correct. The situation becomes even more difficult if a client uses a service A and the service A uses a service B, etc. In this case termination of the client may indirectly depend on service B and service A needs to request information on termination of B as well as additional information to prove its termination.

In this paper, we assume that there are no recursive call-backs, i.e., recursion over service boundaries are excluded. Furthermore, in order to use well-known termination analysis approaches, we exclude service-internal parallelism since this is still an open issue in classical termination analysis. Thus, we tackle in this paper termination analysis of service-oriented systems in dynamically changing environments where recursive call-backs and service-internal parallelism is being excluded.

 ISBN: 978-1-61208-153-3

The paper is organized as follows: Section II introduces into classical termination analysis. The following Section III shows how this approach can be extended in a service-oriented or cloud computing context, respectively, using a service-level agreement approach. In Section IV, we show how clients can verify the results by combining the approach of Section III with approaches used for verification of Web pages. Section V discusses related work and Section VI concludes this work.

II. Termination Analysis

Although termination of programs is undecidable (known as halting problem), there is a lot of work on *conservative* analysis of program termination. A conservative termination analysis guaruantees termination in the case of a positive answer. However, a negative answer may be false. It should be interpreted as *termination cannot be proven*. Thus, termination analysis does not implement the halting problem but it only provides a one-sided solution (similar to model checking) . The following discussion shows that there are a number of works (it just mentions the most important ones) on termination analysis. Each of them is conservative and assume that the whole program to be analyzed is available to them.

The first works on termination analysis or the related field of automatic complexity analysis go back to [5] for pureLisp programs. This was generalized to first-order functional languages [6] and to object-oriented imperative programs [7], [8]. Works on automatic complexity analysis as well as on termination analysis are based on the notion of a termination function. This is a function from program states to natural numbers that strictly decreases when executing the body of loop or when a procedure is called recursively. Since there is no infinite descending chain in the natural numbers, a termination function ensures loop or recursion termination, respectively. More recent work on termination analysis focuses on automatic derivation of termination functions, which is often called the size-change principle, cf. [9]–[12]. Instead introducing into these methods, we informally demonstrate termination analysis by the example in Figure 1. In a service-oriented architecture the four classes will be considered as services (implemented by web services, cf. Figure 2. `Calendar` contains to public procedures `first()` and `next(Month month)` which together can be used to iterate over all 12 months of a year. The class `List` is a classical list implementation with a sentinel `empty`. `MSales` has access to a customer database. The procedure `sales(Month month, Year year)` uses this customer database to calculate the sales of month `month` in year `year`. Procedure `sales(Year year)` calculates the sales of year `year` by summing up the sales of all months of `year`.

Suppose the termination of procedure `YSales.sales` has to be analyzed. Note that all the steps (except possibly the provision of terminations functions which have to annotated) can be performed automatically according to the above mentioned works.

```
class YSales {
  private Msales msales;
  public int sales(Year year) {
    Month month=Calendar.first();
    int sum=0;
    while (month!=Month.complete) {
      int amount=msales.sales(month,year);
      sum += amount;
      month=Calendar.next(month);
    }
    return sum;
  }
}
class Calendar {
  public Month first() { return Month.jan; }
  public Month next(Month m) {
    if (m==Month.jan) return Month.feb;
    ...
    if (m==Month.dec) return Month.complete
  }
}
class MSales {
  private static CustomerDatabase db;
  public int sales(Month month,Year year) {
    List cl=db.getCustomers(month,year);
    int sum=0;
    while (cl!=List.empty) {
      int amount=cl.hd();
      sum += amount;
      cl=cl.tl();
    }
    return sum;
  }
}
class List {
  private int head;
  private List tail;
  static List empty=new List();
  public int hd() { return head; }
  public List tl() {
    return (tail==NULL?empty:tail);
  }
}
```

Figure 1. Sales-Example

Step 1 Analyze each non-recursively called procedure for termination:

Since this procedure calls procedures `MSales.sales`, `Calendar.first`, and `Calendar.next`, these procedures have to be analyzed for termination.

Step 2 Analyze each loop and each recursively called procedure for termination by deriving/introducing an adequate termination function:

The loop termination of the loop in `YSales.sales` apparantly depends on the variable `month`. The termination function φ defined by

$$\varphi(\texttt{month}) \triangleq 13 - sz(\texttt{month}) \qquad (1)$$

where $sz(\texttt{month})$ is the number of the month (i.e., $sz(\texttt{Jan}) = 1, sz(\texttt{Feb} = 2)$, etc. and $sz(\texttt{complete}) = 13$) proves termination. This is because

$$\varphi(\texttt{next}(\texttt{month}))) = \varphi(\texttt{month}) - 1 \qquad (2)$$

(2) can be derived by determining a size change function for `next` with the notion of size defined by (1), i.e., a function $\varphi\_\texttt{next} : \mathbb{N} \rightarrow \mathbb{N}$ such that $\varphi\_\texttt{next}(\varphi(\texttt{month})) = \varphi(\texttt{next}(\texttt{month})))$.

Step 3 Determine the necessary size change functions for procedures:

By a simple case analysis it can be determined that $\varphi\_\texttt{next}$ is defined by $\varphi\_\texttt{next}(n) = n - 1$ thereby proving that the termination function φ decreases by 1 during loop termination.

 ISBN: 978-1-61208-153-3

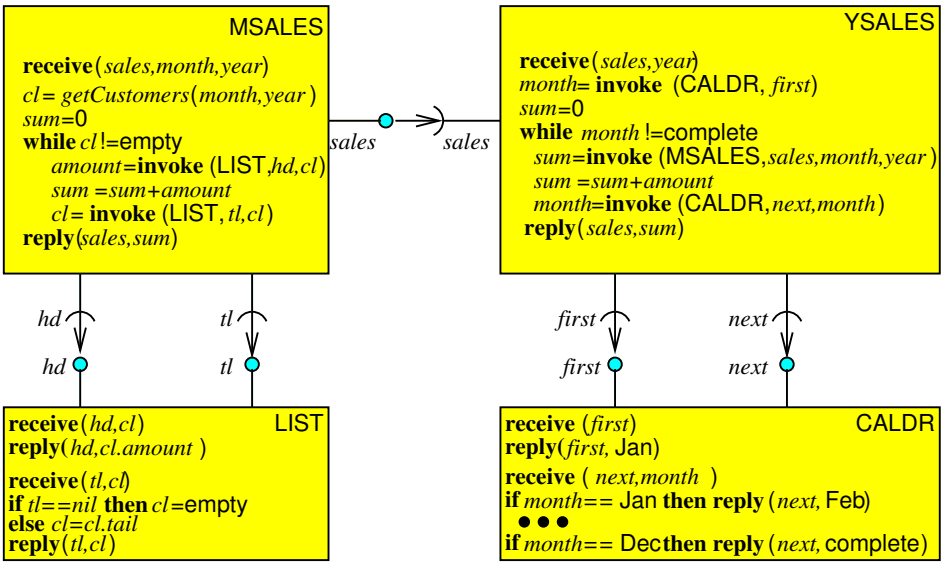


Figure 2. A Service-Oriented Architecture for the Sales Example

The termination of `Calendar.first` and `Calendar.next` can be derived directly as they neither contain a loop nor a procedure call. The termination analysis for `MSales.sales` must follow the same approach as mentioned above:

Step 1: The procedure `MSales.sales` calls procedure `CustumerDatabase.getCustomers`, `List.hd`, and `List.tl`. The latter two terminate since they neither call a procedure nor contain a loop. The former terminates since it executes a database query (not shown in Figure 1).

Step 2: The termination of the loop in `MSales.sales` depends on the length of the list `cl`, i.e., the termination function is recursively defined by

$$\psi(\texttt{cl}) \triangleq \begin{cases} 0 & \text{if } \texttt{cl} = \texttt{NULL} \\ 1 + \psi(\texttt{cl.tail}) & \text{otherwise} \end{cases} \quad (3)$$

This termination function requires the determination of the size change function $\psi\_\texttt{tl} : \mathbb{N} \to \mathbb{N}$ such that

$$\psi\_\texttt{tl}(\psi(\texttt{cl})) = \psi(\texttt{tl}(\texttt{cl})) \quad (4)$$

Step 3: The analysis yields that $\psi\_\texttt{tl}(n) = n - 1$ which completes the proof of termination of the loop in `MSales.sales`

In a nutshell, the termination argument for `YSales.sales` is as follows:

- `YSales.sales` terminates because each procedure called in the body terminates and the loop terminates
- The loop terminates because (1) is a termination function
- φ is a termination function because of (2) which proves that φ strictly decreases after executing the loop body

The steps presented in this section can be formalized as proof rules (see [14] for a short summary). These rules are usually the formal basis for the correctness of termination analysis. If the proof succeeds the program terminates. However, a program may terminate althoug a termination analysis cannot find a proof.

III. An SLA approach for Termination Analysis

The goal of this section is to apply the approach of Section II in a service-oriented context. It is demonstrated by the service-oriented architecture shown in Figure 2 which corresponds to the example in Figure 1 and is implemented by three web services MSALES (with interface *sales*), LIST (with interfaces *hd* and *tl*), CALDR (with interfaces *first* and *next*), and a client YSALES.

Note that the implementations of the web services are not known to their clients. Thus, a termination analysis cannot directly follow the approach as described in Section II. In particular, Step 1 cannot analyze the termination of services being called but it must rely on the information of the termination provided by the called service. For example the invocation of the services CALDR.*first*, CALDR.*next*, and MSALES.*sales* require termination, and the providing web services must know this information. Note that the client YSALES is not aware of the fact that the termination of MSALES.*sales* depends on the termination of LIST.*hd* and LIST.*tl*. Furthermore, in order to proof termination of the loop in YSALES, the service CALDR must provide a strictly decreasing size change function for *next*. The decision whether it must be decreasing or increasing, or how fast it must be decreasing or increasing for proving termination depends on the loop body.

Thus, in a service-oriented setting, a client needs to have the following infomation when it analyzes its termination:

- The information on the termination of each service called

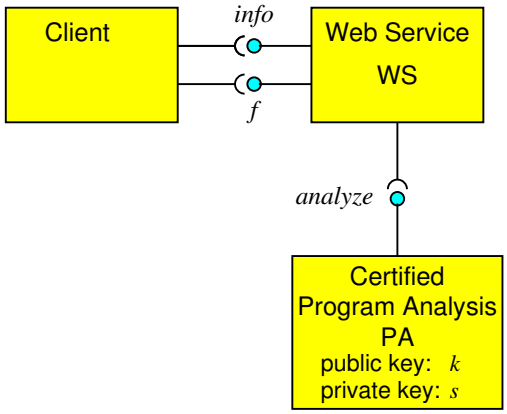


Figure 3. Verifiable Program Analysis Information

- Adequate context-dependent size change functions for those services whose calls influence termination of loops or recursive calls.

While the first information can be provided by the web service providing a called service, the latter must be individually requested by the client while analyzing the termination behaviour of the client. In both cases, the client relies on the correctness of the information provided by the web service.

IV. Certification of Informations Provided by Software Services

A problem with the approach in Section III is the validity of the information on termination of services as well as the validity of the size change function. In contrast to quantitative properties such as, e.g., reliability, availability, or response time, the client has no possibility to check a service level such as termination or the validity of size change functions. We first present an approach that considers basic web services, i.e., they don't use other Web Services. Then, we extend the approach to web services using other Web Services where the use-relation is acyclic.

A. Basic Web Services

Figure 3 shows an approach that may solve this problem. First, a certified program analysis service PA is needed for the analysis of the web service. Second, a public-key infrastructure is needed for enabling the client to verify the results of the analysis. The program analysis service PA must be known to the client as a certified analysis tool. With this infrastructure, a termination/program analysis as discussed in Section III can be implemented such that the client can verify the results from the web service WS:

Step 1: The client requests from web service WS via the service *info* information on the termination or size change of f (as discussed in Section III).

Step 2: Web service WS encrypts its source text with the public key of the certified program analysis service PA and sends it via the interface *analyze* together with the requested analysis to PA.

Step 3: The certified program analysis service PA decrypts the source text of WS with its private key s, performs the requested program analysis, signs the result with its private key s, and returns the signed result to WS.

Step 4: Web service WS returns the signed result to the client together with the public key k of the certified program analysis service PA.

Step 5: The client can decrypt the information with the key k and since the key k is unambiguous, it can verify that the information is obtained by the certified program analysis service PA.

With this approach, the client can verify that the certified program analysis performed its analysis. The encryption in Step 2 is needed because implementers of web services don't want to publish their implementation. With the encryption, the source text is only available to the certified program analysis service PA.

For this approach, the trusted base is certainly the certified program analysis service PA. However, it is not guaranteed that PA really analyzes the source text of WS. A malicious web service WS might send another source text whose analysis results errorneously indicate the client termination or provides an adequate size change function. Currently, we are not aware of a technology that ensures that the WS sends the correct source text to PA.

However, it is possible to make it more difficult for WS to be malicious by keeping the analysis request secret to WS. This can be achieved changing the protocol of the SLA: The client first notifies WS that it wants to perform a program analysis. Then WS returns a public key k of a certified program analysis service PA. The client can use k to verify that PA is indeed certified. Finally, the analysis request is encrypted with k. The above implementation needs only to be changed at Step 3 where the analysis request must be decrypted. If we trust PA and the public key infrastructure, then it is impossible for WS to decrypt the analysis request.

B. Composed Web Services

The approach in Section IV-A doesn't consider the situation as shown in Figure 4. Web service WS_1 uses as a client web service WS_2 and the client is not aware of this usage. Thus, the termination analysis (or other program analyzes) of WS_1 requires the analysis of WS_2 (including possibly the analysis of size change functions).

For the termination analysis or the analysis of size change functions of WS_1's service, web service WS_1 acts as a client of web service WS_2. Hence WS_1 negotiates termination and size change functions with WS_2 as described in Section IV-A. However, this information is needed by the certified program analysis. For example, if the program analysis requires for the termination of f information on the termination of g or a size change function for g where g is an external service call of WS_1, then this information is passed to WS_1 via the interface *painfo* (encrypted with the public key k_1 of WS_1 for security reasons). Service WS_1 decrypts the analysis request and passes it as described in Section IV-A to WS_2.

 ISBN: 978-1-61208-153-3

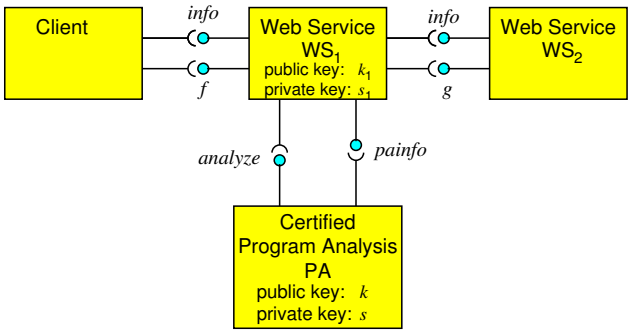


Figure 4. Program Analysis on Composed Web Services

Then, WS_2 returns the information on termination of g or the requested size change function for g, respectively. In contrast to the approach in Section IV-A, the result is not decrypted and verified by WS_1. Instead it is passed to the certified program analysis (as the return value of the service *info* of WS_2) and the certified program analysis verifies whether the analysis results for g can be trusted. Note, that this approach does not require that WS_2 uses the same program analysis service as WS_1.

Apparently, with this approach WS_2 may use a web service WS_3, etc. However, the approach is limited to acyclic architectures. Otherwise, the termination analysis itsself would run into an infinite loop which practically would have the same effect as a Denial-Of-Service attack to the services.

V. Related Work

There is a need for program analysis of service-oriented systems. Canfora, *et al.* [13] states it as a key challenge for software reverse engineering. Currently, there are not many works on program analysis of service-oriented systems – in particular we are not aware of any work on termination analysis of service-oriented systems except [14]. This work is based on interface descriptions of web services containing termination information and size change function. Furthermore, it doesn't verify the information provided by the interface descriptions.

One of the few works considering program analysis is [15], [16]. They consider response time in terms of some notion of input size. Information on response time is provided by the web service interfaces. Their approach generalizes the approach of [17] for the analysis of software complexity of BPEL processes towards response time. For invocations of other services [15], [16] use the information provided by the corresponding service descriptions. However, they don't verify this information and it seems that size change functions play no role in their approach.

For functional verification of web service contracts, [18] discusses a similar approach using a public key infrastructure. Apparently, contracts should be part of web service interface descriptions and are not part of service-level agreements. In contrast to our approach, they require that the analyzers are located on the same machine as the service implementations, respectively. This one-platform approach allows to take into account the operating system and the compiler.

VI. Conclusion and Future Work

This paper has presented a termination analysis of service-oriented systems in a dynamic changing environment. This goal was achieved by using an SLA approach. It was shown that the service has to provide two kinds of informations: its termination and size change functions requested by the client which enables the client to prove its termination. In contrast to quantitative service qualities, these informations cannot be verified by the client. Therefore, a certification process similar to the verification of web pages has been added in order to ensure that the information has been derived from certified tools.

One property of the approach is the violation of the black-box paradigm of services because they must offer their source to a program analysis service. However, we consider such program analysis services as a trusted institutions (analogous to institutions certifying web pages). In any case, the clients never see implementation details of the used services.

Our approach may be used for the analysis of other binary quality attributes which can be verified by program analyses or model checking approaches. Currently, it excludes cycles in the architecture, i.e., there are no recursive call-backs. Such cycles would lead to an infinite loop while negotiating the service-level agreement. We also assume that the services have no internal parallelism. The next steps will be to drop these assumptions and to consider other binary quality attributes.

Another challenge is to prevent malicious analysis results from the web service to be analyzed. As pointed out in Section IV, a web service may send the wrong source text to the program analysis service. We have presented an approach that keeps the requested analysis secret to the web service but this only makes it more difficult to the web service to cheat. A secure approach must enable the client to verify that the source text given to the program analysis service is identical to the source text of the web service. A possible solution might be that the web service signs its source text with its digital signature when sending it to the program analysis. In this case, at least liability is possible if the wrong source text was sent.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] K. Candan, W. Li, T. Phan, and M. Zhou, "Frontiers in Information and Software as Services," in *IEEE International Conference on Data Engineering*. IEEE, 2009, pp. 1761–1768.

[2] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *The Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

 ISBN: 978-1-61208-153-3

[3] W. M. P. van der Aalst, A. P. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski, "Advanced workflow patterns," in *CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems*. London, UK: Springer-Verlag, 2000, pp. 18–29.

[4] W. Reisig, "Modeling- and analysis techniques for web services and business processes," in *In FMOODS*, 2005, pp. 243–258.

[5] B. Wegbreit, "Mechanical program analysis," *Communications of the ACM*, vol. 18, no. 9, pp. 528 – 539, 1975.

[6] W. Zimmermann, *Automatische Komplexitätsanalyse funktionaler Programme*, ser. Informatik-Fachberichte. Springer, 1990.

[7] H. Schmidt and W. Zimmermann, "Reasoning about complexity of object-oriented programs," in *Programming Concepts, Methods and Calculi*, ser. IFIP Transactions, E.-R. Olderog, Ed., vol. A–56, 1994, pp. 553–572.

[8] H. Schmidt and W. Zimmermann, "A complexity calculus for object-oriented programs," *Journal of Object-Oriented Systems*, vol. 1, no. 2, pp. 117–147, 1994.

[9] A. M. Ben-Amram and C. S. Lee, "Program termination analysis in polynomial time," *ACM Transactions on Programming Languages and Systems*, vol. 29, pp. 5:1–5:37, January 2007.

[10] A. M. Ben-Amram, "Size-change termination, monotonicity constraints and ranking functions," in *Proceedings of the 21st International Conference on Computer Aided Verification*, ser. CAV '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 109–123.

[11] M. Codish, C. Fuhs, J. Giesl, and P. Schneider-Kamp, "Lazy abstraction for size-change termination," in *Proceedings of the 17th international conference on Logic for programming, artificial intelligence, and reasoning*, ser. LPAR'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 217–232.

[12] F. Spoto, F. Mesnard, and E. Payet, "A termination analyzer for java bytecode based on path-length," *ACM Transactions on Programming Languages and Systems*, vol. 32, pp. 8:1–8:70, March 2010.

[13] G. Canfora, M. Di Penta, and L. Cerulo, "Achievements and challenges in software reverse engineering," *Commun. ACM*, vol. 54, pp. 142–151, April 2011.

[14] M. Weißbach and W. Zimmermann, "Termination analysis of business process workflows," in *Proceedings of the 5th International Workshop on Enhanced Web Service Technologies*, ser. WEWST '10. New York, NY, USA: ACM, 2010, pp. 18–25.

[15] D. Ivanovic, M. Carro, and M. Hermenegildo, "An initial proposal for data-aware resource analysis of orchestrations with applications to predictive monitoring," in *Proceedings of the 2nd Workshop on Monitoring, Adaptation and Beyond (MONA+), Lecture Notes in Computer Science. Springer*, 2010.

[16] D. Ivanovic, M. Carro, and M. Hermenegildo, "Towards Data-Aware QoS-driven Adaptation for Service Orchestrations," in *2010 IEEE International Conference on Web Services*. IEEE, 2010, pp. 107–114.

[17] J. Cardoso, "Complexity analysis of BPEL web processes," *Software Process Improvement and Practice*, vol. 12, no. 1, pp. 35–49, 2007.

[18] J. Lyle, "Trustable remote verification of web services," *Trusted Computing*, pp. 153–168, 2009.

 ISBN: 978-1-61208-153-3

Cloud Federation

Tobias Kurze*, Markus Klems†, David Bermbach†, Alexander Lenk‡, Stefan Tai† and Marcel Kunze*
*Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
Email: {kurze, marcel.kunze}@kit.edu
†Institute of Applied Informatics and Formal Description Methods (AIFB)
Karlsruhe Institute of Technology (KIT), Kaiserstrasse 12, 76131 Karlsruhe, Germany
Email: {markus.klems, david.bermbach, stefan.tai}@kit.edu
‡FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
Email: {lenk}@fzi.de

***Abstract*—This paper suggests a definition of the term *Cloud Federation*, a concept of service aggregation characterized by interoperability features, which addresses the economic problems of vendor lock-in and provider integration. Furthermore, it approaches challenges like performance and disaster-recovery through methods such as co-location and geographic distribution. The concept of Cloud Federation enables further reduction of costs due to partial outsourcing to more cost-efficient regions, may satisfy security requirements through techniques like fragmentation and provides new prospects in terms of legal aspects. Based on this concept, we discuss a reference architecture that enables new service models by horizontal and vertical integration. The definition along with the reference architecture serves as a common vocabulary for discussions and suggests a template for creating value-added software solutions.**

***Index Terms*—Cloud Computing, Cloud Federation, Reference Architecture, Lock-In, Hold-Up, Integration**

I. Introduction

The Cloud Computing paradigm advocates centralized control over resources in interconnected data centers under the administration of a single service provider. This approach offers economic benefits due to supply-side economies of scale, reduced variance of resource utilization by demand aggregation, as well as reduced information technology (IT) management cost per user due to multi-tenancy architecture [1].

These benefits have contributed to the increasing industry acceptance of Cloud services, which are seen as more affordable and reliable alternatives compared to traditional in-house IT systems and services. However, downsides of the Cloud Computing paradigm are surfacing. Surveys show that potential customers hesitate to outsource their business applications and data into the cloud [2]. Besides security concerns, application users are afraid of loosing ownership and control. The lack of standardized service interfaces, protocols and data formats is a portent of vendor lock-in [3]. This problem can lead to underinvestment, an economically inefficient situation, and therefore deserves our attention.

We propose an extended concept of *Cloud Federation* to enable the design of flexible and interoperable Cloud-based software, thereby lowering the adverse effects of vendor lock-in. We further discuss Cloud Federation as a key concept allowing the development of new types of applications.

The paper is structured as follows: Section II provides an overview of the state of the art on Cloud Stack and describes economic problems related to Cloud Computing. In Section III we state a definition of the term Cloud Federation and explain the concept in detail. Section IV introduces our vision of a reference architecture for federated Clouds. Finally, we give thought to open issues in Section V before concluding in Section VI.

II. Background and Related Work

Cloud Computing distinguishes the service models Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [4] [5]. IaaS offers infrastructure services, such as Compute Clouds, Cloud Storage, Message Queues, etc. PaaS offers complete platforms, solution stacks and execution environments, while SaaS is a software delivery model driven by a multi-tenancy architecture.

A. Cloud Stack

The principal service models IaaS, PaaS and SaaS do relate to one another and can be arranged as a stack. The IaaS layer represents the lowest level of the stack and is very close to the underlying hardware. Inside the IaaS layer two types of services can be differentiated: computational and storage [5]. Typical representatives for infrastructure services are Amazon's EC2 and Amazon's S3 (Appendix: Table A).

PaaS represents the second layer in the stack. Famous examples are Microsoft's Azure, Google's App Engine, SalesForce' Force.com and Amazon's Elastic Beanstalk (Appendix: Table A). Elastic Beanstalk is currently in beta phase and directly based on Amazon's IaaS offerings.

Upper layers such as SaaS (e.g., Google Docs) and Human as a Service (HuaaS) are directly or indirectly based on either IaaS or PaaS. Some secondary services, such as monitoring, accounting, authentication, metering or configuration and management are needed on multiple levels of the stack.

 ISBN: 978-1-61208-153-3

B. Cloud Software and Cloud Products

1) Private Cloud Computing Software: There is a broad spectrum of open source software, which mimics the proprietary systems of Amazon, Google, & Co. For example, Eucalyptus, AppScale, typhoonAE, and OpenNebula (Appendix: Table A). Users can install the open source software "in-house" as *private cloud* solutions. Since such a private cloud solution is partially compatible with the interfaces, protocols, programming models, and deployment options of the proprietary public clouds, this might be an approach to create an interoperable *hybrid cloud*, a composition of private and public clouds [6].

2) Cloud Marketplaces and Federation Offerings: While marketplaces, like Zimory or SpotCloud allow trading with Cloud resources, offerings like CloudKick and ScaleUp provide some federation functionality, e.g., monitoring and management supporting multiple clouds (Appendix: Table A).

C. Economic Theory

1) Vendor Lock-in: Vendor lock-in has been studied in economics research communities, for example by Robin Cowan [7]. Cowan identifies two sources of vendor lock-in: uncertainty of selecting an unknown technology, and the learning curve of a technology. The problem with two technologies A and B is formalized as the dynamic programming problem "Two-armed bandit".

We observe a growing number of Cloud Computing service providers and service offerings, in particular Cloud Storage and Compute services. These offerings tie users to a specific technology, which cannot be switched or replaced without significant switching cost. Apparently, this is the case for PaaS offerings, e.g., Google App Engine, which are closely integrated with proprietary services, such as Google user accounts and the Google e-mail service. Offerings like Amazon Web Services seem to have lower switching costs because they build upon Web service standards. However, a competing service provider would have to provide a similar technology (distributed system) with similar quality levels (availability, reliability, latency, throughput, etc.) and features (launch, stop, start, etc.).

This leads to the consequence, that users depend on the business strategy of the service provider.

2) Hold-up Problem and Underinvestment: The hold-up problem has been described by Klein, Crawford and Alchian [8] as being basically a contract problem. Two firms want to start business relations. In order to do so one party has to make an investment, which is specific in regard to the other party. Transferred to a concrete Cloud scenario, a company could invest in developers and applications, which are using Amazon's Web Services. This particular investment is of virtually no use when not used in the context of the two parties, i.e., the applications can not be used with Google's App Engine for example nor can the spezialized developers work with Microsoft's Azure. It is not possible to write complete contracts, i.e., contracts containing all, even future aspects of business relations, which might have an influence on the returns from the investment [9]. Due to incomplete contracts it is very likely that situations will arise that have not been foreseen at the time of the contract writing, making renegotiations necessary. In such future interactions one party may take advantage of the lock-in situation.

A party, anticipating the risk of a lock-in situation, typically takes suboptimal investment decisions, leading to underinvestment. [10, 11] When already facing the lock-in problem, a company may decide to stop further investment or to expend resources to protect itself against the lock-in. A party anticipating lock-in, hence, ends up in a hold-up situation, which in either case, leads to inefficient results [9].

Ewerhart et al. [12] summarized that in a lock-in situation, market forces are no longer effective and there is a risk of ex-post opportunistic behaviour. A party being forced to accept sub-optimal conditions cannot escape the situation due to the lock-in and finds itself in a hold-up [13].

III. Cloud Federation

Cloud federation comprises services from different providers aggregated in a single pool supporting three basic interoperability features - resource migration, resource redundancy and combination of complementary resources resp. services. Migration allows the relocation of resources, such as virtual machine images, data items, source code, etc. from one service domain to another domain. While redundancy allows concurrent usage of similar service features in different domains, combinations of complementary resources and services allows combining different types to aggregated services. Service disaggregation is closely linked to Cloud Federation as federation eases and advocates the modularization of services in order to provide a more efficient and flexible overall system.

We identify two basic dimensions of Cloud Federation: horizontal, and vertical. While horizontal federation takes place on one level of the Cloud Stack, e.g., the application stack, vertical federation spans multiple levels. In the following we focus on horizontal federation; aspects of vertical federation are out of the scope of this publication.

Several aspects of horizontal federation can be distinguished, e.g., provider domain and geography. Horizontal federation across provider domains may decrease provider dependency and thereby lower the risks of vendor lock-in and hold-up. Increased availability may be achieved through horizontal federation across multiple geographic regions. Also, vertical federation scenarios along similar aspects are imaginable.

Cloud Federation can be of interest for providers as well as for customers. Customers may profit from lower costs and better performance, while providers may offer more sophisticated services. However, hereinafter we focus on the customer perspective.

Two types of scenarios can be linked to Horizontal Federation:

 ISBN: 978-1-61208-153-3

- Redundancy: is used whenever there is a subset of (properly organized) service offerings that provide better utility to a client than any single service offering x_i, i.e., $\exists X \subseteq \bigcup_i x_i$ where $\forall x_i : u(X) > x_i$. the duration is, at least regarding a near time horizon, permanent as the user purposefully uses multiple service providers at the same time.
- Migration: can be triggered when a new service offering offers better utility to a client than any previously used service offering, i.e., $\exists x_{new} \forall x_i : u(x_{new}) > u(x_i) + u(c_s)$ where c_s are the total switching costs and $x_{new} \notin \bigcup_i x_i$.

Figure 1 illustrates the behavior over time of the two scenarios.

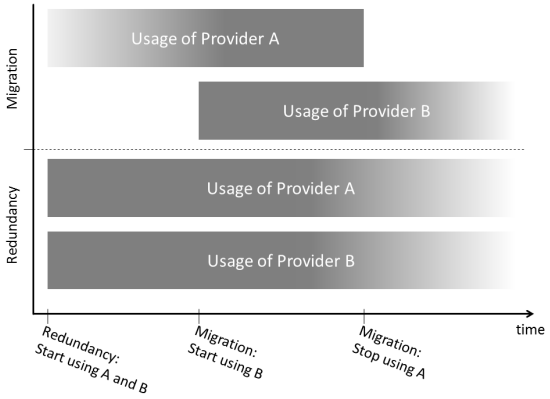


Fig. 1. Migration vs. Redundancy

A. Redundancy

Following the technical Cloud Stack [5], we can distinguish IaaS, PaaS and SaaS as different levels where horizontal redundancy can be used.

1) IaaS:

a) Compute services: know 3 kinds of redundancy:

- Redundant deployment: The same application logic is deployed to different providers. Still, incoming requests are processed by only one instance. Redundant deployment is used to increase the availability while decreasing provider dependence. Other reasons to do so could be compliance with regulations, which require instances in particular geographic locations. Also, customer proximity could be an issue to reduce latency.
- Redundant computation: The same application logic is deployed to different providers. Nevertheless, in contrast to redundant deployment here every request is processed by more than one instance. Reasons to do so could be either to improve performance by reducing the risk of an instance failing right before completing a task, an approach, e.g., taken in Google's MapReduce [14], or limited trust in the provider returning correct results.
- Parallel computation: Here, the data is broken down at bit level and processed at different providers' sites following the same application logic or complimentary services are deployed to different providers. Reasons for the 1st case could be security considerations where each provider only knows a tiny subset of the data. In the 2nd case, tasks are spread to the best fitting VMs to optimize latency and throughput.

b) Storage Services: know 3 kinds of redundancy:

- Replication: Data items are distributed as a whole and multiple copies are stored to increase availability while removing a single point of failure [15, 16, 17, 3] and reducing vendor lock-in. Furthermore, an increased number of replica may improve read latency due to customer proximity and increases durability. This is especially of interest when addressing resilience to correlated failures. However, whenever copies of the same data are kept at different sites there is a general tradeoff between consistency and availability as well as latency depending on how a storage system updates replica. This may happen synchronously, asynchronously in the background or as a combination of both.
- Erasure coding: Erasure coding uses RAID-like algorithms [18, 19] to distribute parts of data. If those parts overlap it is possible to restore data items even if a limited number of parts is missing. This obviously improves security as each provider knows only a tiny subset of the data item.
- Fragmentation: Here, items of type 1 are stored at provider A while type 2 is stored at provider B. This is useful when functional (e.g., data structure) and non-functional requirements (e.g., geographic location, durability, consistency) differ for different types of data.

2) PaaS:

PaaS offerings are hard to use redundantly as they usually not only follow a different programming model and support only a limited number of programming languages but also do applications developed for a particular PaaS offering make use of an entire ecosystem of services provided just within that PaaS offering. Furthermore, PaaS generally introduces limitations on the programming model they build upon so that applications need to be fine-tuned for a particular platform. So, the only sensitive alternative when trying to use federated PaaS offerings is to use one, for which an open source offering exists, which can, hence, be hosted by the customer or on top of IaaS compute resources. An example would be to redundantly use Google App Engine and AppScale running on top of Amazon EC2.

3) SaaS: Multiple SaaS offerings can be used redundantly with focus on different aspects:

- Focus on user experience: In this case, software services with similar functionality are used concurrently. An application could, e.g., allow the end user to toggle between visualization using Google or Bing Maps. This could enhance user experience by enabling a user to use a service he is used to. As a side effect, it would increase availability.
- Focus on availability: In this case software services with similar functionality are required but not used concurrently.

 ISBN: 978-1-61208-153-3

An application might switch over to a backup service in case of unavailability of the primary service.

While fine-grained SaaS offerings, e.g., Map services, can be used in a federation context relatively easy, it is very hard and probably cost-intensive to federate more complex services like, e.g., Salesforce. The difficulty to federate such offerings is caused by the fact, that it is virtually impossible to isolate smaller building blocks of the service as no competing solutions exist, which offer exactly the same functionality. Also, the potentially proprietary data formats and APIs of such services increase the problem. We believe that the issues related to the federation of SaaS offerings with larger granularity cannot be addressed in an adequate way by technical approaches and are therefore beyond the scope of this paper.

B. Migration

Migration incorporates scenarios where data respectively resources are being transferred from one Cloud provider A to another Cloud provider B. We identify two types of migration:

- Shadowed or redundant migration: In a migration scenario multiple similar services are usually only used for a limited amount of time, during which the old service is still operational while the new service is introduced. In the beginning, the new service is shadowing the old service to test it with live data. After switching over, the old service is shadowing the new one as a fallback solution in case of unanticipated failures. Finally, the old service is put out of service and the migration is finished.
- Non-redundant migration: Here, there is a hard switch-over. There is no shadowing period before or after.

In addition to those two types, we distinguish between full and partial migration:

- In the case of full migration an entire service stack is migrated, i.e., all components belonging to a certain service are migrated, e.g., a web server along with its database.
- Partial migration is linked to service disaggregation and describes the migration of service components or modules. A service composed of multiple components can be disaggregated into sub-services, some of which may then be migrated, before being reestablished as separate service.

IV. Towards a Reference Architecture

Cloud services offer access to services, which are associated with pools of stateful *resources*, e.g., virtual machines, data storage, queues, e-mail systems, etc. Our concept of a resource is similar to the notion of resources within the WS-Resource framework [20], however, less formalized because cloud services do not necessarily standardize on Web Service specifications.

We distinguish between two types of programmatic access to these resources:

- Resource API
- Management API

Applications implement the Resource API to access and utilize resources, which are exposed as business logic. For example, Amazon S3 offers a Resource API to create, read, update, and delete basic storage volumes ("buckets") as well as to upload or download data objects. Within the business logic of a photo-sharing application, buckets could be used as photo albums and a data object within a bucket could represent a photo image file. Table I illustrates that a photo-sharing application could be implemented with Cloud services from either Amazon or Google - or with a mix of services from both providers.

TABLE I
EXAMPLE PHOTO-SHARING APPLICATION.

Application feature	AWS	Google App Engine
Photo storage	S3 buckets & obj.	Data store or Blobstore
Photo notification	SQS or SNS	Channel service
Image editing	N/A	Image service
Photo sharing	SES	Mail service

The Management API helps application developers and administrators to manage resources efficiently. This includes a variety of activities: monitoring, deployment, data management, and so on. For example, Amazon EC2 offers a Management API for managing virtual machines (e.g., launch, stop, terminate) along with related settings and add-on services (e.g., security groups, block storage volumes, static IP addresses). Google App Engine offers a Management API to deploy application packages into the runtime environment and a dashboard for monitoring and administration (e.g., logs, cron jobs, datastore indexes, application versions and release management).

A. Two Perspectives on Interoperability

Interoperability challenges can be viewed from the perspective of a service provider or from the perspective of a service user. A service provider could be interested in offering distributed system services, which are interoperable with established, proprietary de-facto standards. Service users, on the other side, could design and implement applications with adaptors to multiple service providers, thereby enabling federation.

Table II shows 5 open source systems, which offer services similar to Amazon EC2 and Amazon S3. These systems support the interface definitions and protocols of their Amazon Web Services counterparts. Currently, all of the open source systems offer merely a small subset of comparable services and therefore only cover a subset of the Amazon Web Services API. The quality of a hosted open source solution, however, significantly depends on the system management skills of the hosting provider with regard to system scalability, performance and fault-tolerance. The systems could for example be backed with open source distributed system solutions, such as Apache HBase or Apache Cassandra, thereby providing a basis for achieving higher quality levels. Still, this induces even more integration challenges.

 ISBN: 978-1-61208-153-3

TABLE II
AMAZON WEB SERVICES (AWS) COMPATIBLE OPEN SOURCE CLOUD MANAGEMENT SYSTEMS.

Name	API	AWS-compatible services
Eucalyptus	AWS	Eucalyptus (EC2), Walrus (S3)
OpenNebula	OCCI, AWS	OpenNebula (EC2)
CloudStack	CloudStack, AWS	CloudStack + CloudBridge (EC2)
OpenStack	OpenStack, AWS	OpenStack: Compute, Image Service (EC2) & Object Storage (S3)
Nimbus	WSRF, AWS	Nimbus (EC2), Cumulus (S3)

TABLE III
COMPARISON OF MULTI-CLOUD LIBRARIES AND UTILITY PROGRAMS.

Name	Lang.	License	AWS[a]	RAX[b]	GOOG[c]	VMW[d]	MS[e]	GG[f]
jclouds	Java	Apache2	yes	yes	yes	yes	yes	yes
JetS3t	Java	Apache2	yes	no	yes	no	no	no
fog	Ruby	MIT	yes	yes	yes	no	no	yes
boto	Python	MIT	yes	no	yes	no	no	no
libcloud	Python	Apache2	yes	yes	no	yes	no	yes
deltacloud	Ruby	Apache2	yes	yes	no	no	no	yes
Whirr	Java	Apache2	yes	yes	no	no	no	no
PyStratus	Python	Apache2	yes	no	no	no	no	no

[a]Amazon [b]Rackspace [c]Google [d]VMWare [e]Microsoft [f]GoGrid

Table III shows a list of multi-cloud libraries, which enable interoperability across similar cloud services on a higher level than the systems discussed before. During the implementation of an application, the libraries jclouds, JetS3t, fog, boto, libcloud, and deltacloud are linked into the build path. When the application has been implemented, it can be deployed using any of the library-supported cloud services. This simplifies migration processes and redundancy setups as described in the sections before as there is no need to re-design the application. Instead, simple configuration options, usually just the service endpoints, must be changed. Figure 2 illustrates the migration of a service and the impacts on the service endpoints and the thereon based application.

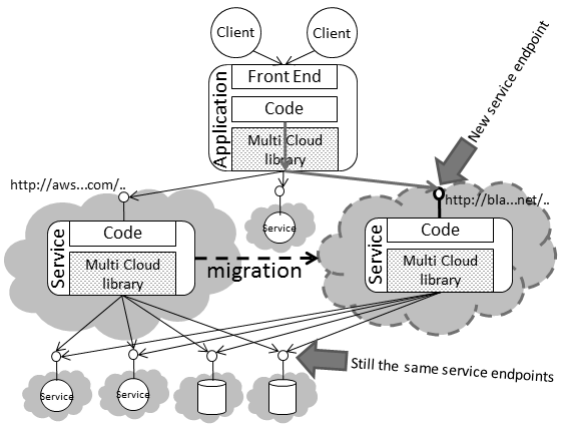


Fig. 2. Migration scenario illustrating impact on service endpoints

Additionally, utilities like Whirr (based on jclouds) and PyStratus can be used to deploy complex distributed systems on top of exchangeable compute clouds, such as EC2 or the Rackspace Cloud.

Both strategies, interoperable open source solutions and multi-cloud application code, can be employed to facilitate transparent application migration. Redundancy is more complicated to establish: Either it is explicitly foreseen in the application's code or there is a federation system providing a suitable programming abstraction. An additional layer decouples the application from the actual resources and permits their transparent reconfiguration, e.g., change redundancy strategy to erasure coding. Figure 3 illustrates the two strategies.

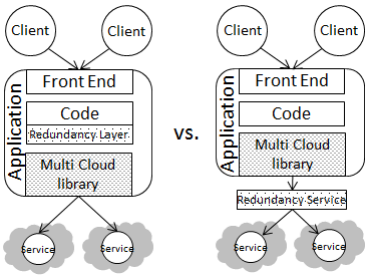


Fig. 3. Redundancy Strategies

B. Potential Reference Architecture Components

The open source cloud management systems in Table II and the multi-cloud software libraries in Table III are crucial elements for creating a federated cloud application. However, the systems and libraries should be discussed in a wider context to answer how applications can be migrated from one cloud service to another or operated on top of redundant cloud services.

We suggest that a reference architecture should contain the following components:

- **Provisioning Engine**: takes an application package along with policies and maps business logic components to a pool of resources. The projected mapping along with management configurations is then executed and enforced through a Distribution Manager.
- **Distribution Manager**: contains multiple sub-components, through which it enforces guarantees specified with policies. For example, enforce consistency between data replica; enforce the same deployment configuration on multiple servers. It may also serve as a redundancy decoupling layer. Principal components are:
  - **Deployment Manager**: is a component of the Distribution Manager. Based on a deployment description the manager executes resource management commands through Resource Managers. It guarantees the avalabillity and the correct configuration of provisioned resources.
  - **Configuration Manager**: is a component of the Deployment Manager. It recreates virtual appliances

resp. application stacks based on stored configuration informations.
  - **Data Distribution Manager**: is a component of the Distribution Manager. It manages the distribution of data, e.g., data replication, data redundancy, according to the distribution strategies.

  The distribution managers secondary components are:
  - **Transformation**: is used to transform incompatible formats, e.g., virtual machine images and to map between different data formats.
  - **Monitoring**: gathers information about resource states and information about their configuration through the Resource Managers. In case of unexpected conditions the Distribution Manager adapts the system to match the projected provision mapping.
- **Resource Manager**: manages all resources in a unified way. It can be realized as a collection of resource-localized components. The Resource Manager provides an abstraction of the APIs of the underlaying services and allows the Distribution Manager to configure resources in different clouds in a unified way. It may use adaptors, for example multi-cloud libraries, to perform its tasks.

Figure 4 depicts our current vision of the reference architecture. We have to point out, that we are still doing research on the reference architecture and that the figure should only be considered a snapshot of our momentary work in progress.

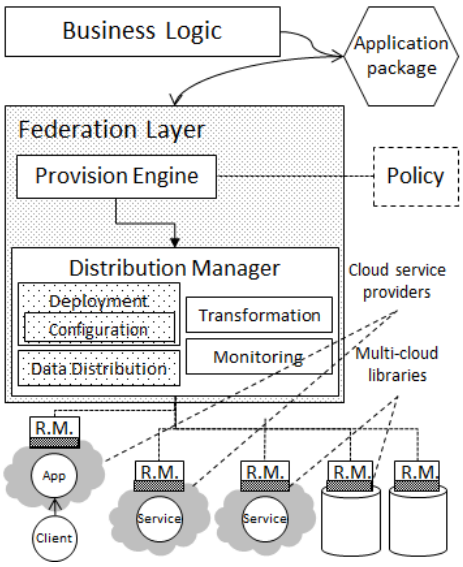


Fig. 4. Reference Architecture

V. Discussion

A. Vendor Lock-In and Cloud Computing

As depicted in Section II, vendor lock-in exists when potential switching costs surpass the benefits the customer would enjoy by switching to another provider. This is currently the case with Cloud Computing: by switching the provider the initial ex-ante investments could be largely lost and new investments, to adapt the software and retrain employees, will be necessary, thus exceeding the benefits of the provider-change. This implicates, that in Cloud Computing, lock-in, and in consequence hold-up, is a result of the different, proprietary interfaces, services and service offerings and the complexity involved in coping with this issues.

Since Cloud Federation resolves the above mentioned issues or - at the least - lowers the costs involved, we claim that it thereby resolves lock-in as well as hold-up and is a key enabler of Cloud marketplaces.

B. Future Work

Thoughts on Vertical and Secondary Services Federation are not incorporated in this article and will be subject to future works. Also the proposed federation reference architecture has to be elaborated in more detail in future works. Notably, we did not outline details on our vision of application packages and how the architecture's components could be realized.

VI. Conclusion

Cloud Federation is a concept, which has a large potential and might have an enormous influence on the way computing resources and applications will be handled, developed and used. It is a further step of providing computing resources in an utility-services-like way, similar to other services, e.g., electricity or water. However the evolution of Cloud Computing and related concepts and technologies is extremely dynamic and it is very difficult to make long-term prognoses. We believe anyhow, that this article can be a substantial contribution to future works on Cloud Federation.

Acknowledgment

The work presented in this paper was performed in the context of the Software-Cluster project EMERGENT [21]. It was partially funded by the German Federal Ministry of Education and Research (BMBF) under grant no. "01IC10S01". The authors assume responsibility for the content.

References

[1] R. Harms and M. Yamartino, "The economics of the cloud," Microsoft Corporation, Redmond, WA, USA, Microsoft Whitepaper, November 2010. [Online]. Available: http://www.microsoft.com/presspass/presskits/cloud/docs/The-Economics-of-the-Cloud.pdf [Accessed: June 24, 2011]

[2] "Microsoft: Smb hosted it commentary report," 2010.

[3] M. Armbrust *et al.*, "Above the clouds: A berkeley view of cloud computing," University of California at Berkeley, Tech. Rep., February 2009. [Online]. Available: http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html [Accessed: June 26, 2011]

[4] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009. [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc [Accessed: June 16, 2011]

 ISBN: 978-1-61208-153-3

[5] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the cloud? an architectural map of the cloud landscape," *Software Engineering Challenges of Cloud Computing, ICSE Workshop on*, vol. 0, pp. 23–31, 2009.

[6] C. Baun, M. Kunze, J. Nimis, and S. Tai, *Cloud Computing: Web-basierte dynamische IT-Services*, ser. Informatik im Fokus. Berlin: Springer-Verlag, Oktober 2009.

[7] R. Cowan, "Tortoises and hares: Choice among technologies of unknown merit," *The Economic Journal*, Jan. 1991. [Online]. Available: http://links.jstor.org/sici?sici=0013-0133(199107)101%253A407%253C801%253ATAHCAT%253E2.0.CO%253B2-S [Accessed: June 20, 2011]

[8] B. Klein, R. G. Crawford, and A. A. Alchian, "Vertical integration, appropriable rents, and the competitive contracting process," *Journal of Law & Economics*, vol. 21, no. 2, pp. 297–326, October 1978. [Online]. Available: http://ideas.repec.org/a/ucp/jlawec/v21y1978i2p297-326.html [Accessed: June 17, 2011]

[9] B. Holmstrom and J. Roberts, "The boundaries of the firm revisited," *Journal of Economic Perspectives*, vol. 12, no. 4, pp. 73–94, Fall 1998. [Online]. Available: http://ideas.repec.org/a/aea/jecper/v12y1998i4p73-94.html [Accessed: June 17, 2011]

[10] P. A. Grout, "Investment and wages in the absence of binding contracts: A nash bargining approach," *Econometrica*, vol. 52, no. 2, pp. 449–60, March 1984. [Online]. Available: http://ideas.repec.org/a/ecm/emetrp/v52y1984i2p449-60.html [Accessed: June 18, 2011]

[11] J. Tirole, "Procurement and renegotiation," *Journal of Political Economy*, vol. 94, no. 2, pp. 235–59, April 1986. [Online]. Available: http://ideas.repec.org/a/ucp/jpolec/v94y1986i2p235-59.html [Accessed: June 18, 2011]

[12] C. Ewerhart and P. W. Schmitz, "Der lock in effekt und das hold up problem," University Library of Munich, Germany, MPRA Paper 6944, 1997. [Online]. Available: http://ideas.repec.org/p/pra/mprapa/6944.html [Accessed: June 21, 2011]

[13] O. E. Williamson, *Markets and hierarchies: An analysis and antitrust Implications*. The Free Press, 1975.

[14] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251254.1251264 [Accessed: June 19, 2011]

[15] J. Broberg, R. Buyya, and Z. Tari, "Metacdn: Harnessing 'storage clouds' for high performance content delivery," *J. Network and Computer Applications*, vol. 32, no. 5, pp. 1012–1022, 2009.

[16] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in *ACM Conference on Computer and Communications Security*, 2009, pp. 187–198.

[17] D. Bermbach, M. Klems, M. Menzel, and S. Tai, "Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs," in *Proceedings of the IEEE Cloud 2011 - to appear*, 2011.

[18] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002, pp. 328–338. [Online]. Available: http://portal.acm.org/citation.cfm?id=646334.687814 [Accessed: June 23, 2011]

[19] Y. Saito, S. Frølund, A. C. Veitch, A. Merchant, and S. Spence, "Fab: building distributed enterprise disk arrays from commodity components," in *ASPLOS*, 2004, pp. 48–58.

[20] K. Czajkowski *et al.*, "The ws-resource framework," Tech. Rep. 1.0, March 2004. [Online]. Available: http://www.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf [Accessed: June 17, 2011]

[21] "Software-cluster emergent." [Online]. Available: www.software-cluster.org [Accessed: June 26, 2011]

APPENDIX

Table A - Cloud product overview

Name	URI
Amazon's EC2	http://aws.amazon.com/ec2/
Amazon's Elastic Beanstalk	http://aws.amazon.com/elasticbeanstalk/
Amazon's S3	http://aws.amazon.com/s3/
AppScale	http://code.google.com/p/appscale
CloudKick	http://www.cloudkick.com/
Google App Engine	http://code.google.com/appengine/
Microsoft Azure	http://www.microsoft.com/windowsazure/
SalesForce' Force.com	http://www.salesforce.com/platform/
ScaleUp	http://www.scaleupcloud.com/
SpotCloud	http://spotcloud.com
Zimory	http://www.zimory.com

Table B - Multi-cloud library overview

Name	URI
boto	http://code.google.com/p/boto/
deltacloud	http://incubator.apache.org/deltacloud/
fog	http://github.com/geemus/fog
jclouds	http://code.google.com/p/jclouds/
JetS3t	http://jets3t.s3.amazonaws.com/
libcloud	http://incubator.apache.org/libcloud/
PyStratus	https://github.com/digitalreasoning/PyStratus/
Whirr	http://incubator.apache.org/whirr/

Table C - Open source cloud management systems overview

Name	URI
CloudStack	http://cloud.com/
Eucalyptus	http://open.eucalyptus.com/
Nimbus	http://www.nimbusproject.org/
OpenNebula	http://opennebula.org/
OpenStack	http://www.openstack.org/

IaaS Clouds vs. Clusters for HPC: A Performance Study

Philip C. Church and Andrzej Goscinski

Deakin University, School of IT
75 Pigdons Road, Waurn Ponds, Victoria, 3216, Australia
Email: pcc@deakin.edu.au, andrzej.goscinski@deakin.edu.au

***Abstract*—The increasing amount of data collected in the fields of physics and bio-informatics allows researchers to build realistic, and therefore accurate, models/simulations and gain a deeper understanding of complex systems. This analysis is often at the cost of greatly increased processing requirements. Cloud computing, which provides on demand resources, can offset increased analysis requirements. While beneficial to researchers, adaption of clouds has been slow due to network and performance uncertainties. We compare the performance of cloud computers to clusters to make clear the advantages and limitations of clouds. Focus has been put on understanding how virtualization and the underlying network effects performance of High Performance Computing (HPC) applications. Collected results indicate that performance comparable to high performance clusters is achievable on cloud computers depending on the type of application run.**

Keywords – Cloud Computing, Benchmarking, Performance, System Biology, N-body simulation

I. INTRODUCTION

Cloud computing provides on demand computational resources of the Internet through use of virtualization, services and a pay-per-use paradigm. There has been interest in applying this computing technology to solve large scientific and industrial problems. By drawing resources from the cloud, even small research groups can solve these problems without investing in large amounts of computer infrastructure. However, cloud computing is still a developing technology and there have been many concerns about the overhead of virtualization and communication latency.

Virtual machines are used in Infrastructure as a Service (IaaS) clouds to provide users with dedicated systems which share underlying physical hardware. However there is a cost to create and maintain these isolated systems, a virtualization overhead, which is constantly subtracted from a user's allocated virtual resources [1]. Inconsistent network traffic flow also exists in clouds, which is problematic when running communication heavy applications [2]. It is in response to these issues, that some cloud providers have provided compute nodes which utilize hardware found in high performance computer clusters [3]. It is claimed that these High Performance Computing (HPC) enabled cloud nodes are optimized for running HPC applications yet it has not been proven in a practical manner.

This paper shows results of both the investigation of the feasibility of running HPC applications on clouds through benchmarking and the comparison of these results to cluster results. Two practical applications, an embarrassingly parallel bio-informatics visualization and communication bound N-body physics simulation, were chosen to represent classes of parallelization, data and functional parallelization. Using these applications HPC enabled clouds, standard IaaS clouds and a HPC cluster have been tested and compared. Of interest are the effects of virtualization and network latency, which have been documented to be the main performance issues [1][2].

The rest of this paper is as follows; Section II describes previous cloud benchmarks, their results and short fallings. Section III introduces the applications used during the benchmark; this is followed by a section introducing each computing platform and their specifications. Section V describes the methodology taken to setup each machine. Section VI presents performance results from the benchmarking, which is followed by a section investigating execution cost of the Amazon Elastic Compute Cloud (EC2) [13]. Finally, a conclusion and future work section is presented.

II. STATE OF THE ART

There are many advantages in using cloud computing for scientific research. For bio-informatics, running sequence alignment on the cloud (on a once per experiment basis) represents significant savings. Despite the increased range of cloud compatible bio-informatics software [4], adoption of on demand computing has been slow. Reasons for this slow adoption include usability and performance uncertainties [5].

A number of recent studies have investigated the performance of cloud computers. A solution by Napper and Bientinesi [6] runs LINPACK on Amazon Extra-Large instances (in both the Standard and High-CPU categories). Results indicate that these Amazon instances are not yet mature enough for HPC computations. Suggestions are made to offer better interconnects or nodes provisioned with more physical memory.

A study done by Indiana University measures the virtualization overhead of Xen and Eucalyptus through three practical applications (matrix multiplication, k-means clustering and the concurrent wave equation solver) Results showed a moderate-to-high virtualization overhead when running Message Passing Interface (MPI) applications [1].

 ISBN: 978-1-61208-153-3

A recent study by W. Guohui and T. S. E. Ng [2] investigated the network interconnect of EC2. An application called CPUtest was used to measure processor sharing, Round-trip Delay Time, Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) throughput and packet loss. Observed results show abnormally large packet delay variations between cloud instances. Unstable TCP/UDP throughput was also seen, caused by end host virtualization.

The main criticisms of these studies were addressed by Amazon's recent addition of HPC cluster instances. These instances have 10 Gb Ethernet interconnect and more physical memory. Limited performance results exist for this machine, the most relevant is a LINPACK study run on a cluster of Amazon's EC2 Cluster Compute instances (consisting of 7040 cores). Results ranked the Amazon EC2 cluster 231 on the TOP500 super computer list [7].

As seen in the above examples, previous performance studies made use of scientific applications, profiling tools or LINPACK. Results from these studies indicate there are problems when running communication bound applications on the cloud. While the LINPACK result from the Amazon EC2 cluster instances indicates these problems are resolved, the EC2 HPC offering has not been studied through practical applications. In addition, cloud setup and cost of running scientific applications on the cloud has not been addressed.

It is because of these short fallings that a cloud benchmark is presented. Focus has been put on investigating the effects of network speed and virtualization on HPC optimized clouds. The financial cost of executing applications on the cloud is also examined. By basing this study on solving common scientific problems in bio-informatics and physics, a realistic case can be made for or against the use of cloud computing for scientific research. Comparisons are made between clouds and the currently used high performance clusters in order to quantify results.

III. APPLICATIONS

Scientific computing is a source of large scale problems. The amount of data collected in the fields of bio-informatics and physics has been exceptional, and data analysis can exceed the available computational time and storage. Cloud computing could be used to support large data analysis and solve large problems. A common application from each scientific field was chosen; in this way the measurements could be applicable to real life problems. This section describes the operation of applications used during the benchmarking study.

A. Bio-informatics Application

A patient's genome can be screened for cancers before any visible symptoms appear, and finding the inflicted subtype of cancer can lead to personalized cancer treatments. To facilitate these personalized treatments of cancer, signatures of cancer subtypes need to be collected. A common bio-informatics workflow used to find these subtypes involves building system models [8]. System models show the interaction of genes in a biological system, and are built by correlating genes together. Building a system model is an N × N problem, given a list of N genes; N correlations are required for each gene. This workflow consists of many steps including; normalization and filtering of data, statistically correlating genes and then visualizing these results in a network diagram.

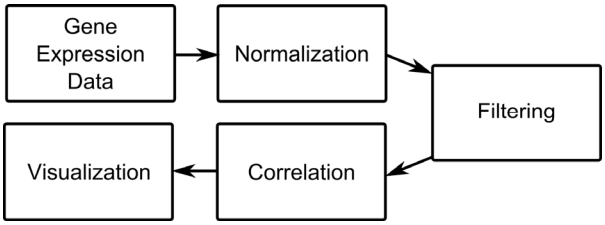


Fig. 1. A Common System Network Workflow.

The system network workflow presented in Fig. 1 makes use of data representing the amount of activated genes, also known as gene expression, in a biological sample. In order to find accurate relationships between genes, collecting both trait exhibiting and control expression datasets is necessary. Collecting this gene expression data involves multiple observations of genes in the biological system of interest. During this observation process human error can be introduced through uneven handling or scanning of samples. Normalization removes this bias by removing background noise from signal intensities and standardizing data so that distribution remains the same. Normalized data is then filtered, reducing the problem set by selecting genes that contain large variation. Correlation algorithms are then used to find the relationships between genes; commonly used correlation algorithms include Pearson's coefficient and Spearman's rho [9].

B. Physics Application

Data collected by particle accelerators such as synchrotrons and the Large Hadron Collider generate terabytes of data. By comparing simulations to collected results, it is possible to gain a better understanding of the laws that govern the universe [10]. We run a simulation of two disk galaxies colliding using an astrophysics application called GADGET [11]. This application is designed to simulate collision-less simulations and smoothed particle hydrodynamics on massively parallel computers. GADGET uses a combination of a physical mesh and tree based algorithms to simulate large range and small range particle interactions.

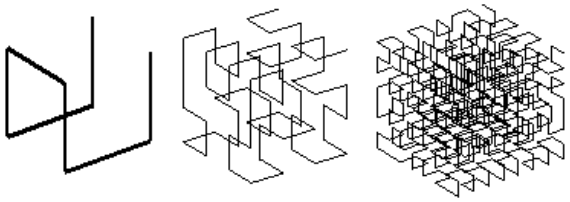


Fig. 2. 3D Representations of the Peano–Hilbert Curve.

Before each simulation step, physical mesh data decomposition is used to break the simulation area into

 ISBN: 978-1-61208-153-3

TABLE I
LIST OF BENCHMARKED COMPUTER PLATFORMS

Names	Nodes	Hypervisor	Platform	Hard Drive	CPU	RAM	Network	Interface
Amazon (Cluster)	8	Modified Xen: HVM	64-bit CentOS	Elastic Block Store	2 x Intel quad-core Nehalem (2.93 GHz)	23 GB	10Gb Ethernet	Web-based console. SSH.
Amazon (Large)	17	Modified Xen: Paravirtual	64-bit Ubuntu 9.10	Elastic Block Store	2 x Xeon equivalent (2.2 GHz)	7.5 GB	High I/O	Web-based console. SSH.
Amazon (Small)	17	Modified Xen: Paravirtual	64-bit Ubuntu 9.10	Elastic Block Store	2007 Xeon equivalent (1.6 GHz)	1.7 GB	Low I/O	Web-based console SSH.
vSphere Cloud	10	VMware	64-bit Ubuntu 9.10	Separate Drives	2.33 Ghz Intel Duel Core	2 GB	InfiniBand 10Gb	Web-based console, SSH. Remote Display.
InfiniBand Cluster	10	None	64-bit CentOS	Shared Drives	2.33 GHz Intel Quad Core Duo	8 GB	InfiniBand 10Gb	SSH.
HPCynergy	20	VMware	64-bit CentOS	Shared Drives	Virtual: Hexa-cores (2.33 GHz) Physical: Dual Quad Cores	8 GB	InfiniBand 10Gb	Web Interface. Web Service.

pieces. To achieve equal load balancing, GADGET makes use of the Peano–Hilbert curve to map 3D space onto a one dimensional curve. The Peano–Hilbert curve (see Fig. 2) is a space-filling curve variant which visits every point of a square grid. Once calculated, this curve is cut into pieces that define the individual domains. After the problem state has been reduced, it is distributed to multiple processors. Because this decomposition step occurs after every simulation step, load on processors are balanced.

In order to simulate the movement of galaxies the gravitational forces operating on close range particles need to be calculated. Calculation of force can be simplified by treating groups of similar particles as a single entity. In this way it is possible to summarize gravitational interactions between particles using a single force value. This force is calculated by adding together the mass of all particles in an area. While this method is quick, it is only accurate when particles are far away and will not work when particles are close. The accuracy of this method is improved through sub-division of the starting area.

IV. INTRODUCTION TO BENCHMARKED PLATFORMS

One physical machine and three cloud systems were used during this benchmarking. Naming conventions of the machines are as follows; the cluster is hereby referred to as the InfiniBand Cluster, while each cloud is referred to by the cloud management interface (vSphere [12], Amazon [13], HPCynergy [14] [15]). The vSphere and HPCynergy clouds are private clouds whereas Amazon is a public cloud. In terms of hardware, these computer platforms were chosen to be as similar as possible to each other, when possible utilizing the same pool of hardware. Of the four machines described below, HPCynergy, the vSphere and InfiniBand Cluster use the same hardware; the Amazon machines use their own individual hardware.

Despite the large effort taken to minimize hardware differences, some Amazon instances differ in the amount of cores per processor. Because of this variation, each process was mapped to a single core and when possible a single node. To validate the mapping process CPU usage was monitored during data collection, for example a duel core system with a single process would be using 50% capacity. This methodology was chosen as it is similar to that used by the cloud computers, in that virtual machines are mapped to physical hardware.

Three Amazon instance types [3] were tested; Small, Large and Cluster. It has been documented that Amazon uses a modified version Xen as the hypervisor. In each case the Amazon Elastic Block Store (an Amazon service which provides persistence storage of virtual hard-drive) was used to store the state of the deployed virtual machines. Amazon measures the performance of CPU's in Amazon Compute Units (ACUs); this is equivalent to an Intel Xeon chip. Each Amazon Small Compute instance contained 1 ACU and 1.7 GB RAM. Large instances contain four ACU and 7.5 GB of RAM. The Amazon Cluster Compute instances contain two Intel "Nehalem" quad-core CPU running at 2.98 GHz and 26 GB of RAM.

The second cloud used in this benchmarking was based on VMware virtualization technology. This private cloud made use of the same physical machines as the InfiniBand Cluster. A ten node virtual cluster was deployed through this VMware cloud, each with duel core processors running at 2.33 GHz. A 10 GB InfiniBand network was used to provide inter-node communication. VMware vSphere is used as the management software providing the ability to create, deploy and access virtual machines.

 ISBN: 978-1-61208-153-3

The third cloud used in this benchmarking was HPCynergy [14]. HPCynergy is a HPC cloud solution developed at Deakin University which incorporates a publishing service and broker. This cloud platform exposed VMware virtualized nodes running on the InfiniBand Cluster. A total of seventeen compute nodes were utilized through HPCynergy, each node containing a hexa-core processor running at 2.33 Ghz. A 10Gb InfiniBand network provided inter-node communication.

The InfiniBand Cluster used in this benchmarking is a bare-metal system consisting of 10 nodes each with an Intel Quad Core Duo processor running at 2.33 GHz. Each node utilizes 8 GB of RAM and runs a 64-bit version of CentOS to take advantage of this amount of RAM. As a machine dedicated to HPC, nodes are connected using 10 GB InfiniBand and a mounted network drive allows users to easily setup MPI applications. In terms of CPU speed and RAM size, this machine is equivalent to the documented specification of the Large Amazon instance. This machine differs from the Amazon instance having a faster network interconnect. Specifications of all platforms used in the following benchmarking are summarized in Table I.

V. SETTING UP THE CLOUD: METHODOLOGY

Setting up computer resources for High Performance Computing is both a time consuming task, and one that serves as an interruption to research. While the InfiniBand Cluster used in these benchmarking could be used once code had been compiled, the Amazon and vSphere clouds required modification to enable HPC. The HPCynergy cloud solution aims to reduce setup time by exposing systems which have middleware already setup.

Amazon and vSphere clouds required a number of steps including; transferring source code, configuring the compiler's dynamic linker, compiling the source code and any dependencies, configuring the sshd client, generating public and private keys, passing public keys to all nodes and creating a machineFile for MPI. The above steps were not required when setting up HPCynergy due to its unique interface. Like other clouds, HPCynergy monitors and acts as a broker to linked (physical and virtual) hardware. However instead of hiding the state and specification of hardware from the users, the opposite approach is taken. Users are informed of the software and underling (virtual) hardware specifications of each machine. This allows jobs to be optimized to the CPU architecture as well as minimizing the need to install specific libraries.

Some clouds had limitations which required additional setup time. The vSphere system did not contain any VM templates thus installation of the Ubuntu OS was required before operation. While all Amazon EC2 instances used in these benchmarks did not have common utilities such as the g++ compiler, the g77 compiler, vim or zip. Software compilation was more time consuming on the cloud systems. Missing library dependencies and compiler specific code meant that software would often fail during compilation.

Once each system was setup, input data and generated results had to be transferred from the user terminal to the cloud. Table II shows the total input/output transfer time and data size for each benchmarked system. For each benchmark, a total of 300 Mb was transferred between computers. Private clouds completed upload and download within seconds, however public Amazon clouds took many minutes. Results indicate that the time taken for data transfer is not just dependent on data size and network speed. Xen virtualization and differences in cloud interconnects can explain the variation between Amazon transfer times [3].

TABLE II
TOTAL DATA TRANSFER TIME

Computer Platforms	Input (Min)	Input Data Size	Output (Min)	Output Data Size
Amazon (Cluster)	3.8	85.2 Mb	4.3	231.2 Mb
Amazon (Large)	5.5	85.2 Mb	5.8	231.2 Mb
Amazon (Small)	6.8	85.2 Mb	23.8	231.2 Mb
vSphere Cloud	0.2	85.2 Mb	0.4	231.2 Mb
InfiniBand Cluster	0.2	85.2 Mb	0.4	231.2 Mb
HPCynergy	0.2	85.2 Mb	0.4	231.2 Mb

VI. BENCHMARKING

Comparisons made between collected results highlight the effects of virtualization and network latency of specific cloud platforms for high performance scientific computing. HPCynergy and Amazon's Cluster compute claim to address many of these weaknesses [3], [14]. HPCynergy is a software based solution while Amazon makes use of faster hardware. Benchmarking is used to prove that these HPC cloud platforms are feasible in regards to performance. To test performance, the system biology pipeline (Section III.A) and GADGET application (Section III.B) were run on a number of commercial cloud solutions, dedicated clusters, as well as virtual nodes discovered and used via HPCynergy. To ensure optimal performance, before analysis, input data was transferred to the local file system of each machine.

A. Bio-informatics Benchmarking

Performance of the system biology pipeline (described in Section III.A) was recorded from five machines, the Small and Large Amazon virtual clusters, the private vSphere cloud, the HPCynergy cloud and the InfiniBand Cluster. Results for each machine were measured up to four nodes; each test was run three times in order to ensure the validity of results.

As seen in Fig. 3, results show an almost linear increase of performance to available resources; this is expected as most of the system network workflow is embarrassingly parallel. When compared to physical hardware, the VMware based cloud shows a noticeable increase in required computational

 ISBN: 978-1-61208-153-3

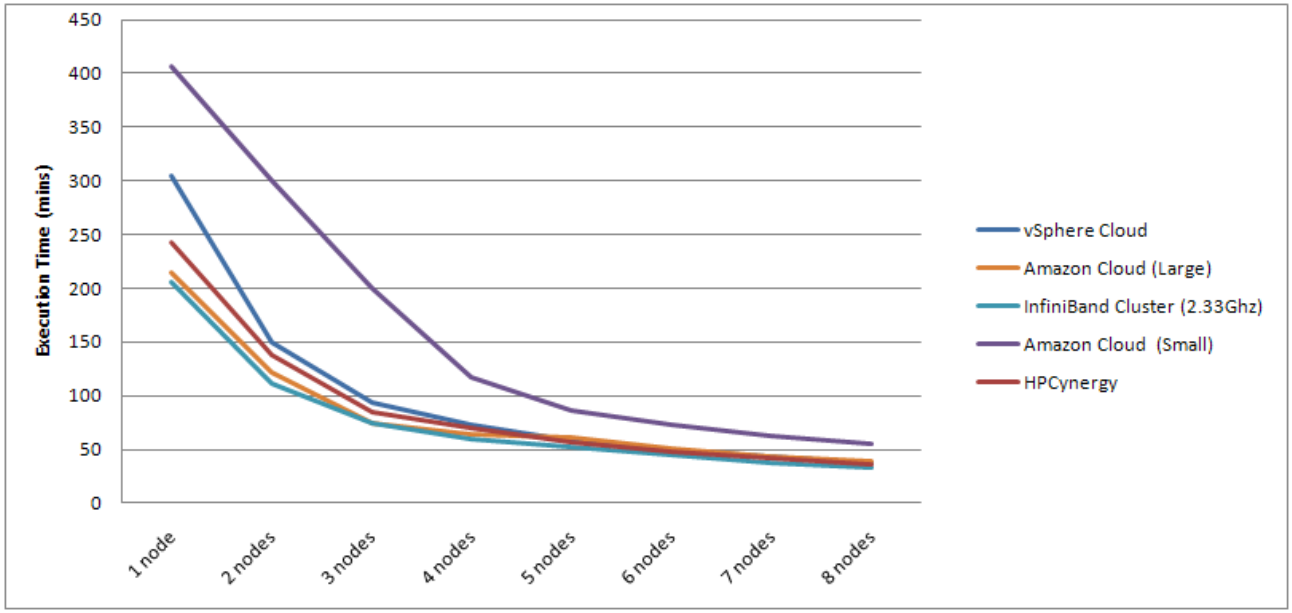


Fig. 3. IaaS Cloud Performance Comparison: Biological System Networks.

time. It is likely that this increase is due to virtualization overhead, in which part of the CPU is constantly being delegated to simulate the specified environment. Additional cloud services may also be responsible for decreased performance; this is seen in the HPCynergy platform which makes use of the same resource pool as the VMware cloud. When compared to the vSphere cloud, average performance is improved by 16%. The simple interface of HPCynergy allows for this improved performance, but it is not streamlined enough to match the physical hardware.

Additionally, collected results show an interesting relationship where the quicker a job runs the closer cloud performance matches physical hardware. This is due to virtualization overhead being distributed over many nodes. In the system biology pipeline, once job execution time became less than 35 minutes, virtualization overhead of clouds were indistinguishable from clusters.

In conclusion, different hypervisors and cloud service implementations have varying effects on performance. Amazon which uses a modified Xen hypervisor is very close to physical hardware, while the vSphere cloud which makes use of VMware virtualization suffered the most overhead. This virtualization overhead is minimised as jobs are spread across nodes.

B. Physics Benchmarking

The Small, Large and Cluster Amazon EC2 clouds, the private vSphere and HPCynergy clouds and an InfiniBand Cluster (see Section IV for extended specification details) were also utilized for the physics benchmarking. Benchmarking made use of full machine capacity, tests running up to 17 nodes. Each point was run three times in order to ensure the validity of results.

The results from this benchmarking can be seen in Fig. 4. As seen in the physical hardware results, the ideal performance of this GADGET benchmarking is a constant decrease as more compute nodes are added. The vSphere cloud, which runs on the same hardware, shows this shape with a similar offset seen in the bio-informatics study (Section A). Despite utilizing the same pool of resources and hypervisor, the HPCynergy solution sees an average performance improvement of 16% compared to the vSphere cloud. It is this simple interface of HPCynergy that allows for the improved performance results, but it is not streamlined enough to match the performance of the physical hardware.

Performance of the Amazon EC2 cloud varies depending on the instance type chosen. Performance of the Amazon Small instance shows a sharp computational increase at 2 nodes before performance becomes optimal at 3 nodes. The Amazon Large instance with higher I/O shows a similar early computational spike before optimizing at 5 nodes. Both the Small and Large Amazon EC2 cloud instances show an increase in computation time as more nodes are added past this optimal performance threshold. This relationship is an indication of a communication bottleneck, where each node is spending more time communicating then processing. Amazon's recently added Cluster Compute instance [13] has been optimized for running computation heavy applications. The performance of this instance shows a decrease in execution time mirroring other high speed clusters. This optimal performance is only guaranteed when allocating cluster instances at the same time. Because of this requirement the user loses one of the biggest draws to the cloud, the ability to elastically scale their applications.

Unlike the system biology problem presented in Section III.A, this N-body algorithm requires communication between nodes. Collected results from Amazon show that performance is not necessarily linked to amount of machines used. When running communication based applications, it is important that load is balanced between nodes and that

 ISBN: 978-1-61208-153-3

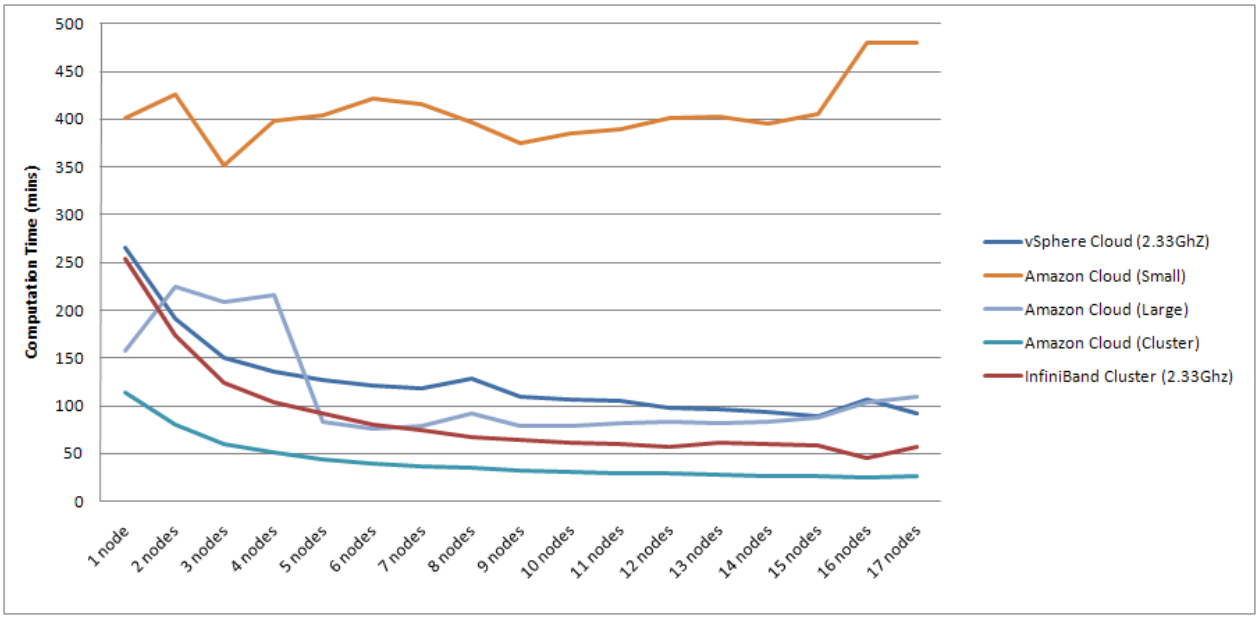


Fig. 4. IaaS Cloud Performance Comparison: N-body Simulations.

communication is minimized. If each node is communicating more than it is processing, the computation time will increase as resources are added. Cloud computers resources are highly distributed and performance of communication heavy applications can vary depending on the network architecture and the location of machines that have been allocated to the user.

VII. Cost Investigation

One of the big draws to the cloud is hardware scalability. Running a single machine for 5 hours costs the same as running 5 machines for 1 hour. Theoretically, this means the cost running an application should be the same regardless of time. This however may not be the case. Fig. 5 presents the cost per execution time of the Amazon instances run during the benchmark.

In terms of cost, the embarrassingly parallel bio-informatics application was the most efficient. While originally under-performing, the expected cost stabilization does occur in both the Small and Large Amazon instances at 5 nodes. Results from the physics benchmark did not show this trend. Running GADGET on the Small Amazon instance was wasteful, performance decreasing with each dollar spent. The large and cluster instances showed performance improvements with cost, the cluster instance scaling more consistently.

In conclusion, embarrassingly parallel applications are well suited to the pay on demand cloud model. Results show that execution time can decrease while maintaining the same total cost. Communication bound applications are not as cost efficient. Collected results show inconsistent performance per node and inconsistent cost-performance ratios. The main problem when utilizing the cloud for communication bound HPC applications is this performance unpredictability. Even the cluster instance, which showed the most consistent improvements did not show any hint of eventual cost stabilization.

VIII. Conclusion And Future Work

The results presented in this paper show that even standard public and even more private clouds can achieve performance similar to that of dedicated HPC clusters depending on the class of problem. When running embarrassingly parallel applications a near linear speed up is achievable and the results are comparable to those achieved on a cluster.

Clearly the effects of virtualization vary with the type of hypervisor used; Xen seems to have minimal performance effect on computation while VMware is noticeable. When running communication bound applications performance results vary. On the clouds with slow network speeds the N-body application achieved maximum performance at 5 nodes and then required compute time steadily increased due to communication overhead. The two clouds with HPC hardware (Amazon Cluster Compute instance, HPCynergy and VMware) showed the same decreasing performance trend as the InfiniBand Cluster. These performance results indicate that communication bound applications should be run only on clouds which provide high speed interconnect.

While some performance issues have been resolved, cloud setup is difficult and time consuming. A user must construct a virtual cluster and install analysis software. This setup process often starts through modifying of a pre-existing template. Templates can be difficult to utilize as they are often not documented, missing common dependencies (compilers, text editors, etc.) and may have a range of security access setups.

Benchmarking showed that transferring data to public clouds was a major issue. Compared to local clouds and clusters, public clouds increased data transfer requirements

 ISBN: 978-1-61208-153-3

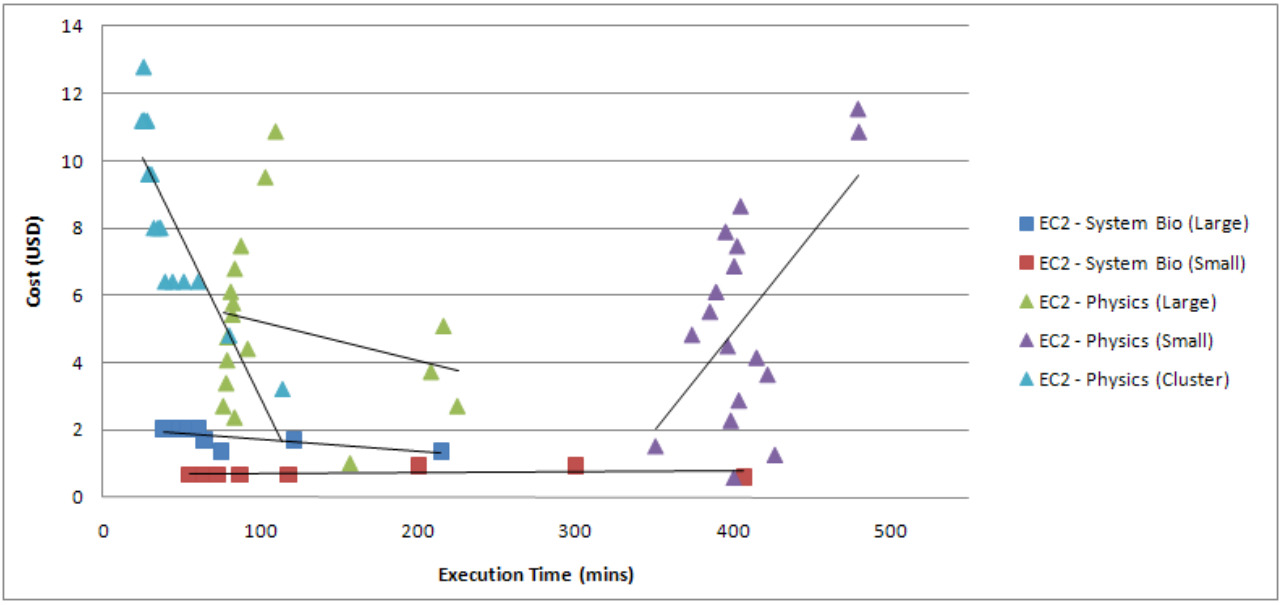


Fig. 5. Comparison of cost and execution time of the EC2 cloud.

by a factor of 25. This is problematic as the scientific applications described in this paper can make use and generate gigabytes of experimental data. At first glance the large transfer times are merely an artefact of the physical distance between cloud storage and user terminal. However collected data transfer results show significant variation between Amazon Cloud instances. This indicates that differences in cloud interconnects is also a concern, cloud storage and cloud instances often being separated. It is hoped that the adoption of faster broadband technologies should remove much of this data transfer delay.

Future work is planned to investigate the performance of clouds when running a wide range of applications. Of interest are other bio-informatics applications including; protein simulation and sequence alignment. With increased data, these applications will have profound effects in the fields of medicine and drug discovery.

It is also important to devise algorithms that take advantage of the cloud platform. To obtain maximum benefit from clouds, these algorithms must scale to large amounts of data and compute nodes while integrating solutions to minimise data transfer. It is possible to reduce the amount of input data by devising analysis methods which use compressed data. Another possibility is to devise cloud workflows which utilize the power of the user's desktop computer to perform data filtering and pre-processing. Currently we are investigating ways to stream data to the cloud; this allows faster processing turn-around (by minimizing idle compute time).

ACKNOWLEDGMENT

This work was supported in part by an Amazon Web Service Research Grant and the "Innovations through Broadband" FRC through the Deakin University, Faculty of Science and Technology contact.

REFERENCES

[1] D. R. Avresky, *et al.*, "High Performance Parallel Computing with Clouds and Cloud Technologies," in *Cloud Computing*. vol. 34, O. Akan, *et al.*, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 20-38.

[2] W. Guohui, and T. S. E. Ng, "*The Impact of Virtualization on Network Performance of Amazon EC2 Data Center."* pp. 1-9.

[3] Amazon (2010) *Amazon EC2 Instance Types*. Accessed 24 September 2010, http://aws.amazon.com/ec2/instance-types/

[4] B. Langmead, *et al.*, "Cloud-scale RNA-sequencing differential expression analysis with Myrna," *Genome Biology,* vol. 11, p. R83, 2010.

[5] H.-L. Truong and S. Dustdar, "Cloud computing for small research groups in computational science and engineering: current status and outlook," *Computing,* vol. 91, pp. 75-91, 2011.

[6] Jeffery Napper and Paolo Bientinesi, *"Can Cloud Computing Reach the TOP500?"* Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop (2009).

[7] Top500 (11/2010) *Amazon EC2 Cluster instances - TOP500.* Accessed 15 June 2010, http://www.top500.org/system/details/10661

[8] Khalil, I., Brewer, M.A., Neyarapally, T. and Runowicz, C.D. (2010) ***"****The potential of biologic network models in understanding the etiopathogenesis of ovarian cancer****."*** Gynecol Oncol. **116**(2):282-5

[9] J. L. Rodgers and W. A. Nicewander. *Thirteen ways to look at the correlation coefficient.* The American Statistician, **42**(1):59–66, February 1988.

[10] J S Bagla and T Padmanabhan (2008), *'Cosmological N-body simulations'*, Pramana, 49 (2), 161-192.

[11] Springel V (2005), *'The cosmological simulation code GADGET-2'***,** MNRAS, submitted, astro-ph/0505010

[12] VMware (2010) *VMware vSphere 4: Private Cloud Computing, Server and Data Center Virtualization.* Accessed 15 January 2011 http://www.vmware.com/products/vsphere/

[13] Amazon (2010) *Amazon Elastic Compute Cloud*. Accessed 24 September 2010, http://aws.amazon.com/ec2/

[14] A. Goscinski and M. Brock. Toward dynamic and attributed-based publication, discovery and selection for cloud computing. Future Generation Computer Systems V. 26, I. 7, 2010.

[15] Andrzej Goscinski, Michael Brock and Philip Church. ***"****HIGH PERFORMANCE COMPUTING CLOUDS",* Cloud computing: methodology, system, and applications (2011). CRC, Taylor & Francis group.

 ISBN: 978-1-61208-153-3

Introducing Federated WebDAV Access to Cloud Storage Providers

Sebastian Rieger
Steinbuch Centre for Computing
Karlsruhe Institute of Technology
Karlsruhe, Germany
sebastian.rieger@kit.edu

Harald Richter
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
hri@tu-clausthal.de

Yang Xiang
Rechenzentrum Garching
Max-Planck-Society
Garching, Germany
yang.xiang@rzg.mpg.de

***Abstract*—Affordable access to large online hard disks via the Internet has emerged by the continuous evolving of public and private storage clouds. However, difficulties arise as soon as users of such storages want to employ services from different cloud providers simultaneously, e.g., for collaboration among institutions that use different storage providers or for distribution of data backups. The reasons for this are dissimilar user accounts and incompatible access methods. This contribution describes a solution to that problem that does not need additional middleware to achieve the goal of unified authentication, authorization (AA) and access except WebDAV, which is an open standard. Our method is based upon a dynamic localization of the user by means of a world-wide unique user name. The solution is thus suitable for implementing federations of storage clouds in which multiple organizations can jointly provide a unified access to file systems that are distributed across the Internet.**

Keywords-Dynamic Federation; SAML; WebDAV; Storage; Cloud Computing

I. INTRODUCTION

A continuously increasing number of providers are offering online storage over the Internet. These providers are utilized by home users and professionals as well, for example to backup or to share files. The access to these files - although stored at different physical hard disks in the Internet - is transparent to the users because the services are cloud-based. Since the number of storage providers is already high, users also want to benefit from more than one provider at the same time, e.g., for exploiting the free storage space the providers offer them. We propose WebDAV [20] as a convenient way to access files in the cloud and over the Web. It is supported by various operating systems. However, WebDAV relies on the AA mechanisms of the underlying Web servers, which is why users have to maintain different credentials for each provider using an individual Web server. Furthermore, they have to login separately to every provider, thus creating multiple sessions simultaneously. A unified access across different public cloud storage providers is therefore not possible as of today. This also holds for private storage clouds that are established to offer access to distributed storage for users across different institutions. These problems are addressed in part by federated and user-centric identity management systems based on SAML [22] or OpenID [34] that offer Single Sign-On and unified AA across distributed Web applications.

In this paper, we introduce a solution developed to enhance WebDAV access to online storage with federated, SAML-based AA. An augmented WebDAV client was implemented by us to support HTTP redirects and sessions, as defined in the SAML profiles. The client is based on Shibboleth [17], which is a widespread SAML implementation in scientific communities.

Shibboleth is focusing on Web applications and requires the user to access his resources using a fully configured Web browser to handle HTTP sessions with storage providers and to manage the JavaScript- or HTTP redirects that enable the SAML-based Single Sign-On or the selection of the institution the user is affiliated to. To allow direct WebDAV-based file access e.g., in a file explorer without using a Web browser, we extended our client to support dynamic federation that allows an automatic discovery of the users' institution. The solution described in this paper allows consistent file access across different providers as in a single virtual file system. It enables federations that are spanning over multiple locations and companies to build-up a distributed, scalable and fault-tolerant file system across multiple cloud storage providers.

In Section 2, the state-of-the-art in WebDAV-based storage clouds is explained. Section 3 describes the mechanisms to enable federated AA for WebDAV-based file access in storage clouds using our novel combination of these techniques. Section 4 presents the implementation, together with the extensions to WebDAV and Shibboleth. Finally, Section 5 summarizes the results and gives an outlook to future research.

II. STATE-OF-THE ART IN WEBDAV-BASED STORAGE CLOUDS

There are several research groups such as [2] and [25] that are also working on Shibboleth-compatibility in WebDAV, however at the server side. An early version of this concept was for instance introduced by [17]. A solution similar to that but for grid environments was described in [10]. It is based on iRODS [10]. Other differences of these projects to our concept are that they do not employ WebDAV clients to support Shibboleth- and SAML-based

 ISBN: 978-1-61208-153-3

AA and focus on a Web browser instead. Additionally they treat storage providers as isolated items and therefore do not allow a unified, transparent access to different storage locations as offered in storage clouds.

A. *Isolated Storage Clouds*

State-of-the-art in storage cloud technology is that clouds are isolated as islands. Each cloud can be accessed by a user individually, and without any relation to other clouds as an online hard disk, either via a proprietary Web interface or via special software delivered by the cloud provider. Both methods allow to access and administer directories (sometimes called buckets) and files. Recent examples for such storage clouds are Amazon S3 [21], Google Storage [8] and Microsoft Azure Storage [1]. Based on these isolated clouds, additional services and providers have come into existence, which simplify access to and usage of online cloud storage for the end-user, such as DropBox [6], Mozy [16] or Ubuntu One [30]. However, a common de-facto or de-jure standard for an overall AA and access to multiple storage clouds does not exist yet. The industry consortium SNIA (Storage Networking Industry Association) [27] is working on such a standard called Cloud Data Management Interface [3] but it is unknown when it will be available. Furthermore, beside public clouds that already exist in IT infrastructures of scientific communities, private clouds are more and more emerging, e.g., as described in [11] and [28].

Private clouds are often based on open source implementations of online storage such as Eucalyptus Walrus [31]. They excel by providing a unified and thus simplified access method for a closed group of users from different institutions that is independent of the specific location the user wants to access his files from. The realization of such comfortable access normally needs proprietary applications and user interfaces to handle the AA. An example therefor is the AA infrastructure (AAI) of the Internet2 [4] or of the German research network DFN [5]. The underlying technology is usually SAML [22]. Shibboleth enables Single Sign-On and therefore unified AA across services, such as storage providers, that are joined in a cloud.

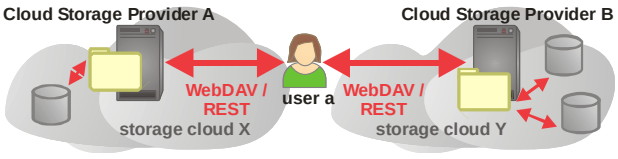


Figure 1. Access to WebDAV- or REST-based online file systems of different storage clouds.

This method is shown in Figure 1, together with a WebDAV- or REST-based access method. The REST application protocol is described in [18]. Providers for public storage clouds may also have a proprietary user interface. Examples therefore are the REST-based APIs in Amazon S3 or Google Storage which can be accessed only by special API function calls and by clients that are downloadable from these companies. The clients needed therefore typically map all file and directory accesses onto the HTTP methods PUT, GET, POST and DELETE which is similar to a REST-based approach. On top of these services, a few other providers offer a WebDAV access which enables users to create, read, write, move, rename and delete files and directories in an isolated storage cloud without proprietary applications or APIs. This was a paragon to us.

B. *User Credentials*

AA is typically performed with username and password as credentials. However, users must keep and maintain individual usernames and passwords for every provider e.g., because of company-dependent regulations with respect to password lengths and restrictions in the usage of numbers and special characters, because of the users' security concerns and because of the disjoint management of storage clouds used by different providers. As a consequence, no contemporary provider offers Single Sign-On across cloud borders yet. Additionally, the granting of read and write permissions is technically possible only within the runtime environments of the individual Web servers of a cloud storage provider that are in turn limited by their underlying file systems.

III. AA AND ACCESS IN FEDERATED STORAGE CLOUDS

The natural extension of isolated storage clouds lies in the coupling of them into federations. In [33], such a federation is described, which is based on REST and which uses uniform resource identifiers (URIs) instead of uniform resource locators (URLs). Beside HTML, it employs XML in its REST response packets. The disadvantage of that solution is that additional middleware is needed at the client side for accessing online file systems.

This paper describes a new approach for utilizing federations of storage clouds without extra middleware. To achieve this, the WebDAV protocol was employed that substantially augments the REST paradigm. Additionally, a Shibboleth-capable WebDAV-Client was developed by us as a replacement for Web browser-based user localization. It substitutes also static authentication and authorization by AA for a dynamic federation as it was described in [33]. By these measures, end-users can access federations that are evolving over time with respect to the number of users (which is mostly the case) and that are composed of various cloud providers (which is new) without installing middleware and without repetitive login. The latter feature results in Single Sign-On.

Using our approach, the WebDAV module mod_dav [12], which is native to the Apache Web-Server, can be engaged together with the mod_shib Shibboleth module [13] without any extensions on the server side and without using a Web browser to access the files on the client side, while maintaining Shibboleth compatibility at the same time. Our solution is also compatible with other Web servers such as Microsoft IIS. Only the WebDAV functionality at the client side had to be extended.

 ISBN: 978-1-61208-153-3

A. *Unique User IDs and Access Rights*

In our concept, user rights are not mapped to a username for the purposes of each individual provider but to a world-wide unique user ID, which is the email address [29]. From the email address, a second mapping is made to user rights that are valid for a service within a federation, i.e., also across cloud boundaries. Other examples for world-wide unique IDs are the Mícrosoft Security Identifier (SID) [26], the Globally Unique Identifier (GUID) e.g., for DCOM in Windows, or the Universal Unique Identifier (UUID) e.g., in the Interface Definition Language (IDL) of the Distributed Computing Environment (DCE). Usual Unix/Linux systems, however, have only the User Identifier (UID) [29] as defined in the POSIX standard, which is not globally unique. Thus, the simultaneous access to multiple storage clouds results in a problem when mapping the user name to a UID in Unix. Using our solution, this can be circumvented by outsourcing AA to an extra service, which is called identity provider, as described in the next section.

B. *Identity Provider*

In SAML-based federations [22], service providers (SPs) completely outsource the authentication to one or more identity providers (IdPs). A prominent example for such a federation in a scientific environment is the AA infrastructure of the Internet2 (= InCommon [4]) or the German research network DFN (= DFN-AAI [5]). We explain the functioning of DFN-AAI by means of an arbitrary employee at Max-Planck Society. This employee is easily authenticated and authorized by the IdP that is located at his home institute because the user is registered there. The IdP is responsible for all users of that specific institute and manages a well-defined set of access rights for its users via their usernames. However, the example employee can also profit from services that are offered at institutions outside of his home institute, even outside of Max-Planck Society itself, solely by his institute username, provided that those institutions also participate in DFN-AAI.

With this concept, Single Sign-On is possible across cloud borders because only the IdP that is responsible for its user performs AA. This holds if all cloud service providers (CSPs) of a (dynamic) federation make use of this IdP that is responsible for the example user. The described method is depicted in Figure 2.

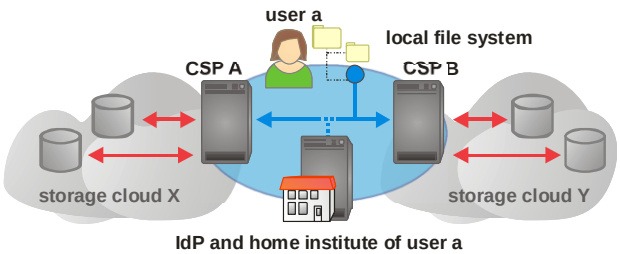


Figure 2. Single Sign-On in a (dynamic) federation of storage clouds by means of an Identity Provider (IdP).

Because of better scalability and the inclusion of many institutes, typically more than one IdP will exist in a federation of multiple cloud storage providers (CSPs). All IdPs must be registered at every provider. To simplify the operation, administration and management of this set of IdPs, the federation can make use of a function that is called „Identity as a Service“ (IDaaS).

In a scenario of CSPs, IdPs and IDaaS, the identity service is a central instance that is connected to the CSPs in a star topology and that acts for them as a proxy of the IdPs. The advantage for the CSPs resulting from the star topology and the proxy method is a significant simplification in AA because CSPs have to establish only a trust relationship to this central identity service and not to all IdPs. Subsequently, three indirections of trust come into existence according to the law of transitivity, starting with the trust relation from one CSP to the central IDaaS, and continuing with the trust relation to the individual IdP and finally to the user.

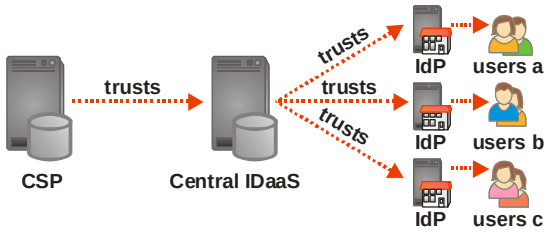


Figure 3. Multiple indirection of trust for AA in a federation according to [33].

This chain of indirection is depicted in Figure 3 and was first described in [33].

C. *Discovery Service*

In the following, it is assumed that the SAML-based Shibboleth system is used for AA. This is the groundwork for our solution. Shibboleth in turn, employs for user localization, i.e., for the determination of the user's home institute or organization a so-called discovery service (DS). The DS provides a Web page to all users under a static URL that is a priori known so that the CSPs can redirect the users to this page. If subsequent users request a service from a provider in the federation, the request is redirected first to the DS's Web site. On this Web page, a list of organizations that constitute the federation is presented to the user. Then, the user selects his home organization from the list, and thereby the IdP that is responsible for him. Afterwards, the browser window of the user is redirected a second time by an HTTP method to the responsible IdP, and the user must enter his username and password for AA.

D. *Confederations*

Several federations of storage clouds may even be coupled to an umbrella organization called confederation, for instance by using eduGAIN [7]. Then, the DSs of the constituting federations are cascaded, and the user must select first his home federation on the Web page of that DS

of the highest level, and proceed afterwards to the Web sites of the next lower levels. For this hierarchical procedure, the first DS presents a list of federations to the user that build the confederation, and in which the user selects his own storage cloud federation. After the selection, the Web page is redirected via HTTP to the DS of that federation where the user can select his home organization, and finally from there to his IdP where AA is accomplished as described.

E. User Comfort

Users wish for reasons of comfort a direct connection to their online file systems in the way of a virtual file system, as it was depicted in Figure 2, and they do not want to open an extra browser for this purpose, i.e., in order to select their federation or home organization. Nor do they want to switch continuously between a file explorer, such as Windows Explorer, and a Web browser for every list, move, rename and delete operation across cloud borders. A direct Web-based file access is instead accomplished by WebDAV. Furthermore, separate logins into all storage providers would result in a user-unfriendly approach. Finally, a Web page in the browser would fail if the number of IdPs in a storage cloud is rapidly changing over time or if the number of all clouds in a federation is time-dependent. Because of these reasons, we implemented a dynamic discovery service for user localization similar to that described in [32].

F. Dynamic User Localization

For dynamic user localization, the user first enters a static and a priori known URL of his CSP. Then he enters his email address instead of selecting his home organization, and our dynamic discovery service sends a DNS request to the responsible DNS server based on the domain name of his email address. The DNS response is a DNS NAPTR record in which his home organization has previously entered the IP address of the IdP that is responsible for him. If later a login of one of the users has to be performed then the user's NAPTR record is evaluated by the CSP, and with the data contained herein the proper IdP can be requested for AA ([33]). By this dynamic user localization, users can perform AA at their IdP from everywhere, and cloud borders are irrelevant. After successful AA, the user is automatically logged-in to all other CSPs of the federation (= Single Sign-On).

It is also possible to use the domain name of the user's email address as a shortcut to the domain of the IdP that is responsible for him. Another alternative to allow a domain-based DS using cascaded IdPs was described in [24].

IV. Software Architecture of Our Implementation

In this section, the software architecture of a Shibboleth-capable WebDAV client is described that can access federations of storage clouds. The client is based on an extension of the open source WebDAV client Sardine [23] and is therefore written as a Java Swing application. Our prototype supports down- and upload of directories and files from and to the online file system with a simple drag-and-drop command. Sardine in turn utilizes the known Apache HTTP Client [9] to access the WebDAV server. To allow for AA with Shibboleth, Sardine was extended to support HTTP sessions, redirects and SAML profiles.

This extension processes the HTTP redirects that are needed for the SAML-based Shibboleth system. Additionally, the extension provides the dynamic user localization for the CSP. Furthermore, it extracts the SAML response and the so-called relay state the user's IdP has sent after successful AA and transmits these data back to the user's CSP via a SAML HTTP POST profile. We have also extended Sardine with a simple session management to allow for stateful connections of the user while he is logged in. The user state is preserved in HTTP session cookies [19] that the CSP and the IdP have written by means of the WebDAV client during the user session. The WebDAV client can make use of this state information as needed. The session management enables Single Sign-On across different federated CSPs. Finally, the AA procedure of the IdP is performed such that the user's IdP and CSP exchange SAML attributes that define his access rights. Beside the CSP, also the WebDAV server can utilize the attributes (represented in HTTP headers), which grant or deny access to the underlying file system.

Inside of the Apache Web server used at the CSP, two modules named mod_dav and mod_davfs constitute the WebDAV server. While mod_dav implements the WebDAV protocol handling, the mod_davfs allows direct usage of the file systems that are available on the server side. The module mod_dav uses the module mod_auth from the Apache server and extensions of it for AA. Shibboleth is such an extension to mod_auth called mod_shib.

Using these modules it is possible to configure a so-called WebDAV location inside of Apache that uses the access control features of mod_shib. By requesting a resource from this location, the user is redirected to the DS and a subsequent AA is performed as described in the previous section. After successful AA (using the email address), the files in the location can be accessed and modified by our WebDAV client.

An example of an HTTP session offering Single Sign-On across different storage clouds and federations is shown in Figure 4.

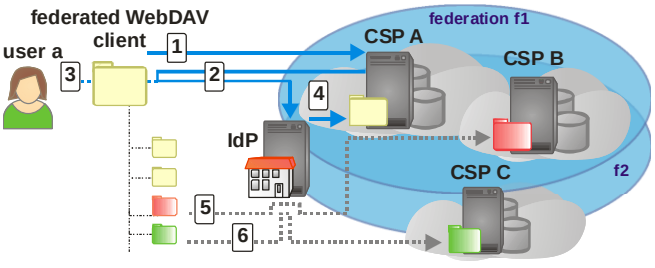


Figure 4. Federated WebDAV access to multiple cloud storage providers (CSP) and federations.

In step 1, our federated WebDAV client is used to access a cloud storage resource of CSP A, which is a member of federation f1. The corresponding request contains the e-mail address of user a. CSP A uses the e-mail address provided in step 1 to dynamically discover the user's home organization and IdP as described in the previous section. Step 2 depicts the redirection of the user to the corresponding IdP using the SAML redirection profile. The IdP requests the user to provide his e-mail address and password as login credentials using the WebDAV client in step 3. During the redirection process the WebDAV client stores the session cookies initialized by CSP A and the IdP. If the IdP is able to successfully authenticate the user by the credentials provided, then it transmits a SAML assertion to CSP A in step 4. Technically this is again performed using a redirection process.

As shown in Figure 4, the user has access to different folders via the federated WebDAV client. In step 5, the user requests a resource from CSP B that is also a member of federation f1. Using the e-mail address of the user sent by the federated WebDAV client in step 5, CSP B discovers the same IdP and redirects the client to ensure the AA. As the federated WebDAV client has already established an HTTP session with the IdP, we include the corresponding session cookie in the request on the client side. Hence the IdP does not require an additional authentication of the user, and the user gains seamless access to the resource.

One of the application areas of our solution is the federation of storage providers for private storage clouds. Such private clouds can be found, for example, in scientific communities in which bodies that are funded by the same organization and that are members of the same federation want to jointly aggregate their storage environments. Another scenario could be the aggregation of multiple public cloud storage providers e.g., in order to enhance flexibility and fault tolerance on the users' side.

In such scenarios, the need to access resources offered by CSPs that are not members of the same federation arises, especially with respect to distributed scientific communities that cooperate between multiple countries. Step 6 in Figure 4 illustrates this case. It is shown how resources of CSP C are accessed while C is a member of federation f2 and therefore outside the boundaries of federation f1. As we use a dynamic discovery of the user, our solution is able to handle such confederations.

A severe problem with respect to security arises if the federation is time-dependent in the number of their constituting CSPs, and thus in their number of users because the reliability and trustworthiness of new users may be unknown to the other users and to the CSPs as well. To overcome this problem, we have developed a so-called Trust Estimation Service (TES) [32][33]. This service can be incorporated into every IdP and CSP to increase security. The service is implemented as an extension to Shibboleth and it is thereby also an extension to the dynamic discovery service (DS). The extension utilizes the Internet Domain Name System DNS as described. The basic procedure of trust estimation is depicted in Figure 5.

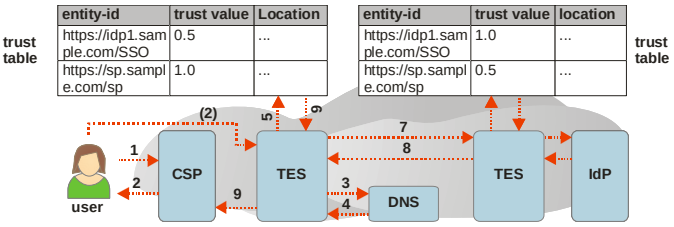


Figure 5. Extension of Shibboleth with a Trust Estimation Service according to [33].

In step 1, the user accesses a CSP and is requested to provide an e-mail address that is transmitted to the TES in step 2. The TES uses the domain name system (DNS) to obtain the Entity-ID of the user's home IdP (steps 3 and 4). Using this Entity-ID, the local TES selects the corresponding end-location in his trust table (steps 5 and 6). If the IdP is trusted, according to its previously calculated trust value (as described in [33]), the local TES of the CSP sends a request to the IdP's TES to retrieve the IdP's meta data in steps 7 and 8. Finally, the meta data is forwarded to the CSP in step 9.

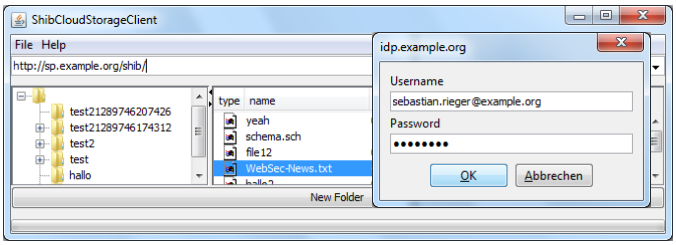


Figure 6. User interface to access federations of storage clouds via Shibboleth and WebDAV.

The resulting user interface of our WebDAV client is shown in Figure 6.

V. Conclusion and Future Work

The proposed solution allows unified authentication, authorization and data access in a federation of storage clouds. It can simultaneously present multiple online file systems from different cloud service providers to users as a virtual file system. It is based on Shibboleth and WebDAV that are extended by a dynamic user localization and a trust estimation service. This allows for Single Sign-On across cloud borders, for increased security in the federation, and for user-friendly access to the various storage clouds. Applications for the solution may be the federation between the scientific institutes of the Max-Planck-Society MPG-AAI [14] and the universities of the country of Lower Saxony Nds-AAI [15].

In the future, we will evaluate the performance of our approach and the influence of the WebDAV server configuration and protocol headers. The performance

 ISBN: 978-1-61208-153-3

evaluation will include an in-depth analysis of the TES and the established dynamic federation.

Additionally we'll focus on how object-oriented file systems e.g., based on NoSQL databases, can be integrated as a backend on the WebDAV server side. Furthermore we are planning to evaluate the inclusion of user-centric authentication (e.g., OpenID) and special authorization mechanisms such as OAuth to allow for the delegation of access rights across aggregated global file systems.

REFERENCES

[1] Windows Azure Storage, http://www.microsoft.com/windowsazure/ 9.5.2011.

[2] T. Bellembois and R. Bourges, "The open-source ESUP-Portail WebDAV storage solution", http://www.esup-portail.org/download/attachments/43515911/ESUPWebDAV.pdf 9.5.2011.

[3] Cloud data management interface, SNIA Web Site, April 2010, http://cdmi.sniacloud.com 9.5.2011.

[4] InCommon Identity and Access Management, http://www.incommonfederation.org/ 9.5.2011.

[5] DFN-AAI - Authentication and authorization infrastructure, https://www.aai.dfn.de 9.5.2011.

[6] Dropbox, http://www.dropbox.com/ and http://en.wikipedia.org/wiki/Dropbox_%28service%29 9.5.2011.

[7] eduGAIN, http://www.edugain.org/ 9.5.2011.

[8] Google Storage, http://code.google.com/intl/de-DE/apis/storage/ 9.5.2011.

[9] Apache HttpComponents, http://hc.apache.org/ 9.5.2011.

[10] S. Zhang, P. Coddington, A. Wendelborn, and A. Davis, "A generic interface for iRODS and SRB", 10th IEEE/ACM International Conference on Grid Computing, Banff, 2009.

[11] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications", Cloud Computing and Applications, 2008.

[12] Apache Module mod_dav, http://httpd.apache.org/docs/2.0/mod/mod_dav.html 9.5.2011.

[13] R. L. Morgan, S. Cantor, W. Hoehn, and N. Klingenstein, "Federated Security: The Shibboleth Approach", EDUCAUSE Quarterly, Vol. 27, 2004, S. 12-17.

[14] Max-Planck-Gesellschaft - MPG-AAI, https://aai.mpg.de 9.5.2011.

[15] Nds-AAI, Authentifizierungs- und Autorisierungs-Infrastruktur für Niedersachsen: http://www.daasi.de/projects/ndsaai.html 9.5.2011.

[16] Mozy, http://mozy.com and http://en.wikipedia.org/wiki/Mozy 9.5.2011.

[17] L. Ngo and A. Apon, "Using Shibboleth for Authorization and Authentication to the Subversion Version Control Repository System", Fourth International Conference on Information Technology - ITNG '07, Las Vegas, 2007.

[18] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures". PhD thesis, University of California, Irvine, 2000 and http://de.wikipedia.org/wiki/Representational_State_Transfer 9.5.2011.

[19] D. Kristol and L. Montulli, "HTTP State Management Mechanism", ftp://ftp.rfc-editor.org/in-notes/rfc2965.txt 9.5.2011.

[20] L. Dusseault, "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)" ftp://ftp.rfc-editor.org/in-notes/rfc4918.txt 9.5.2011.

[21] Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/de/s3/ 9.5.2011.

[22] OASIS: Security Services (SAML) TC, http://www.oasis-open.org/committees/security/ 9.5.2011.

[23] sardine - an easy to use webdav client for java, http://code.google.com/p/sardine/ 9.5.2011.

[24] S. Rieger, "User-Centric Identity Management in Heterogeneous Federations", Fourth International Conference on Internet and Web Applications and Services, 2009.

[25] DataFinder: A Python Application for Scientific Data Management, EuroPython 2008: The European Python Conference, Vilnius, 2008

[26] Security Identifier, http://en.wikipedia.org/wiki/Security_Identifier 9.5.2011.

[27] Storage Networking Industry Association, http://www.snia.org and http://en.wikipedia.org/wiki/Storage_Networking_Industry_Association 9.5.2011.

[28] J. Staten, S. Yates, J. Rymer, and I. Nelson, "Which Cloud Computing Platform is Right for You?", Forrester Research, 2009.

[29] User identifier, http://en.wikipedia.org/wiki/User_identifier 9.5.2011.

[30] Ubuntu one, https://one.ubuntu.com/ 9.5.2011.

[31] Interacting with Walrus (2.0) - Storage Service, http://open.eucalyptus.com/wiki/EucalyptusWalrusInteracting_v2.0 9.5.2011.

[32] Y. Xiang, J. A. Kennedy, H. Richter, and M. Egger, "Network and Trust Model for Dynamic Federation", The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences, Florence, 2010.

[33] Y. Xiang, S. Rieger, and H. Richter, "Introducing a Dynamic Federation Model for RESTful Cloud Storage", The First International Conference on Cloud Computing, GRIDs, and Virtualization, Lisbon, 2010.

[34] OpenID specification, OpenID Web site, http://openid.net/developers/specs/ 9.5.2011.

 ISBN: 978-1-61208-153-3

Cloud Capacity Reservation for Optimal Service Deployment

Iñigo San Aniceto, Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente
Distributed System Architecture Research Group (dsa-research.org)
Dept. de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid, 28040
Madrid, SPAIN
Email: inigosananiceto@pdi.ucm.com, rmoreno@dacya.ucm.es, rubensm@dacya.ucm.es, llorente@dacya.ucm.es

***Abstract*—Cloud computing is a profound revolution in the way it offers the computation capability. The Information Technology organizations do not need to oversize their infrastructure anymore, potentially reducing the cost of deploying their services. The main objective now is to reduce the cost of deploying a service in the cloud. Some research attempts have focus on deploying one service in multiple clouds, to benefit from different billing models. In this work, we propose a way to minimize that cost by using a single cloud provider with an optimal mixture of reserved and on-demand instances to take advantage of different billing models within the same provider. We tested this optimal combination of reserved and on-demand instances with real world workload traces. The results show a 32% deployment cost reduction compared to on-demand deployment.**

***Index Terms*—Cloud computing; capacity reservation; resource provisioning; service deployment; cost optimization**

I. INTRODUCTION

Cloud computing takes advantage of workload consolidation to operate more efficiently the resources and provide a service at lower cost. This itself is a tremendous advantage and makes the cloud computing services competitive in terms of prices. Information Technology (IT) companies used to oversize their resources to meet peak demands but now they have the option of using cloud computing [1][2][3][4][5][6][7].

Apart from the typical on-demand instances, current providers also offer reserved capacity. Although the pricing schemes for this reserved instances vary among cloud providers, they all offer discounts in the hour rates on one side and obligations or one-time payments on the other [8]. This means it is necessary a minimum amount of running hours to reduce the final price compared to on-demand instances.

In this paper, we present a novel algorithm to cover variable computation demands with mixed reserved and on-demand instances with the minimum cost. The idea is to eliminate the over-provisioning in the reserved instances to use the number of reserved instances that minimizes the final cost for the IT companies.

In addition, to avoid the performance degradation of the system, this novel algorithm estimates the number of instances that might be required in the next period, and provisions the instances in advance to hide start-up times. The provisioned instances are started-up and ready to use and they are a combination of reserved and on-demand instances.

The algorithm works as follows. In the first stage, the algorithm has to determine the optimal number of reserved instances.

The algorithm selects the number of reserved instances to reduce the cost of service for the IT companies This number is directly related with the number of running hours each reserved instance has.

Then the algorithm has to evaluate, for each period, the optimum number of provisioned instances. If the number of requested instances is lower than the number of reserved instances, all the provisioned instances are mapped on reserved instances. Otherwise, the difference would be fulfilled with on-demand instances.

To test the algorithm, reservation and provision cost of a standard instance in Amazon cloud provider [8] and real world traces from the Grid Workloads Archive are used [9].

The main contributions of this paper are the following:

1) We present an algorithm that minimizes the final cost of deploying a service in the cloud.
2) We present a model that predicts the optimal number of reserved instances for each period and use an algorithm to reserve them.
3) The model also predicts the optimum number of provisioned instances, and make advanced provision of those instances to hide start-up times.

This paper is organized as follows: Section II presents the state of the art and the current cloud computing market. Section III presents the definition of the problem with an appropriate statistical definition. Section IV presents the statistical analysis. Section V presents the reservation and provisioning algorithm. Section VI presents the improvements and Section VII presents the conclusions of the work and future work.

II. CURRENT CLOUD COMPUTING MARKET AND STATE OF THE ART

Currently, there are different pricing models in the market, being On-demand, Reservation and Spot the most common pricing schemes. Although these are the leading pricing configuration groups, there are differences among different cloud providers.

 ISBN: 978-1-61208-153-3

- **On-demand:** Probably, the most common pricing model. The main idea of this pricing configuration is to pay for the actual use with no other commitments. Most of the large providers offer this pricing model: Amazon [8], GoGrid [10], Rack Space [11] and Cloud Sigma [12] among others. Although the pricing model is similar in all the cloud providers, there are some differences.
  Amazon, for example, has some preconfigured instances with a certain amount of RAM, CPU, Storage, etc. For the following analysis, it is interesting to focus on the standard instance. It has 1.7GB of RAM, 1 Virtual core with 1 GHz and 160GB of storage. For this standard configuration, the price is 0.085$/h in N.Virginia [8].
  The situation in RackSpace is similar to the one in Amazon. There are preconfigured instances with different amount of RAM and storage and each one has a fixed price. The most similar configuration to the Amazon standard instance has 2GB of RAM, and 80 GB of storage and its price is 0.12$/h [11].
  Go Grid also offers preconfigured on-demand instances for 0.19$/h with 1GB of RAM, 1 CPU with 1GHz, and 50GB of storage [10].
  In Elastic Host, the on-demand pricing configuration is slightly different to the previous ones. There are not preconfigured instances, instead the instance types are user defined, and prices for the components are CPU(1GHz) 0.036$/h, RAM (1GB) 0.05$/h, Storage (1GB) 0.20$ /month. Comparing with the Amazon standard instance the price would be 0.164$/h [13].
  In Cloud Sigma, the on-demand pricing configuration is similar to Elastic Host but with price variability. The cost of RAM, CPU, Storage, etc. is not a fixed amount, but it is conditioned by the servers load. The boundary rates, for each charasteristic, are: CPU (1GHz) 0.0121-0.0504$/h. RAM (1GB) 0.0196-0.0579$/h. With this prices, an instance similar to the Amazon standard instance would cost 0.045-0.252$/h [12].
- **Reserved:** It is also a common pricing model. In this price configuration, there are always long-term commitments on one side, and discounts in the hour rates on the other. It is offered by most of the big cloud providers: Amazon [8], Rack Space [11], GoGrid [10] and Cloud Sigma [12] among others. The different providers also present some differences.
  Amazon establishes a one-time payment for the reservation. For each standard instance it is 227.5$ for 1 year reservation and 350$ for 3 year reservation. After the one-time payment, the discounts in hourly rates are fixed for each instance type, and they are approximately of 60%-65% depending on the type. For the standard instance the price reduces from 0.085$/h to just 0.03$/h without any compromise of use, i.e., the hourly price is charged only if the instances are running [8].
  Elastic Host uses a similar price configuration. It establishes one-time payment for reservation as Amazon does. The prices for the subscription are 77.76$ per month or 777.60$ per year, after the payment the instance prices have a fixed 50% discount regardless of the subscription period[13].
  In Cloud Sigma, the situation is different. It offers different discounts depending on the reservation period going from 3% for 3 months up to 45% for a 3 year reservation period, after the reservation is made the user has to pay for each instance as running [12].
  In Go Grid, the reserved price configuration is also slightly difference from the previous ones. It requires a monthly payment to acquire a certain quantity of usage hours. For the smallest instance, this payment is of 199$/month acquiring 2500 RAM hours. Considering 1 month has 744h a instance with 3.35GB of RAM can be used 100% of the time [10].
- **Spot instances:** This price configuration is not available in many cloud providers. The main idea of this pricing configuration is to set the maximum rate for the service hour, called bid price. Depending on the servers load, on the cloud providers, the spot pricing can change so that if the price is smaller than the bid price, the user have set the service will become available, on the other hand, a higher spot price makes the service unavailable. One of the few cloud providers that offer this price configuration is Amazon[8] .

These differences in the pricing models have lead to the creation of multiple cloud brokers. This cloud brokers try to minimize the cost of deploying the cloud service choosing the best price model across different cloud providers. There are many commercial solutions: RightScale, SpotCloud, Kavoo or CloudSwitch among others.

There are also some European projects especially oriented for Multi-Cloud deployment. Mosaic [14] is one of them and offers an open-source cloud API to develop multi-cloud oriented applications, and Optimis [15] is another that offer tools to simplify the construction and usage of hybrid clouds.

Finally, there is also some research works in this area: in [5], Moreno studies with the cost per-job with different cluster configurations. In the work we present, we also target to minimize the cost of deploying the cloud service by choosing the best price models, however, we do not use multiple-clouds; instead, we use the different price models within the cloud: on-demand and reserved instances.

For the availability problems that cloud computing might generate, some studies [16][17] focused on how to avoid availability problems, the algorithm we present faces the problem of availability with advanced provisioning based on load prediction instead of using instance leasing.

In [18], Konstanteli studies the flexible reservation periods to schedule the workflow and maintain the QoS. In this work, we have focused on the Amazon cloud provider and this provider, only offers 1 and 3 year reservation periods. After a brief study we have conclude that a 3 year reservation period is too long because the predictions are not accurate enough, and hence, we have used a 1 year fixed reservation period

 ISBN: 978-1-61208-153-3

In [19], Gardfjäll studies the credit based regulation of grid capacity allocation to avoid the performance loss due to the overuse also known as the "tragedy of the commons".

In conclusion, there are several pricing models in the market, and cloud brokers, take advantage of these differences to reduce the final prices. However, this produces several problems such as compatibility across different cloud providers that researches are trying to solve with some new tools such as Mosaic or Optimis. In this work, we targeted the same price reduction for service deployments, but we use the combination of reserved and on-demand instances in the same cloud provider. Hence, we have not any compatibility issues the multi-cloud environments produce.

III. Definition of the problem

The goal of the IT companies is to reduce the cost of deploying their service in the cloud provider without any performance degradation.

To achieve that, we propose a prediction model based on the historical data. With this prediction, the IT companies will use a mixture of reserved and on-demand instances to cover their demand.

This model estimates the number of reserved instances that minimizes the final cost, and the number of provisioned instances that fulfils the service requirements in 99.95% of cases. With this advance provisioning, it hides the start-up time (2-5 minutes [1]).

In order to create the prediction model, we obtain the cost of the cloud services and define the statistic sample space.

A. Costs

First we obtain the provisioning and reservation costs. For this analysis, we use the price configuration of a standard instance in the Amazon cloud provider [20].

Hence, the reservation cost of an instance is 227.5$/year with a provisioning cost of 0.03$/h, and on-demand instances have just a 0.085$/h provisioning cost [8].

With the previous values, we calculate the cost of the service for the IT companies. This cost will have two parts:

1) The first is the cost of reserving instances in the cloud provider

$$Cost_1(t) = Pres \times [Res(t) - Res(t-1)] \quad (1)$$

where Pres = 227.5$ is the reservation cost and Res(t) is the number of reserved instances at t [8].

2) The second is the cost of provisioning the instances. The cost of provisioning the instances will be $Pc_1 = 0.03$$/h for reserved instances and $Pc_2 = 0.085$$/h for on-demand instances [8].

$$Cost_2(t) = (R(t) \times Pc_1) \quad (2)$$

$$Cost_2(t) = (Res(t) \times Pc_1) + ((R(t) - Res(t)) \times Pc_2) \quad (3)$$

where R(t), is the number of requested instances at t.

The total cost will be

Algorithm 1 Instance mapping algorithm for cost calculation

if $R(t) \leq Res(t)$ **then**
  All the requested instances can be mapped to reserved instances and equation 2 is used.
else
  On-demand instances are necessary and equation 3 is used.
end if

$$Cost_{1year} = \sum_{t=0}^{1year} [Cost_1(t) + Cost_2(t)] \quad (4)$$

This is the equation that should be minimized by optimizing the resource reservation and provisioning of instances.

B. Definition of the Sample Space

In this work, we present a prediction tool. This prediction tool is based in a statistical analysis of the problem and hence it needs a proper definition of the sample space.

For now on, an instance will be treated as an indivisible unit being the total number of requested instances a discrete number. With this assumption the sample space will be finite and numerable. $\Omega = (0, 1, ..., L)$ where L is the maximum number of instances the service might need.

The algorithm provision a certain number of instances dividing the sample space in two relevant subsets representing mutually exclusive events.

The first $A \subset \Omega$ represents the case in which the necessary instances are less than the provisioned instances and, hence, there is no performance degradation. The second $B \subset \Omega$ where the IT companies need more instances than the ones the algorithm has provisioned and, hence, a performance degradation due to start-up time may occur [1]. Obviously the sample space satisfies $A \cap B = \phi$ and $A \cup B = \Omega$ creating a complete event system.

IV. Statistical analysis

As stated in the previous section, the goal of this work is to reduce the cost of deploying a service in the cloud provider without any performance degradation. The first step to get that reduction is to develop a statistical analysis of the workload of the IT companies. In this section, we first introduce the data used for the statistical analysis and then we describe the statistical analysis.

A. Trace data

We get the trace used for the study from the Grid Workload Archive [9]. In this website, different workload traces of different grids around the world are available.

These traces contain historical information about JobID, SubmitTime, WaitTime, RunTime, Number of Processors, Average instances Time Used, Used Memory, Requested Number of Processors, Requested Time, Requested Memory, Status, among other information.

 ISBN: 978-1-61208-153-3

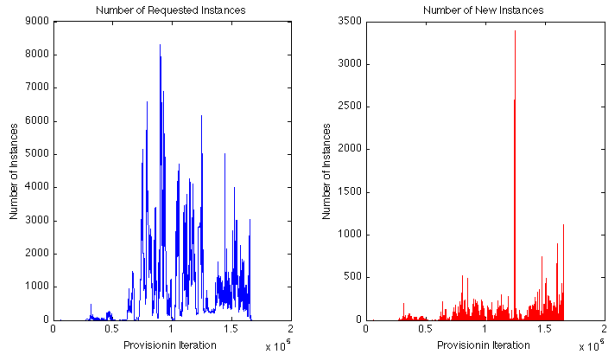


Fig. 1. Number of requested and new instances in 10 min samples from Nordugrid

From this data, we can easily obtain the number of new requested instances at each moment:

$$N(t) = \sum P_i(S_t = t) \quad \forall JobID \tag{5}$$

where N(t) are the new requested instances, P_i is the number of processors of the job i and S_t the submit time.

We can also calculate the number of terminated instances as:

$$F(t) = \sum P_i((S_t + W_t + D_t) = t) \quad \forall JobID \tag{6}$$

where F(t) are the instances that are not requested any more, W_t is the wait time and D_t is the demanded running time. The total number of requested instances at each moment is:

$$R(t) = R(t-1) + N(t) - F(t) \ \ \forall t \tag{7}$$

To evaluate the implementation of the novel algorithm we use a real world trace from NorduGrid [21]. This trace represents the load of the NorduGrid grid for nearly 3 years. With this trace, and using the equations (5) and (7) we get the number of requested instances R(t) and the number of new instances N(t) every 10 min. Figure 1 shows this values for the 3 year period.

This information is the base for the statistical analysis.

B. Statistical data model

To calculate the optimum number of reserved and provisioned instances, it is necessary to know the average usage of each instance. The instance reservation period is 1 year; hence, we use the normal distribution of the requested instances over 1-year to obtain the average utilization [22].

$$f_x(R(t)) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{1}{2\sigma^2}(R(t)-\mu)^2}, R(t)\epsilon[0,...L] \tag{8}$$

Where μ is the average requested instances, σ is the variance of requested instances and R(t) is the number of requested instances that can be any number from 0 to L.

However, this is not all the statistical information we have. The number of new instances can be statistically modelled as a Poison distribution if we assume that the number of users is large [22].

We model the number of new instances in the period λt with:

$$p_t(N(t)) = \frac{(\lambda t)^{N(t)}}{N(t)!} \tag{9}$$

where N(t) is the number of new instances at the moment and λt is the expected number of new instances in provisioning interval.

This last equation predicts the required number of provisioned instances. The provisioned instances are the instances that are started-up and ready to use. If the number of requested instances is lower than the number of reserved instances, all the provisioned instances will be mapped on reserved instances. Otherwise, the difference would be fulfilled with on-demand instances.

The reason to calculate the expected new instances for the following provisioning interval, and provision that value is to hide the performance degradation that the 2-5 minute launching time [1] might cause.

V. Reservation and Provisioning Algorithm

In the previous section, a statistical analysis of the historical data has been presented. Using that statistical analysis, we present two algorithms to determine the reservation and provisioning values. These algorithms use the historical load data of the IT companies and make the load predictions from different time periods.

1) With the long-term load prediction, the algorithm chooses the optimum number of reserved instances to reduce the cost of the service. The reason to use a long-term prediction is that reserved instances have 1-year utilization.
2) With the short-term load prediction, the algorithm chooses the optimal number of provisioned resources dynamically to hide launching delays. The algorithm start-up the instances the IT companies might need to offer the service to his clients.

A. Reserved Instances

In this section, we present a solution of the reservation problem.

As previously mentioned, due to the importance Amazon has on the cloud computing market, this is the cloud provider that will be used for the study.

We use the expected Differential Reservation Cost (DRM) to determine the reserved instances. Any time the expected differential reservation cost is negative, statistically reserving a new instance will reduce the final cost of deploying the service. On the other hand, if the expected differential reservation cost is positive, reserving a new instance will probably increase the final cost of deploying the service.

The expected DRC represents the difference in the expected statistical value of the cost of reserving one more instance in the cloud provider.

 ISBN: 978-1-61208-153-3

In this section, we explain all the steps given to obtain the optimum number of reserved instances. First, we introduce the expected differential reservation cost and then based on the statistical analysis, we develop a reservation algorithm that minimizes the cost.

1) Expected differential reservation cost: Suppose the IT company has already reserved n instances in the cloud provider. The IT company will provision the following instance only if more than n instances are provisioned.

From the statistics obtained from the users historical workload, the probability of provisioning more than n instances $\rho_{>n}$ can be obtained.

The expected differential reservation cost is:

$$\Delta P_{R:n+1} = Pres - (Res_{hours} \times \Delta P_c \times \rho_{>n}) \quad (10)$$

where $\Delta Pc = Pc_2 - Pc_1$ is the difference in the provisioning cost between reserved and on-demand instances, $\rho_{>n}$ is the probability of having more than n provisioned instances, and Res_{hours} is the period (in hours) that the instance is reserved.

Let us see this with one example: The cost of a standard instance in Amazon is 0.085$/h for on-demand and 0.03$/h for reserved instances. The instances are reserved for 1 year (8760h) paying 227.5$ for this reservation. Imagine that the provisioned instances follow a normal distribution with a mean of 100 and a variance of 10.

An iteration starts checking the first machine to see if is worthwhile reserving based on the expected DRC. In order to make the example concise, we show only the two key iterations:

- Iteration $n°1$ to 100:

$$\Delta P_{R:n} \ll 0 \quad n = 1, 2, ...100 \quad (11)$$

  At the end of the reservation period the IT company expects to pay a lot less in each iteration. In this experiment, the IT company pays a total of 74192.3$ with no reserved machines and 50929.7$ with 100 reserved machines.

- Iteration $n°$ 101: The DRC off adding the 101-th reserved machine is:

$$\Delta P_{R:101} = 227.5\$ - (8760h \times (0.085\$/h - 0.03\$/h) \times (1 - normcdf(100, 100, 10))) = -13.4\$ \quad (12)$$

  where normcdf is the normal cumulative distribution function with the values: test value, mean and variance. At the end of the reservation period the IT company expects to pay 13.4$ less to the cloud provider than reserving 100 machines. In this experiment, it pays a total of 50923.8$ or 5.9$ less than with 100 reserved machines. Hence, it is worthwhile to reserve the 101-th machine.

- Iteration $n°$ 102: The DRC off adding the 102-th reserved machine is:

$$\Delta P_{R:102} = 227.5\$ - (8760h \times (0.085\$/h - 0.03\$/h) \times (1 - normcdf(101, 100, 10))) = +5.8\$ \quad (13)$$

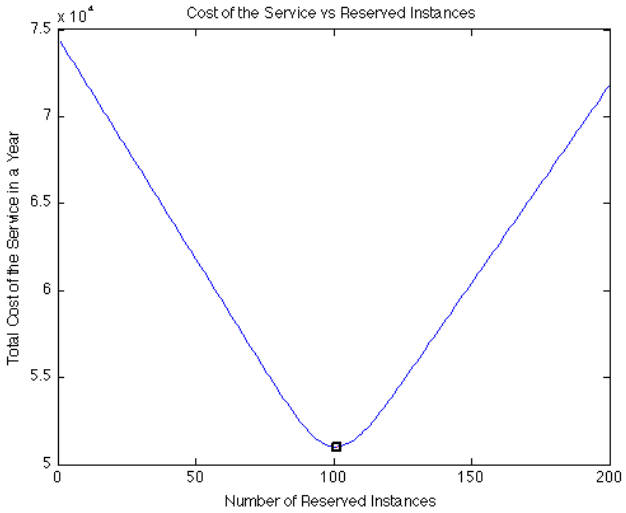


Fig. 2. Total cost in a year vs the number of reserved instances

This time the IT company expects to pay more to the cloud provider because the expected DRC of the 102-th machine is positive. In this experiment it pays a total of 50938$ or 14.2$ more than with 101 reserved reserving 102 machines. The 102 instance will not reach the minimum number of hours that make the reservation worthwhile. Hence, the reservation cost will be higher than the discount obtained from the price difference.

The algorithm conclude that reserving 101 machines is the most economical option. The Figure 2 shows the final cost of the service for the IT company in this experiment after a year with different number of reserved instances.

To make the algorithm faster, the implemented method does not calculate the expected DRC. The method just gets the last reserved instance with a negative expected DRC.

To obtain that number of reserved instances, the method uses the limiting percentage of utilization that provide a negative value of the DRC.

2) Reservation Algorithm: Applying the previous statistics and the utilization value that makes the expected DRC negative we have that:

$$F_x(Res(t)) = \int_{Res(t)}^{\infty} \frac{1}{\sqrt{2\pi\sigma(t)^2}} e^{-\frac{1}{2\sigma(t)^2}(R(t)-\mu(t))^2} dR(t) = \rho_{res}^{min} \quad (14)$$

where Res(t) is the number of reserved instances, μ is the mean of R(t) in the last year, σ is the variance of R(t) in the last year and ρ_{res}^{min} is the minimum load to make the expected DRC negative which in Amazon is 47% [8].

This equation does not get the expected DRC of each instance, however, it calculates which is the last instance that has a negative expected DRC.

In Amazon, we can only change Res(t) to a higher value and it is necessary to wait one year to reduce. Hence, the

following algorithm is used to determine the Res(t) at each moment.

Algorithm 2 Instance reservation algorithm

if $Res(t) \geq Res(t-1)$ **then**
  $Res(t)$
else $\{Res(t) < Res(t-1)$ and No reservation expires$\}$
  $Res(t) = Res(t-1)$
else $\{Res(t) < Res(t-1)$ and n reservations expire$\}$
  if $Res(t) \leq (Res(t-1) - n)$ **then**
    $Res(t) - n$
  else
    $Res(t)$
  end if
end if

In the next section, we present the results for reserved and provisioned instances.

B. Provisioned Instances

In this section, we present a solution for the provisioning problem. This problem has a direct relationship with the possible performance degradation due to the 2-5 minute launch time of new instances [1]. In this paper, we set this parameter to 0.05% because this percentage will produce a negligible performance degradation (the new provisioned instances will be ready to use in 99.95% of the time).

With the Poisson distribution represented in (9), the algorithm sets the value of provisioned instances P as

$$P_t(P(t) - R(t-1)) = \sum_{P_2(t)-R(t)}^{\infty} \frac{(\lambda t)^N(t)}{N(t)!} = \rho_{up}^{opt} \quad (15)$$

where $P(t) - R(t-1)$, is the difference between the requested and the provisioned instances. The reason to use this difference is that this statistical distribution calculates expected new instances at t.

C. Results

The optimum algorithm would provision in advance exactly the same instances that the ones requested and reserve instances in advance, only with more than 47% of load, as explained in the Section V-A. However, this is impossible because it means that the IT company knows his computation needs in advance.

Knowing which is the optimum result, the closer the algorithm is to this result the better it is. A good algorithm is the one that provision close to the requested instances, but always provisioning more than the requested instances, because if it provisions less a possible performance degradation occurs.

Figure 3 shows the number of provisioned and reserved instances compared to the number of requested instances at each moment.

In Figure 4, the percentage of time in which underprovisioning occur is shown. This is a way to show the

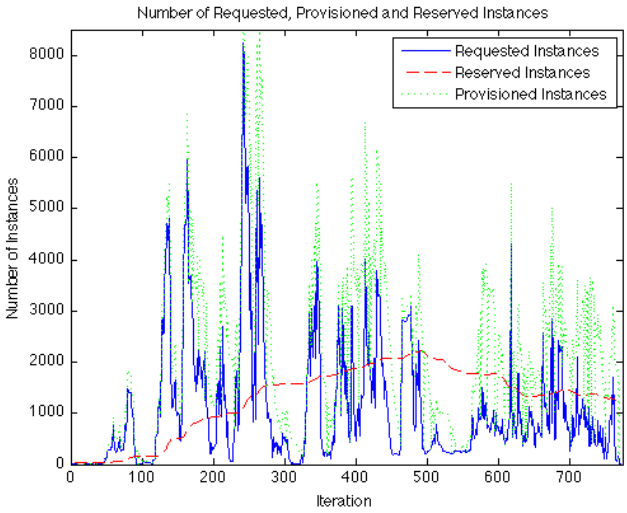


Fig. 3. Provisioning and reserved instances vs requested instances

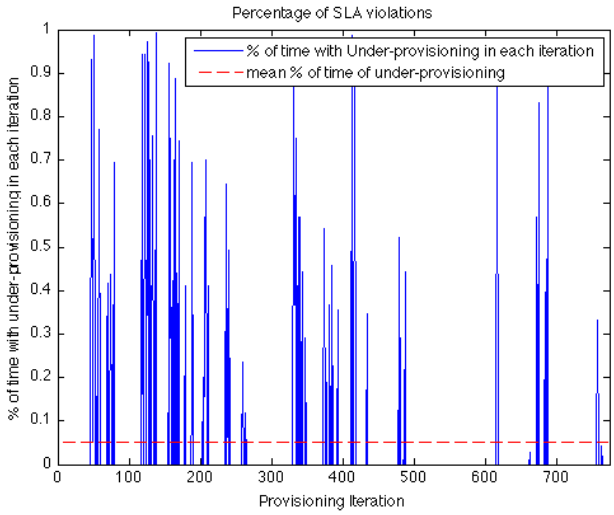


Fig. 4. Percentage of underprovisioning in each provisioning period

performance loss that occur in the system in a standardized way. The mean time of under-provisioning is 0,05%. This was the goal when defining the provisioning value and hence, the prediction model is accurate.

One of the most influential factors is the cost of the service for the user. This cost is the one that would determine if the service is competitive or if another solution is preferable.

To show this factor the cost for the user of using this algorithm has been recorded and compared with Amazons on-demand instances. Figure 5 shows the cost for the service user of using the broker.

At the end of the period the IT company have spent $5x10^5$ dollars or 32% less in cloud computing services.

VI. Improvements

In this section, we present the improvements of the basic prediction model and test the performances of each improve-

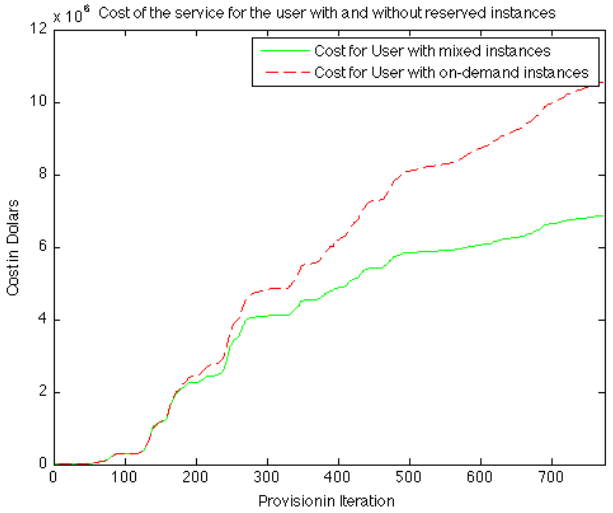


Fig. 5. Cost of the service for the user.

ment.

A. Reconfiguration based on under-provisioning

If an under-provisioning occur, the actual provisioned instances and requested instances should be analysed to see if there should be a change in the provisioned instances. This is extremely crucial to avoid performance degradation.

Even if the prediction model forecasts the necessary provisioned instances for a certain probability of under-provisioning there may be a change in the user patterns at a certain moment. If this change in the patterns creates continuous under-provisioning the performance of the service drops [23]. When under-provisioning occur, the algorithm recomputes the number of provisioned instances as the number of requested instances at the moment plus the number of expected new instances in the prediction interval.

B. Close control loop

The second improvement was focused in adjusting the statistic and prediction intervals to determine which was the one with the smaller prediction error in the expected under-provision value.

The statistic interval is the period in which the algorithm apply the Poisson distribution to predict the future values. The prediction interval is the period in which this future values are predicted.

The results presented in Table I represent the average error in the prediction of the algorithm at the end of the trace.

What we see here is that the error is not the same for different statistic and prediction ranges. If the predictions were perfect, they all should present the same error value and this value should be zero.

$$|\rho_{up}^{opt} - \rho_{alg}| = 0 \quad (16)$$

TABLE I
MEAN ERROR OF THE PREDICTOR AT THE END OF THE TRACE CONSIDERING DIFFERENT STATISTIC AND PREDICTION RANGES

Statistics / Prediction	1 day	2 days	3 days	1 week	2 weeks	3 weeks
3 hours	0.025	0.024	0.023	0.022	0.019	0.019
6 hours	0.023	0.024	0.02	0.02	0.019	0.018
9 hours	0.024	0.019	0.019	0.019	0.017	0.013
12 hours	0.02	0.013	0.014	0.012	0.014	0.012
18 hours	0.025	0.021	0.021	0.017	0.014	0.007
24 hours	0.013	0.02	0.011	0.02	0.012	0.002

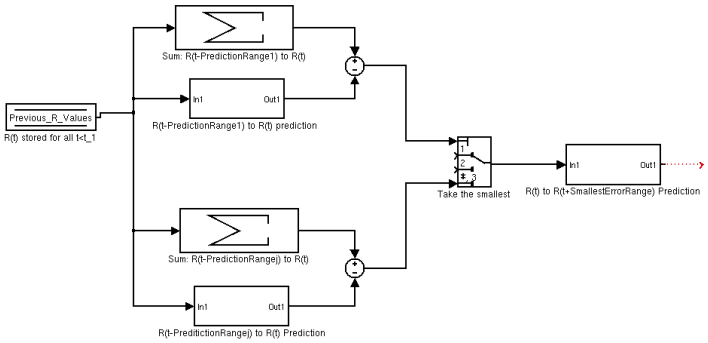


Fig. 6. Close control loop that sets optimum ranges

where ρ_{up}^{opt} is the probability of under-provisioning for which the algorithm is designed, and ρ_{alg} the real under-provisioning it achieves.

In this experiment, it is clear that the bigger the statistics and prediction ranges we take, the smaller the error is, but this can not be applied in all cases. To solve that situation, we present a tool that automatically set the optimum historic and prediction range.

The algorithm first select many different statistic and prediction ranges in a near past. With the results, it compares the errors, and it applies the statistic and prediction range that generates the smallest error for the next prediction .

With this method, we get the expected probability of under-provision with the best accuracy.

This is useful because the number of requested instances may exhibit a variance in the statistics with the time. Figure 6 shows a schematic view of the close control loop that the algorithm implements to select the best ranges.

The algorithm used to implement this control loop is explained in the following lines.

Algorithm 3 Close control loop algorithm

```
for i = 1 : Statistic ranges do
  for j = 1 : Prediction ranges do
    | ρ_opt − ρ_real |= error_ij
    if error_ij < errormin then
      errormin = error_ij → best_ij
    end if
  end for
end for
```

Applying this solution to the trace the average under-provisioning obtained at the end of the trace is remarkably close to 0,05%.

VII. Conclusion and future work

This work studies the reservation and provisioning values that minimize the cloud computing service cost with a controlled performance degradation. To reduce the cost, the algorithm uses mixed on-demand and reserved instances a single cloud provider.

We tested the algorithm that reserve and provision dynamically with real world traces obtained from the Grid Workload Archive, and compared the result after different improvements. The results show that the IT companies reduce their cost of service deployment by up to 32% with less than 0.05% performance degradation.

As future work, we will consider flexible reservation periods with different discounts. As well as, other kinds of workloads to determine, how the statistic models change and how are results altered.

VIII. Acknowledgment

The research leading to these results has received funding from the European Unions Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n^o 261552 (StratusLab); from Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional, and Fondo Social Europeo through MEDIANET Research Program S2009/TIC-1468; and from Ministerio de Ciencia e Innovación of Spain through research grant TIN2009-07146.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Tech. Rep., 2009.

[2] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky computing," *Internet Computing, IEEE*, pp. 43 –51, sept 2009.

[3] E. Walker, "The real cost of a cpu hour," University of Texas at Austin, Tech. Rep., April 2009.

[4] M. D. Assunção, A. D. Costanzo, and R. Buyya, "Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters," *Proceedings of the 18th ACM international symposium on High Performance Distributed Computing*, pp. 141–150, 2009.

[5] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Multi-cloud deployment of computing clusters for loosely-coupled mtc applications," *Transactions on Parallel and Distributed Systems*, 2010.

[6] M. de Assunção, A. di Costanzo, and R. Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters," *Cluster Computing*, Jan 2010.

[7] R. Harms and M. Yamrtino, "EU Public Sector Cloud Economics," Microsoft, Tech. Rep., 2011.

[8] "Amazon pricing web page, http://aws.amazon.com/ec2/pricing," May 2011. [Online]. Available: http://aws.amazon.com/ec2/pricing

[9] "The grid workload archieve web page, http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Main.Home," May 2011. [Online]. Available: http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Main.Home

[10] "Gogrid pricing web page, http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.php," May 2011. [Online]. Available: http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.php

[11] "Rackspace pricing web page, http://www.rackspacecloud.com/cloud_hosting_products/servers/pricing/," May 2011. [Online]. Available: http://www.rackspacecloud.com/cloud_hosting_products/servers/pricing/

[12] "Cloud sigma pricing web page, http://www.cloudsigma.com/en/pricing/price-schedules," May 2011. [Online]. Available: http://www.cloudsigma.com/en/pricing/price-schedules

[13] "Elastichost pricing web page, http://www.elastichosts.com/cloud-hosting/pricing," May 2011. [Online]. Available: http://www.elastichosts.com/cloud-hosting/pricing

[14] M. Armbrust, A. Fox, R. Griffith, and A. Joseph, "mOSAIC," European Commission: Information Society and Media, Tech. Rep., May 2010. [Online]. Available: www.mosaic-cloud.eu

[15] "Optimis home page, http://www.optimis-project.eu/content/welcome-optimis," May 2011. [Online]. Available: http://www.optimis-project.eu/content/welcome-optimis

[16] B. Sotomayor, "A resource management model for vm-based virtual workspaces," Master's thesis, The University of Chicago, 2007.

[17] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity leasing in cloud systems using the opennebula engine," *Workshop on Cloud Computing and its Applications*, October 2008.

[18] K. Konstanteli, D. Kyriazis, T. Varvarigou, T. Cucinotta, and G. Anastasi, "Real-Time Guarantees in Flexible Advance Reservations," in *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, 2009, pp. 67–72.

[19] P. Gardfjäll, E. Elmroth, L. Johnsson, O. Mulmo, and T. Sandholm, "Scalable grid-wide capacity allocation with the swegrid accounting system (sgas)," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 2089–2122, December 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1458640.1458641

[20] "Amazon instances web page, http://aws.amazon.com/ec2/instance-types/," May 2011. [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[21] "Nordugrid trace web page, http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-3/trace_analysis_report.html," May 2011. [Online]. Available: http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-3/trace_analysis_report.html

[22] G. Zhao, J. Liu, Y. Tang, W. Sun, F. Zhang, X. Ye, and N. Tang, "Cloud computing: A statistics aspect of users," *Cloud Computing*, pp. 347–358, 2009.

[23] V. Machiraju, M. Sayal, A. V. Moorsel, and F. Casati, "Automated sla monitoring for web services," *IEEE International Symposium on Integrated Network Management*, pp. 28–41, 2002.

 ISBN: 978-1-61208-153-3

Debit: A Diversity-based Method for Implicit Role Transition in RBAC Deployments

Shanshan LI, Qingbo WU, Lianyue HE, Lisong SHAO, Jie YU
School of Computer
National University of Defense Technology
Changsha, China
{shanshanli, qingbo.qu, lianyuehe, lisongshao, jieyu} @nudt.edu.cn

***Abstract*-Role-based access control (RBAC) is a widely used access control paradigm in operating system due to its simplicity, scalability and fine-grained control ability. Current approaches need re-login to transit role when the permissions of assigned role are inadequate for operation. This usage is easy for secure administration, while inflexible in practical use, especially for those authenticated users. This paper describes a diversity-based access control model supporting implicit role transition, called DRT-RBAC. By measuring users' authentication trustworthiness, a range for role transition can be computed, and user whose diversity between the old role and the new one fall into this range is allowed for automated role transition. Further, we propose Debit, which calculates the diversity between roles in operating system through an analytic hierarchy process. In Debit, the roles are decomposed to fine grained system privileges, capability. Debit computes a weight for each category of capability through constructing a pair wise comparisons matrix. The diversity of two roles is finally obtained based on the weight of each capability category and the number difference of capabilities on the category. We implement Debit in Centos 5.4 to support implicit role transition based on Authentication Trustworthiness of login user.**

Keyword-DRT-RBAC; authentication trustworthiness; Debit.

I. INTRODUCTION

Access control is an indispensable component of operating system, which mediates requests to resources of the system and makes decisions about whether or not they should be granted. Relative to Classical Discretionary Access control (DAC), Mandatory Access Control (MAC), Role-based Access Control (RBAC) model is more emphasized recently due to its simpleness, scalability, fine-grained control ability, and has been proven to be efficient to improve security administration with flexible authorization management. In RBAC, users are assigned to roles, and permissions are granted to roles. The protection state is characterized by the triple <UA, PA, RR>, where UA is the user-role assignment relation, PA is the permission role assignment relation and RH is a role composition in systems. RBAC can greatly simplify the management of authorizations within a system, because a group of subjects are usually given the same permissions.

For many mainstream operating systems, a user is generally assigned a role either selected in system authentication module or based on the least privilege principle. For ease of secure administration, once the permissions of assigned role are inadequate for operation, the user need re-login and select another role from his available list. Actually, if user can pass strong authentication, he is well trustworthy and should be allowed to transit role transparently. Current usage of roles requires manual intervene of users, thus inflexible in practical use.

In this paper, we investigate a diversity-based access control model supporting implicit role transition, called DRT-RBAC. DRT-RBAC model associates the strength of authentication trustworthiness with a transition range of role, which takes the diversity between roles as the decision condition to transit role. Only those users whose diversity between the old role and the new one fall into the transition range can make transition implicitly. The model keeps the advantage of permission management, while emphasizes on the flexibility of user-role assignment and makes operating system friendly to users.

Based on DRT-RBAC model, we propose Debit, an analytic hierarchy process to measure the diversity between roles in operating system. Debit analyzes the inherent factors which result in the difference among roles, and constructs a hierarchy with fine-grained system privileges, capability, each layer is analyzed independently. Through constructing a pair wise comparisons matrix, Debit computes a weight for each category of capability. The diversity of two roles is finally obtained based on the weight of each capability category and the number difference of capabilities on the category.

The rest of this paper is organized as follows. We briefly review the related work in Section 2. In Section 3, we present some basic knowledge, including the concept of authentication trustworthiness in single and multiple authentication mechanisms. In Section 4, we describe the diversity-based RBAC model DRT-RBAC supporting implicit role transition, and present an analytic hierarchy process Debit to calculate diversity. In Section 5, we implement Debit in Centos 5.4 and verify its effectiveness. We conclude the work in Section 6.

II. RELATED WORK

One of the most challenging problems in managing operating systems is the complexity of security administration. Role-based access control has become the predominant model for advanced access control since it reduces the cost of security management. There has been

 ISBN: 978-1-61208-153-3

much work done to explore the role assignment, time constraint and security controlled mobility to enhance the network performance.

Odell and Parunak [1] found that an important characteristic of real-world systems is that the roles of subject may change over time. These changes can be of several different kinds. They analyze and classify the various kinds of role changes over time that may occur, and show how this analysis is useful in developing a more formal description of the application. Liao and Hong [2] found that IRBAC 2000 model [3] had not considered the separation of duties, and they analyze the scenarios where dynamic role translations violate statically mutually exclusive role constraints, then propose a protective mechanism utilizing prerequisite conditions to enforce the security of the IRBAC 2000 model. These works provide guide for role transition among multiple domains in theory; however, they are not fit for local role transition, especially for operating system.

Some works consider role transition from temporal and spatial perspective [4-7], that is, roles of subject may change in different time periods and environments. Bertino et al. proposed the Temporal-RBAC (TRBAC) model that addresses some of the temporal issues related to RBAC [8]. The main features of this model include periodic enabling of roles and temporal dependencies among roles which can be expressed through triggers. James, et al. argued that TRBAC model addresses the role enabling constraints only. They proposed a Generalized Temporal Role-based Access Control (GTRBAC) model capable of expressing a wider range of temporal constraints [9]. In particular, the model allows expressing periodic as well as duration constraints on roles, user-role assignments, and role-permission assignments. Joshi and Ghafoor [10] showed how RBAC can be extended to incorporate environmental contexts, such as time and location.

For remote access control, a few models have been proposed [11-13], which benefit from the advantages of both RBAC and trust management systems in an open environment. In particular, the TrustBAC model [12] supports automatic user-role assignment based on not only credentials of a stranger but its past behavior and recommendations. Saffarian et al. proposed a new dynamic user-role assignment approach for remote access control [14]. It addresses the principle of least privilege without degrading the efficiency of the access control system. Moreover, it takes into account both credentials and the past behavior of the requestor in such a way that he cannot compensate for the lack of necessary credentials by having a good past behavior.

Due to the uncertainty of execution time and task allocation, the methods mentioned above cannot fit well access control in operating systems.

III. BACKGROUND CONCEPT

For most secure operating systems, user is treated as trustworthy if he passes the authentication mechanism. This principle, however, is hard to apply for current uses. In one side, hackers may obtain the authentication credence of users and login system bypassing the authentication module, obviously, these hackers cannot be regarded as trusted users. In the other side, trustworthiness is a value of experience and should differ in different authentication mechanism.

In our previous work [15], we borrowed the idea of uncertainty reasoning in expert system and proposed a reasoning model for measuring authentication trustworthiness. In this paper, we associate the authentication mechanism with access control in supporting automated role transition.

Definition 1. **Authentication Trustworthiness:** the trustworthy degree of the subject who has passed system authentication, denoted by $t_{au}(u)$. The value of $t_{au}(u)$ is between 0 and 1. The larger the value is, the more the degree of trustworthy is.

$$t_{au}(u) = p(H \mid E) \tag{1}$$

H denotes that user is trustworthy and E is the authentication mechanism. The precondition E is independent of H. When user selects a role r, the authentication trustworthiness of current role inherits that of the user, that is, $t_{au}(u) = t_{au}(r)$.

Definition 2. **Trustworthiness Increase Degree** reflects the trustworthy increase after passing the system authentication, denoted by $ASTF(H, E)$, E is the system authentication mechanism.

$$ASTF(H, E) = \begin{cases} \dfrac{p(H \mid E) - p(H)}{1 - p(H)} & p(H) < 1 \\ 0 & p(H) = 1 \end{cases} \tag{2}$$

Normally, $ASTF(H, E)$ is given by experience, with (2), we can get the Authentication Trustworthiness by (3):

$$p(H \mid E) = ASTF(H, E) + (1 - ASTF(H, E))p(H) \tag{3}$$

IV. DIVERSITY-BASED ROLE TRANSITION

In this section, we investigate a diversity-based implicit role transition method in RBAC model. As we know, explicit role transition needs user intervene thus inflexible for application, while complete implicit transition may result in privilege management out of control. In order to reduce the risk while keeping flexibility, we introduce the DRT-RBAC model, which enforce some restriction on implicit role transition. According to the strength of authentication mechanism, a range for role transition can be computed, and users whose diversity between the old role and the new one fall into this range are allowed for automated transition.

Based on DRT-RBAC model, a diversity measurement method, Debit, is further proposed for real system deployments.

 ISBN: 978-1-61208-153-3

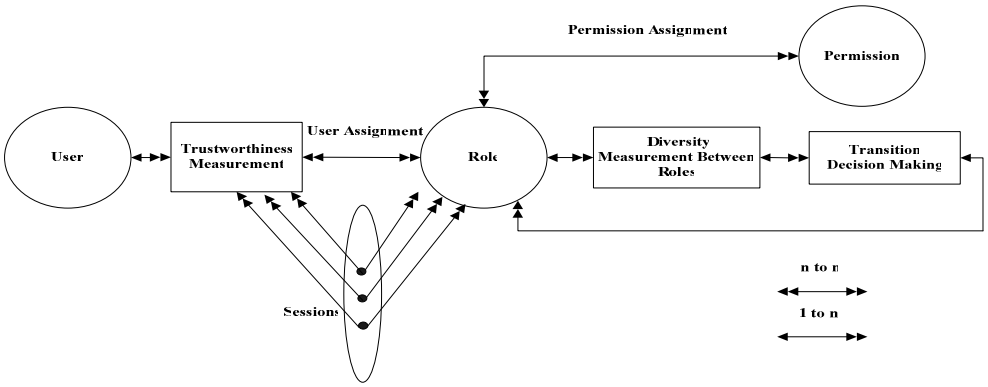


Figure 1. DRT-RBAC Model

A. *DRT-RBAC Model*

DRT-RBAC inherits basic elements from RBAC96 model and makes some extensions, as illustrate in Figure 1.

Similar to RBAC96 model, users are assigned to roles and the roles are mapped to permissions. While distinguishingly, DRT-RBAC adds a new concept of role diversity and support automated role transition. A user is usually assigned several roles in a given system, and only selects one in login. Automated role transition means a user can transit to that role implicitly if the current role of the user has no the privilege for current operation, while another available role has the corresponding privilege.

Definition 4. **Role Diversity:** the difference between roles, denoted by $d(r_1, r_2)$.

Automated role transition do not need manual intervene and largely enhance the flexibility of user operation. While unlimited transition may render system security, we enforce some restriction on automated transition, only those whose diversity between the old role and the new one fall into a transition range are allowed for automated transition. Transition range is decided on his authentication trustworthiness. Basically, the stronger the authentication mechanism, the larger the transition range. Transition threshold defines the maximum transition range. DRT-RBAC model keeps the advantage of permission management, while emphasizes on the flexibility of user-role assignment and made operating system friendly to users. Figure 2 illustrates the role transition decision process.

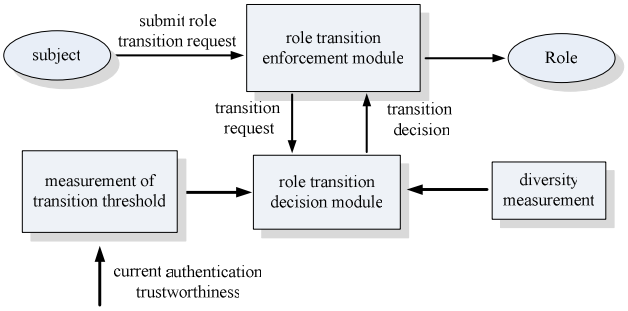


Figure 2. Role Transition Decision Process

Definition 5. **Transition threshold:** the maximum role diversity in current authentication mechanism, denoted by $d_{\max}(u)$.

Rule 1. Role transition rule: With authentication trustworthiness of $t_{au}(r_1)$, user can transit role implicitly from r_1 to r_2 if $d(r_1, r_2) < d_{\max}(u)$.

The role transition decision module is the centre part in the transition process. We will give the measurement of its input, role diversity and transition threshold, in the following sections.

B. *Debit Design*

Basically, role diversity can be measured from many aspects. In operating system, capability differentiates roles on system privilege and is a good reflector on role diversity; therefore, we proposed a capability based method named Debit to measure role diversity in this paper.

Different capabilities weigh differently since each of them has different effect on system, such as system management, security management, network management and so on. In order to measure role diversity accurately, Debit uses an analytic hierarchy process [16], in which we have two layers, capability and role. Through constructing a pair wise comparisons matrix, Debit calculates a weight for each category of capability. The diversity of two roles is finally obtained based on the weight of each capability category and the number difference of capabilities on the category.

Supposed we have k roles and n capabilities. Debit works as followed:

(1) Capability categorization

According to their function, capabilities are classified into m categories, $T_1, T_2, \ldots\ldots, T_m$, let CN_i^j be the number of T_j capabilities in $role_i$.

(2) Constructing Pair wise comparisons matrix in capability layer

TABLE I. THE FUNDAMENTAL SCALE FOR PAIR WISE COMPARISONS

Intensity of importance	Definition	Explanation
1	Equal importance	Two elements contribute equally
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Strong importance	Experience and judgment strongly favor one element over another
7	Very strong importance	One element is favored very strongly over another, its dominance is demonstrated in practice

Through comparing the effect of each category on operating system, we construct a pair wise comparisons matrix. Pair wise comparing matrix reflects the intensity of importance between each pair of capability categories. The scale of the intensity is referenced from table I. The pair wise comparisons matrix of capability is shown in figure 3, a_{ii} shown is one of the value 1, 3,5,7,9 or it's reciprocal, $a_{ii}=1,\ a_{ij}=1/a_{ji}$.

(3) Checking Consistency

Debit should check the consistency of the pair wise comparisons matrix. The reason which results in the inconsistency is the improper decision of the intensity of importance between each pair of capability categories.

TABLE II. RI REFERENCED VALUE

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

Rule 2. Consistency checking rule: the pair wise comparisons matrix is consistent if $a_{ij}a_{jk}=a_{ik}$, $1 \le i, j, k \le n$ or the maximal matrix eigenvalue equal to its order.

$$V=\begin{array}{c} \\ T_1 \\ T_2 \\ \\ T_{m-1} \\ T_m \end{array}\begin{array}{c} \begin{array}{ccccc} T_1 & T_2 & \cdots & T_{m-1} & T_m \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m-1} & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m-1} & a_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m-11} & a_{m-12} & \cdots & a_{m-1m-1} & a_{m-1m} \\ a_{m1} & a_{m2} & \cdots & a_{mm-1} & a_{mm} \end{bmatrix} \end{array}$$

Figure 3. Pair wise comparisons matrix of capability

Basically, incomplete consistency is acceptable in some extend. Debit uses (4) to judge whether the matrix has a satisfying consistency. If $CR < 0.1$, it's acceptable, else we should adjust the matrix V until satisfying.

$$CR = CI / RI \qquad (4)$$

CI is computed through (5), and RI is obtained from table II.

$$CI = \frac{\lambda_{\max}(V)-n}{n-1} \qquad (5)$$

(4) Computing weight for each category of capability

Debit computes the maximal matrix eigenvalue W, which is corresponding to the weight of each capability category.

(5) Measuring diversity for roles

In role layer, in order to measure the diversity between two roles, for example, role *a* and role *b*, we need to construct one pair wise comparisons matrix for each capability category, thus we are able to measure the difference on each capability category between role *a* and role *b*. And their diversity is finally got through the weighed summation of these differences.

For each category of capability, we construct a pair wise comparisons matrix for each pair of roles. In these matrixes, the intensity of importance is decided by their number difference of each capability category, $CN_i^h - CN_i^k$, and also referenced from table I. For role *i*, capability category *h, k,* let its matrix eigenvalue is $W_i=(b_{ih},b_{ik})$, then the diversity between role *i* and role *j* is:

$$D(R_i,R_j)=\sum_{n=1}^{m} W \times |(b_{in}-b_{jn})| \qquad (6)$$

C. Transition Threshold

In this paper, we use authentication trustworthiness to get the transition threshold. Let the maximum authentication trustworthiness is 1, the transition threshold of current authentication mechanism is in proportion to the corresponding authentication trustworthiness. User who needs implicit role transition checks the diversity between two roles; and only those whose diversity between the old role and the new one fall below the threshold are allow for implicit transition.

V. IMPLEMENTATION IN CENTOS 5.4

We implement DRT-RBAC model and Debit in Centos 5.4. The kernel version is linux 2.6.18, in which we have 31 capabilities. Each bit of the low 32 bits denotes one capability and the high 32 bits are left for extension. We

TABLE III. THE $ASTF(H,E)$ UNDER DIFFERENT AUTHENTICATION MECHANISMS

authentication mechanism	$ASTF(H, E)$
password	0.1
u-key	0.3
fingerprint	0.6

 ISBN: 978-1-61208-153-3

implement three authentication mechanisms, which are password, u-key and fingerprint. Their trustworthiness increase degree is set in table III.

According to (4), we are able to get the authentication trustworthiness of each authentication mechanisms, and set it in the structure of current active task by PAM module.

We set five roles in Centos 5.4, which is system admin, security admin, audit admin, net admin and default role. The capability of each role is illustrated using hexadecimal mode in table IV. The triples represent inherit (i), permitted (p) and effective (e) capability respectively. In general, the execute capability of a process denotes the active capability, and is inherited from the inherit capability of its role. Thus we use the first element in the triples of role, inherit capability, to measure diversity between roles.

TABLE IV. CAPABILITY OF EACH ROLE

role	Capability <i,p,e>
default role	<0，0，0>
net admin	<9800feff，0，0>
system admin	<9ffffeff，0，0>
security admin	<200006，0，0>
audit admin	<60810000，0，0>

We classify the 31 capabilities into 5 categories, which is system management (SYM), security management (SEM), audit management (AUM), net management (NEM) and routine (ROU). By weighing their importance, we get the pair wise comparisons matrix of capability. Table V gives the composition of capabilities of each category in each role.

This matrix is consistent, we get the normalized maximal eigenvalue, $\boldsymbol{W}$= (0.28894, 0.28894, 0.28894, 0.0802, 0.053). This is the relative weight of all roles.

TABLE V. CAPABILITIES OF EACH CATEGORY IN EACH ROLE

role	ROU	NEM	SYM	SEM	AUM
Default role	0	0	0	0	0
net admin	11	4	0	0	0
system admin	13	4	11	0	0
security admin	2	0	1	1	0
audit admin	0	0	2	0	2

In role layer, we construct several pair wise comparison matrixes for each pair of roles. Each matrix denotes their difference on each category of capability. The intensity of importance is decided on the number difference in Table 4, and the diversity between roles is finally obtained from (6).

In the pair wise comparison matrix, the maximal intensity of importance is 7, and thus we are able to compute the maximal diversity between roles, which is 0.75. Figure 5 illustrate the transition threshold for all pairs of roles.

	AUM	SEM	SYM	NEM	
AUM	1	1	1	4	5
SEM	1	1	1	4	5
SYM	1	1	1	4	5
NEM	1/4	1/4	1/4	1	2
ROU	1/5	1/5	1/5	1/2	1

Figure 4. Pair wise comparisons matrix of capability in CentOS 5.4

I. CONCLUSION

In this paper, we propose a diversity-based access control model DRT-RBAC. DRT-RBAC support implicit role transition according to the authentication trustworthiness of users. This model keeps the advantage of permission management, while emphasizes on the flexibility of user-role assignment and made operating system friendly to users. In our future work, we will consider the temporal factor affecting the transition on roles.

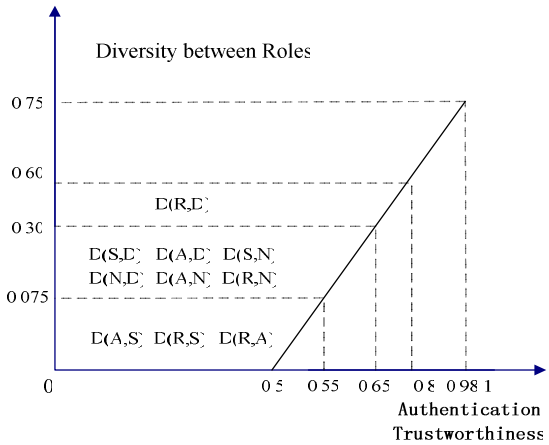


Figure 5. Transition threshold of each pair of roles

REFERENCES

[1] J. Odell, H.V.D. Parunak, S. Brueckner, and J. Sauter, "Changing Roles: Dynamic Role Assignment," Journal of Object Technology, vol 2, no 5, pp 77-86, 2003.

[2] J. Liao, X. Zhu, H. Xiao, "Separation of Duty in Dynamic Role Translations Between Administrative Domains," Journal of Computer Research and Development, pp. 43(6):1065~1070, 2006.

[3] A. Kapadia, J. Al-Muhtadi, R.H. Campbell, and M.D. Mickunas, "IRBAC 2000: Secure interoperability using dynamic role translation," In Proceedings of the 1st International Conference on Internet Computing, pp. 231-238, 2000.

[4] A. Samuel, A. Ghafoor, and E. Bertino, "A Framework for Specification and Verification of Generalized Spatio-Temporal Role-based Access Control Model," Technical report, Purdue University, CERIAS TR 2007-08, February 2007.

 ISBN: 978-1-61208-153-3

[5] V. Atluri and S.A. Chun, "A geotemporal role-based authorisation system," International Journal of Information and Computer Security, v.1 n.1/2, pp.143-168, 2007.

[6] S.M. Chandran and J.B.D. Joshi, "LoT-RBAC: A Location and Time-based RBAC Model," In Proceedings of the 6th International Conference on Web Information Systems Engineering, pp. 361-375, New York, NY, USA, November 2005.

[7] I. Ray and M. Toahchoodee, "A Spatio-Temporal Role-Based Access Control Model," In Proceedings of the 21th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pp.211-226, Redondo Beach, CA, July 2007.

[8] E. Bertino, P.A. Bonatti, and E. Ferrari, "TRBAC: A Temporal Role-based Access Control Model," In Proceedings of the 5th ACM workshop on Role-based access control, pp.21-30, Berlin, Germany, July. 2001.

[9] J.B.D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "A generalized temporal role-based access control model," IEEE Transactions on Knowledge and DataEngineering, pp.17(1):4~23, January 2005.

[10] I. Ray and M. Toahchoodee, "A Spatio-temporal Access Control Model Supporting Delegation for Pervasive Computing Applications," In Presented at Proceeding of the 5th international conference on trust, privacy and security in Digital Business, Turin, Italy, 2008.

[11] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor, "Access control meets public key infrastructure, or: Assigning roles to strangers," In Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp.2-9, Washington, DC,USA, 2000.

[12] S. Chakraborty and I. Ray, "TrustBAC: integrating trust relationships into the RBAC model for access control in open systems," In Proceedings of the eleventh ACM symposium on Access control models and technologies, New York, NY, USA, 2006.

[13] Y. Zhong, B. Bhargava, and M. Mahoui, "Trustworthiness based authorization on www," Department of Computer Science, Purdue University CERIAS Tech Report 2002-08, 2002.

[14] M. Saffarian, Q. Tang., W. Jonker, and P. Hartel, "Dynamic User-Role Assignment in Remote Access Control," CTIT-09-14, 2009.

[15] L. Wang, L. Wei, X. Liao, and H. Wang, "AT-RBAC: an Authentication Trustworthiness-based RBAC Model," In Proceeding of the 3rd Grid and Cooperative Computing -GCC 2004 Workshops, 2004.

[16] T.L. Saaty, "How to make a decision: the analytic hierarchy process," Interfaces, vol. 24, pp.19-27, 1994.

 ISBN: 978-1-61208-153-3

Trust Model for File Sharing in Cloud Computing

Edna Dias Canedo, Robson de Oliveira Albuquerque and Rafael Timóteo de Sousa Junior
Electrical Engineering Department– University of Brasília – UNB
Brasília – DF, Brazil, 70910-900.
E-mails: {ednacanedo@unb.br}{robson@unb.br}{desousa@unb.br}

***Abstract*—The recent advances in cloud computing have risen a number of unforeseen security related issues in different aspects of cloud environments. Among these, the problem of guaranteeing secure access to computing resources in the cloud is gathering special attention. In this paper, we address open issues related to trust in cloud environments proposing a new trust model for cloud computing which considers a higher level view cloud resources. A simulation of trust calculation between the nodes of the clouds is performed. The simulation was possible to verify that a node is reliable when it reaches the minimum index of trust.**

Keywords-Cloud Computing; Distributed Computing; Security; Integrity; Confidentiality; Trust and Availability.

I. INTRODUCTION

The widespread use of Internet connected systems and distributed applications has triggered a revolution towards the adoption of pervasive and ubiquitous cloud computing environments. These environments allow users and clients to purchase computing power according to necessity, elastically adapting to different performance needs while providing higher availability. Several web-based solutions, such as Google Docs and Customer Relationship Management (CRM) [2] applications, now operate in the software as a service model. Much of this flexibility is made possible by virtual computing methods, which can provide adaptive resources and infrastructure in order to support scalable on-demand sales of such applications. Virtual computing is also applied to stand-alone infrastructure as a service solutions, such as Amazon Elastic Cloud Computing (EC2) and Elastic Utility Computing Architecture Linking Your Programs to Useful Systems (Eucalyptus) [2].

As a result, the cloud computing frameworks and environments are able to address different issues in current distributed and ubiquitous computing systems.

The availability of infrastructure as a service and platform as a service environments provided a fundamental base for building cloud computing based applications. It also motivated the research and development of technologies to support new applications. As several large companies in the communications and information technology sector have adopted cloud computing based applications, this approach is becoming a de facto industry standard, being widely adopted by different organizations.

Since the adoption of the cloud computing paradigm by IBM Corporation around the end of 2007, other companies such as Google (Google App Engine), Amazon (Amazon Web Services (AWS), EC2 (Elastic Compute Cloud) and S3 (Simple Storage Service)), Apple (iCloud) and Microsoft (Azure Services Platform) have progressively embraced it and introduced their own new products based on cloud computing technology [11]. However, cloud computing still poses risks related to data security in its different aspects (integrity, confidentiality and authenticity).

Cloud computing provides a low-cost, scalable, location independent infrastructure for data management and storage. The rapid adoption of Cloud services is accompanied by increasing volumes of data stored at remote servers, so techniques for saving disk space and network bandwidth are needed. A central up and coming concept in this context is deduplication, where the server stores only a single copy of each file, regardless of how many clients asked to store that file. All clients that store the file merely use links to the single copy of the file stored at the server. Moreover, if the server already has a copy of the file, then clients do not even need to upload it again to the server, thus saving bandwidth as well as storage (this is termed client-side deduplication). Reportedly, business applications can achieve deduplication ratios from 1:10 to as much as 1:500, resulting in disk and bandwidth savings of more 90%. Deduplication can be applied at the file level or at the block level.

In a typical storage system with deduplication, a client first sends to the server only a hash of the file and the server checks if that hash value already exists in its database. If the hash is not in the database then the server asks for the entire file. Otherwise, since the file already exists at the server (potentially uploaded by someone else), it tells the client that there is no need to send the file itself. Either way the server marks the client as an owner of that file, and from that point on the client can ask to restore the file (regardless of whether he was asked to upload the file or not).

The client-side deduplication introduces new security problems. For example, a server telling a client that it need not send the file reveals that some other client has the exact same file, which could be sensitive information. A malicious client can use this information to check whether specific files were uploaded by other users, or even run a brute force attack which identifies the contents of certain fields in files owned by other users, by trying to upload multiple variants of the same file which have different values for that field. The findings apply to popular file storage services such as MozyHome and Dropbox, among others.

In this paper, we review the main cloud computing architecture patterns and identify the main issues related to security, privacy, trust and availability. In order to address such issues, we present a high level architecture for trust models in cloud computing environments.

This paper is organized as follows. In Section II, we present an overview of cloud computing, presenting a

 ISBN: 978-1-61208-153-3

summary of its main features, architectures and deployment models. In Section III, we present related works. In section IV, we introduce the proposed trust model. Finally, in Section V, we conclude with a summary of our results and directions for new research.

II. CLOUD COMPUTING

Cloud computing refers to the use, through the Internet, of diverse applications as if they were installed in the user's computer, independently of platform and location. Several formal definitions for cloud computing have been proposed by industry and academia. We adopt the following definition: "Cloud computing is a model for enabling 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" [14]. This definition includes cloud architectures, security, and deployment strategies.

Cloud computing is being progressively adopted in different business scenarios in order to obtain flexible and reliable computing environments, with several supporting solutions available in the market. Being based on diverse technologies (e.g. virtualization, utility computing, grid computing and service oriented architectures) and constituting a whole new computational paradigm, cloud computing requires high level management routines. Such management activities include: (a) service provider selection; (b) virtualization technology selection; (c) virtual resources allocation; (d) monitoring and auditing in order to guarantee Service Level Agreements (SLA).

Computational trust can be leveraged in order to establish an architecture and a monitoring system encompassing all these needs and still supporting usual activities such as planning, provisioning, scalability and security. Chang et al. [15] present a few challenges related to security, performance and availability in the cloud.

A. Characteristics of Cloud Computing

One advantage of cloud computing is the possibility of accessing applications directly from the Internet, with minor requirements of user computing resources. There are other significant advantages and disadvantages [13], as shown in Table I.

Cloud computing combines a shared and statistical service model. It presents three basic characteristics [1]: a) hardware infrastructure architecture – based on low cost scalable clusters. The computing infrastructure in the cloud is composed of a great number of low cost servers, such as standard X86 server nodes; b) collaborative development of basic services and applications with maximal resource utilization, thus improving traditional software engineering processes. In the traditional computational model, applications become completely dependent on the basic services; c) the redundancy among several low cost servers is guaranteed through software. Since a large number of low cost servers is used, individual node failures cannot be ignored. Therefore, node fault tolerance must be taken into account in the design of software.

TABLE I. ADVANTAGES AND DISADVANTAGES OF CLOUD COMPUTING

Advantages	Disadvantages
Lower IT infrastructure cost	Requires a constant Network connection
Increased computing power	Dependable of network bandwidth
Unlimited storage capacity	Features might be limited
Improved compatibility between operating Systems	Stored data might not be secure
Easier group collaboration	If the cloud loses your data, you will not have access to your information.
Universal access to documents	

B. Cloud Computing Architecture

Cloud computing architecture is based on layers. Each layer deals with a particular aspect of making application resources available. Basically there are two main layers: a lower and a higher resource layer. The lower layer comprises the physical infrastructure and is responsible for the virtualization of storage and computational resources. The higher layer provides specific services. These layers may have their own management and monitoring system, independent of each other, thus improving flexibility, reuse and scalability. Figure 1 presents the cloud computing architectural layers [11].

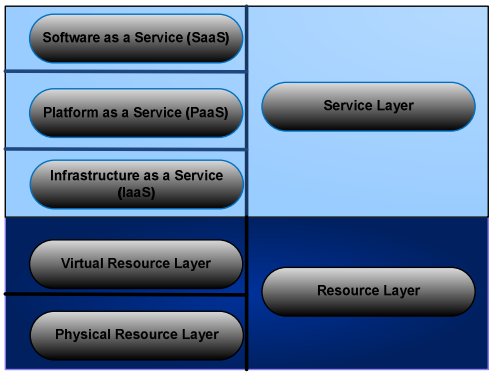


Figure 1. Cloud Computing Architecture [11]

C. Software as a Service

Software as a Service (SaaS) provides all the functions of a traditional application, but provides access to specific applications through Internet. The SaaS model reduces concerns with application servers, operating systems, storage, application development, etc. Hence, developers may focus on innovation, and not on infrastructure, leading to faster software systems development.

SaaS systems reduce costs since no software licenses are required to access the applications. Instead, users access services on demand. Since the software is mostly Web based, SaaS allows better integration among the business units of a given organization or even among different software services. Examples of SaaS include [2]: Google Docs and Customer Relationship Management (CRM) services.

 ISBN: 978-1-61208-153-3

D. Platform as a Service

Platform as a Service (PaaS) is the middle component of the service layer in the cloud. It offers users software and services that do not require downloads or installations. PaaS provides an infrastructure with a high level of integration in order to implement and test cloud applications. The user does not manage the infrastructure (including network, servers, operating systems and storage), but he controls deployed applications and, possibly, their configurations [4].

PaaS provides an operating system, programming languages and application programming environments. Therefore, it enables more efficient software systems implementation, as it includes tools for development and collaboration among developers. From a business standpoint, PaaS allows users to take advantage of third party services, increasing the use of a support model in which users subscribe to IT services or receive problem resolution instructions through the Web. In such scenarios, the work and the responsibilities of company IT teams can be better managed. Examples of SaaS [2] include: Azure Services Platform (Azure), Force.com, EngineYard and Google App Engine.

E. Infrastructure as a Service

Infrastructure as a Service (IaaS) is the portion of the architecture responsible for providing the infrastructure necessary for PaaS and SaaS. Its main objective is to make resources such as servers, network and storage more readily accessible by including applications and operating systems. Thus, it offers basic infrastructure on-demand services. IaaS has a unique interface for infrastructure management, an Application Programming Interface (API) for interactions with hosts, switches, and routers, and the capability of adding new equipment in a simple and transparent manner. In general the, user does not manage the underlying hardware in the cloud infrastructure, but he controls the operating systems, storage and deployed applications. Eventually he can also select network components such as firewalls.

The term IaaS refers to a computing infrastructure, based on virtualization techniques that can scale dynamically, increasing or reducing resources according to the needs of applications. The main benefit provided by IaaS is the pay-per-use business model [4]. Examples of IaaS [2] include: Amazon Elastic Cloud Computing (EC2) and Elastic Utility Computing Architecture Linking Your Programs To Useful Systems (Eucalyptus).

F. Roles in Cloud Computing

Roles define the responsibilities, access and profile of different users that are part of a cloud computing solution. Figure 2 presents these roles defined in the three service layers [3].

The provider is responsible for managing, monitoring and guaranteeing the availability of the entire structure of the cloud computing solution. It frees the developer and the final user from such responsibilities while providing services in the three layers of the architecture.

Developers use the resources provided by IaaS and PaaS to provide software services for final users.

This multi-role organization helps to define the actors (people who play the roles) in cloud computing environments. Such actors may play several roles at the same time according to need or interest. Only the provider supports all the service layers.

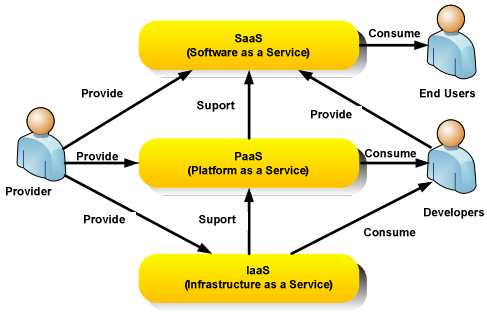


Figura 2. Roles in cloud computing [3].

G. Cloud Computing Deployment

According to the intended access methods and availability of cloud computing environments, there are different models of deployment [4]. Access restriction or permission depends on business processes, the type of information and characteristics of the organization. In some organizations, a more restrict environment may be necessary in order to ensure that only properly authorized users can access and use certain resources of the deployed cloud services. A few deployment models for cloud computing are discussed in this section. They include private cloud, public cloud, community cloud and hybrid cloud, which are briefly analyzed below.

TABLE II. MODELS OF DEPLOYMENT OF CLOUD SERVICES [4]

Cloud Model	Description
Private	In this model, the cloud infrastructure is exclusively used by a specific organization. The cloud may be local or remote, and managed by the company itself or by a third party. There are policies for accessing cloud services. The techniques employed to enforce such private model may be implemented by means of network management, service provider configuration, authorization and authentication technologies or a combination of these.
Public	Infrastructure is made available to the public at large and can be accessed by any user that knows the service location. In this model, no access restrictions can be applied and no authorization and authentication techniques can be used.
Community	Several organizations may share the cloud services. These services are supported by a specific community with similar interests such as mission, security requirements and policies, or considerations about flexibility. A cloud environment operating according to this model may exist locally or remotely and is normally managed by a commission that represents the community or by a third party.
Hybrid	Involves the composition of two or more clouds. These can be private, community or public clouds which are linked by a proprietary or standard technology that provides portability of data and applications among the composing clouds.

 ISBN: 978-1-61208-153-3

Private Cloud computing presents a few challenges related to protection, trust, privacy and security of user data.

III. CLOUD RELATED WORK ON SECURITY AND TRUST

This section review some related work about security, file system and trust in the cloud.

A. Security in the Cloud

A number of technologies have been employed in order to provide security for cloud computing environments. The creation and protection of security certificates is usually not enough to ensure the necessary security levels in the cloud. Cryptographic algorithms used with cloud applications usually reduce performance and such reduction must be restricted to acceptable levels [21].

Cloud computing offers users a convenient way of sharing a large quantity of distributed resources belonging to different organizations. On the other hand, the very nature of the cloud computing paradigm makes security aspects quite more complex. Trust is the main concern of consumers and service providers in a cloud computing environment [7]. The inclusion of totally different local systems and users of quite diverse environments brings special challenges to the security of cloud computing. On one hand, security mechanisms must offer users a high enough level of guarantees. On the other hand, such mechanism must not be so complex as to make it difficult for users to use the system. The openness and computational flexibility of popular commercially available operating systems have been important factors to support the general adoption of cloud computing. Nevertheless, these same factors increase system complexity, reduce the degree of trust and introduce holes that become threats to security [7].

Huan et al. [22] investigate the different security vulnerability assessment methods for cloud environments. Experiments show that more vulnerabilities are detected if vulnerable tools and servers are in the same LAN. In other word, the hackers can find an easier way to get the target information if it is on the same LAN of compromised systems. Experimental results can be used to analyze the risk in third party compute clouds.

Popovic et al. [23] discuss security issues, requirements and challenges that Cloud Service Providers (CSP) face during cloud engineering. Recommended security standards and management models to address these are suggested both for the technical and business community.

B. Filesystem Security

As the number of devices managed by users is continually increasing, there is a growing necessity of synchronizing several hierarchically distributed file systems using ad-hoc connectivity. Uppoor et al. [6] present a new approach for synchronizing of hierarchically distributed file systems. Their approach resembles the advantages of peer-to-peer synchronization, storing online master replicas of the shared files. The proposed scheme provides data synchronization in a peer-to-peer network, eliminating the costs and bandwidth requirements usually present in cloud computing master-replica approaches.

The work in [9] presents CDRM, a scheme for dynamic distribution of file replicas in a cloud storage cluster. This scheme periodically updates the number and location of file block replicas in the cluster. The number of replicas is updated according to the actual availability of cluster nodes and the expected file availability. The dynamic distribution algorithm for replica placement takes into account the storage and computational capacity of the cluster nodes, as well as the bandwidth of the communication network. An implementation of the proposed scheme using an open source distributed file system named HDFS (Hadoop Distributed File System) is discussed. Experimental measurements point out that the dynamic scheme outperforms existing static file distribution algorithms.

C. Trust

The concepts of trust, trust models and trust management have been the object of several recent research projects. Trust is recognized as an important aspect for decision-making in distributed and auto-organized applications [19] [20]. In spite of that, there is no consensus in the literature on the definition of trust and what trust management encompasses. In the computer science literature, Marsh is among the first to study computational trust. Marsh [19] provided a clarification of trust concepts, presented an implementable formalism for trust, and applied a trust model to a distributed artificial intelligence (DAI) system in order to enable agents to make trust-based decisions. Marsh divided trust into three categories: **1. Basic Trust** – This is the level of trust which represents the general trust disposition of agent X ∈ 2 A at time t. 2. **General Trust** – Given agents x, y ∈ A, the general trust $Tx(y)^t$ represents the amount of trust that x has in y at time t. 3. **Situational Trust** – Given agents x, y ∈ A, and a situation α, the situational trust $Tx(y,\alpha)^t$ represents the amount of trust that x has in y in situation α at time t.

Beth et al. [20] also proposed a trust model for distributed networks. They derived trust recommendations from direct trust and gave them formal representations, as well as rules to derive trust relationships and algorithms to compute trust values. Josang et al. [24] describe a trust model where positive and negative feedback about a specific member is accumulated. The model is based on the Bayesian network model, using the beta probability density function to calculate a member's expected future behavior.

Trust is considered to be more than the authorized nature of security relations between human societies, which achieve stable and healthy operation, to a large extent thanks to the trust relationship between the individuals, groups and organizations. Therefore, in a large number of dynamic user-oriented open network environments, the study of the trust relationships between the trust-based security mechanisms to ensure the safe operation of distributed applications has become a fundamental topic. Currently, most scholars have

 ISBN: 978-1-61208-153-3

reached a consensus that trust should have three important features [25], which are discussed bellow.

1) Subjectivity (different entities of the same view of things which will be affected by factors such as individual preferences may vary);

2) The expected probability (the degree of trust can be extracted and formalized as the estimated likelihood of a given event);

3) Relevance (trust is an aspect of things, for specific content).

In recent works on trust, mainly two distinct methods are used for subjective trust reasoning: probabilistic reasoning based on statistical hypothesis testing; and approaches based on fuzzy theory, expert systems and artificial intelligence techniques. However, these methods do not fully reflect the essential nature of trust. Subjective trust, in essence, is based on the belief that it has great uncertainty. In the subjective, objective world, random and fuzzy uncertainties are the two main forms that have become the industry consensus [26]. Thus, the axiomatic methods based on probability theory or fuzzy set theory do not achieve a comprehensive assessment of trust information.

D. Trust in the Cloud

Trust and security have become crucial to guarantee the healthy development of cloud platforms, providing solutions for concerns such as the lack of privacy and protection, the guarantee of security and author rights.

Privacy and security have been shown to be two important obstacles concerning the general adoption of the cloud computing paradigm. In order to solve these problems in the IaaS service layer, a model of trustworthy cloud computing which provides a closed execution environment for the confidential execution of virtual machines was proposed [5]. This work has shown how the problem can be solved using a Trusted Platform Module. The proposed model, called Trusted Cloud Computing Platform (TCCP), is supposed to provide higher levels of reliability, availability and security. In this solution, there is a cluster node that acts as a Trusted Coordinator (TC). Other nodes in the cluster must register with the TC in order to certify and authenticate its key and measurement list. The TC keeps a list of trusted nodes. When a virtual machine is started or a migration takes place, the TC verifies whether the node is trustworthy so that the user of the virtual machine may be sure that the platform remains trustworthy. A key and a signature are used for identifying the node. In the TCCP model, the private certification authority is involved in each transaction together with the TC [5].

Shen et al. [7] presented a method for building a trustworthy cloud computing environment by integrating a Trusted Computing Platform (TCP) to the cloud computing system. The TCP is used to provide authentication, confidentiality and integrity [7]. This scheme displayed positive results for authentication, rule-based access and data protection in the cloud computing environment.

Cloud service providers (CSP) should guarantee the services they offer, without violating users' privacy and confidentiality rights. Li et al. [8] introduced a multi-tenancy trusted computing environment model (MTCEM). This model was designed for the IaaS layer with the goal of ensuring a trustworthy cloud computing environment to users. MTCEM has two hierarchical levels in the transitive trust model that supports separation of concerns between functionality and security. It has 3 identity flows: a) the consumers, who hire the CSP cloud computing services; b) the CSP, that provides the IaaS services; c) the auditor (optional, but recommended), who is responsible for verifying whether the infrastructure provided by the CSP is trustworthy on behalf of users. In MTCEM, the CSP and the users collaborate with each other to build and maintain a trustworthy cloud computing environment.

Zhimin et al. [12] propose a collaborative trust model for firewalls in cloud computing. The model has three advantages: a) it uses different security policies for different domains; b) it considers the transaction contexts, historic data of entities and their influence in the dynamic measurement of the trust value; and c) the trust model is compatible with the firewall and does not break its local control policies. A model of domain trust is employed. Trust is measured by a trust value that depends on the entity's context and historical behavior, and is not fixed. The cloud is divided in a number of autonomous domains and the trust relations among the nodes is divided in intra and inter-domain trust relations. The intra-domain trust relations are based on transactions operated inside the domain. Each node keeps two tables: a direct trust table and a recommendation list. If a node needs to calculate the trust value of another node, it first checks the direct trust table and uses that value if the value corresponding to the desired node is already available. Otherwise, if this value is not locally available, the requesting node checks the recommendation list in order to determine a node that has a direct trust table that includes the desired node. Then it checks the direct trust table of the recommended node for the trust value of the desired node. The process continues until a trust value for the desired node is found in a direct trust table of some node. The inter-domain trust values are calculated based on the transactions among the inter-domain nodes. The inter-domain trust value is a global value of the nodes direct trust values and the recommended trust value from other domains. Two tables are maintained in the Trust Agents deployed in each domain: form of Inter-domain trust relationships and the weight value table of this domain node.

In [17] a trusted cloud computing platform (TCCP) which enables IaaS providers to offer a closed box execution environment that guarantees confidential execution of guest virtual machines (VMs) is proposed. This system allows a customer to verify whether its computation will run securely, before requesting the service to launch a VM. TCCP assumes that there is a trusted coordinator hosted in a trustworthy external entity. The TCCP guarantees the

 ISBN: 978-1-61208-153-3

confidentiality and the integrity of a user's VM, and allows a user to determine up front whether or not the IaaS enforces these properties.

The work [18] evaluates a number of trust models for distributed cloud systems and P2P networks. It also proposes a trustworthy cloud architecture (including trust delegation and reputation systems for cloud resource sites and datacenters) with guaranteed resources including datasets for on-demand services.

IV. HIGH LEVEL TRUST MODEL FOR FILE SHARING

According to the review and related research [5] [6] [7] [8] [10] [12] [17], it is necessary to employ a cloud computing trust model to ensure the exchange of files among cloud users in a trustworthy manner. In this section, we introduce a trust model to establish a ranking of trustworthy nodes and enable the secure sharing of files among peers in a public cloud.

We propose a trust model where the selection and trust value evaluation that determines whether a node is trustworthy can be performed based on node storage space, link and processing capacity. For example, if a given client has access to a storage space in a public cloud, it still has no selection criterion to determine to which cloud node it will send a particular file.

When a node wants to share files with other users, it will select trusted nodes to store this file through the following metrics: processing capacity (the average workload processed by the node, for example, if the node's processing capacity is 100% utilized, it will take longer to attend any demands), storage capacity and link (better communication links and storage resources imply greater trust values, since they increase the node's capacity of transmitting and receiving information). The trust value is established based on queries sent to nodes in the cloud, considering the metrics previously described.

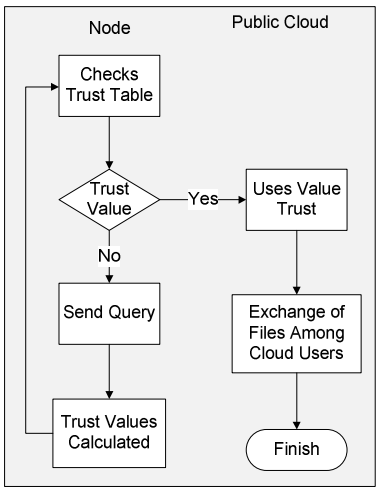


Figure 3. Proposed Trust Model.

Each node maintains two trust tables: direct trust table and the recommended list. a) If a node needs to calculate the trust value of another node, it first checks the direct trust table and uses the trust value if the value for the node exists. If this value is not available yet, then the recommended lists are checked to find a node that has a direct trust relationship with the desired node the direct trust value from this node's direct trust table is used. If there's no value attached, then it sends a query to its peers requesting information on their storage space, processing capacity and link. The trust values are calculated based on queries exchanged between nodes.

b) The requesting node will assign a greater trust value to nodes having greater storage capacity and / or processing and better link.

The trust value of a node indicates its suitability for storage and cloud operations. This value is calculated based on the historical interactions of the node, being represented by T_{np}, for a given node. Its value may range from 0 to 1. As we have previously stated, the value of T_{np} is calculated from queries exchanged between nodes regarding their overall system capacities. Figure 3 presents a high level view the proposed trust model, where the nodes query their peers to obtain the information needed to build their local trust table.

In this model, a trust rank is established, allowing a node A to determine whether it is possible to trust a node B to perform storage operations in a public cloud. In order to determine the trust value of B, node A first has to obtain basic information on this node. Figure 4 depicts the query exchange process used for gathering the necessary trust information from a node B by a node A.

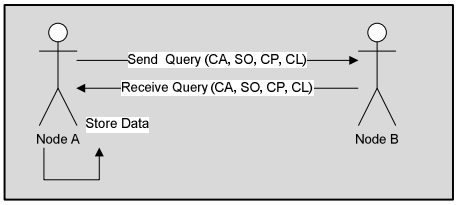


Figure 4. Scenario of Information Request

Node A needs to exchange a file in the cloud and wants know if the node B can be trusted to store and send the file. The protocol Trust Model can be described as follows: In step 1, A sends a query to B regarding its storage capacity, operating system, processing capacity and link. In step 2, B sends a response to he query sent by A, providing the requested information. In step 3, node A evaluates the information received from B and, if the information is consistent, it is stored in A's local trust table. In general, the trust of node A in node B, in the context of a public cloud NP, can be represented by:

$$T_{a,b}^{np} = V_{np}^{b} \quad (1)$$

Where V_{np}^{b} is the trust value of B in the public cloud NP analyzed by A and $T_{a,b}^{np}$ represents the trust of A in B, in the public cloud NP. According to the definition of trust, V_{np}^{b} equals the queries sent and received (interaction) by A and B in the cloud NP.

 ISBN: 978-1-61208-153-3

The trust information may be stored as individual records of interaction with the respective node, being recorded in a local database that contains information about the behavior of each node in the cloud. Thus, the trust of node A in node B in the cloud NP can be represented by:

$$T_{a,b}^{fnp} = \frac{\sum_{i=1}^{j} V_{npi}^{b}}{j}, \text{ for } j > 0 \qquad (2)$$

$T_{a,b}^{fnp}$ represents the final trust of A in B in the cloud NP, while j represents the number of interactions / querys between nodes A and B in the cloud NP.

A. Trust Calculation

Three aspects can have an impact on calculating the direct trust of a node, as shown in the table III. A larger storage space and processing capacity have greater weight in the choice of more reliable nodes, because these characteristics ensure the integrity and storage of files. Thus, to calculate the direct trust of the node, storage space and processing capacity is assigned with weights of 40% and the links with the remaining 20%.

Knowing that any node can have its trust value ranging from 0 to 1, and knowing that these values vary in time, it means that one node can have its storage capacity increasing or decreasing, becoming necessary the behavior reflection of the Direct Trust be in time. This way, nodes with characteristics more constant are more reliable because they have less sway in its basic features.

TABLE III. ISSUES AFFECTING THE DIRECT TRUST OF A NODE

STORAGE SPACE	PROCESSING CAPACITY	LINK CAPACITY	DIRECT TRUST
HIGH	HIGH	HIGH	HIGH
HIGH	HIGH	LOW	HIGH
HIGH	LOW	HIGH	MEDIUM (ACCORDING TO THE VALUES OF STORAGE AND PROCESSING)
HIGH	LOW	LOW	LOW
LOW	HIGH	HIGH	MEDIUM (ACCORDING TO THE VALUES OF STORAGE AND PROCESSING)
LOW	HIGH	LOW	LOW
LOW	LOW	HIGH	LOW
LOW	LOW	HIGH	LOW
LOW	LOW	LOW	LOW

Below is shown the simulation of direct trust and trust index. To perform the simulation, it was used the method of Monte Carlo [28] to generate random or pseudo-random numbers for storage, processing and link, due to the nodes attributes of a cloud not having variations no-correspondent to deterministic behavior but the stochastic behaviors. The values of each attribute are numbers ranging from 0 to 1 corresponding the percentages to each node. For the trust index, the conditions were established as in the following table.

TABLE IV. CONDITIONS OF THE TRUST INDEX

Index	Situation
$I_{a,b}^{fnp} => 0,1$	Do not trust on the node
$I_{a,b}^{fnp} < 0,1$	Trust on the node

TABLE V. REFERENCE VALUES FOR CONSENSUS IN TRUST

Value	Description	Decision
0	No Trust in the node Cloud Public	No opinion
[0, 0.39]	Low Trust in the node Cloud Public	Not trust
[0.4, 0.59]	Medium Trust in the node Cloud Public	Not trust
[0.6, 0.89]	High Trust in the node Cloud Public	Trust
[0.9, 0.99]	Very High Trust in the node Cloud Public	Trust

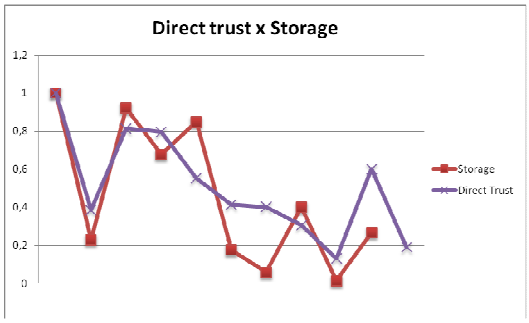


Figure 5. Direct Trust and Storage

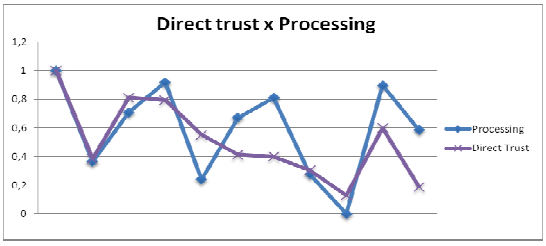


Figure 6. Direct Trust and Processing

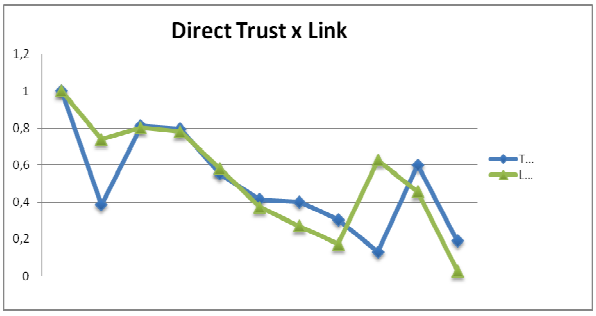


Figure 7. Direct Trust and Link

With simulation you can see how values influence the trust index.

 ISBN: 978-1-61208-153-3

V. CONCLUSION

We have presented an overview of the cloud computing paradigm, as well as its main features, architectures and deployment models. Moreover, we identified the main issues related to trust and security in cloud computing environments.

In order to address these issues, we proposed a trust model to ensure reliable exchange of files among cloud users in public clouds. In our model, the trust value of a given node is obtained from a pool of simple parameters related to its suitability for performing storage operations. Nodes with greater trust values are subsequently chosen for further file storage operations.

As a future work, we plan to implement the proposed trust model and analyze node behavior after the ranking of trustworthy nodes is established.

REFERENCES

[1] Chen Kang and Zen WeiMing, "Cloud computing: system instance and current research," Journal of Software, pp. 20(5):1337-1347. 2009.

[2] Minqi Zhou, Rong Zhang, Dadan Zeng, and Weining Qian, "Services in the cloud computing era: a survey," Software Engineering Institute. Universal Communication. Symposium (IUCS), 4th International. IEEE Shanghai, pp. 40-46. China. 978-1-4244-7821-7 (2010).

[3] A. Marinos and G. Briscoe, "Community cloud computing," in First International Conference Cloud Computing, CloudCom, volume 5931 of Lecture Notes in Computer Science, pp. 472–484. Springer (2009).

[4] P. Mell and T. Grance, The NIST Definition of Cloud Computing (Draft). National Institute of Standards and Technology. http://csrc.nist.gov/groups/SNS/cloud-computing. 2009. 30 may 2011.

[5] Wang Han-zhang and Huang Liu-sheng, "An improved trusted cloud computing platform model based on DAA and Privacy CA scheme," IEEE International Conference on Computer Application and System Modeling (ICCASM 2010). 978-1-4244-7235-2. 2010.

[6] S. Uppoor, M. Flouris, and A. Bilas, "Cloud-based synchronization of distributed file system hierarchies," Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), IEEE International Conference, pp. 1-4. 2010.

[7] Zhidong Shen, Li Li, Fei Yan, and Xiaoping Wu, "Cloud Computing System Based on Trusted Computing Platform," Intelligent Computation Technology and Automation (ICICTA), IEEE International Conference on Volume: 1, pp. 942-945. China. 2010.

[8] Xiao-Yong Li, Li-Tao Zhou, Yong Shi, and Yu Guo, "A Trusted Computing Environment Model in Cloud Architecture," Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, 978-1-4244-6526-2. Qingdao, pp. 11-14. China. July 2010.

[9] Qingsong Wei, Bharadwaj Veeravalli, Bozhao Gong, Lingfang Zeng, and Dan Feng, "CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster," 2009 IEEE International Conference on Cluster Computing (CLUSTER), pp. 188-196, 2010.

[10] Kai Hwang, Sameer Kulkareni, and Yue Hu, "Cloud Security with Virtualized Defense and Reputation-Based Trust Mangement," 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC '09), pp. 717-722, 2009.

[11] Xue Jing and Zhang Jian-jun, "A Brief Survey on the Security Model of Cloud Computing," 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Hong Kong IEEE, pp. 475 – 478. Aug 2010.

[12] Zhimin Yang, Lixiang Qiao, Chang Liu, Chi Yang, and Guangming Wan, "A collaborative trust model of firewall-through based on Cloud Computing," Proceedings of the 2010 14th International Conference on Computer Supported Cooperative Work in Design. Shanghai, China. pp. 329-334, 14-16. 2010.

[13] M. Miller, Cloud Computing – Web-Based Applications That Change the Way You Work and Collaborate Online, Que Publishing, Pearson Education, Canada 2008.

[14] P. Mell and T. Grance, "Draft nist working definition of cloud computing - v15," 21. Aug 2009.

[15] T. Dillon, Chen Wu, and E. Chang, "Cloud Computing: Issues and Challenges," 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 27-33. Australia, 2010.

[16] Li Xiaoqi, Lyu M R, and Liu Jiangchuan. "A trust model based routing protocol for secure AD Hoc network," Proceedings of the 2004 IEEE Aerospace Conference, pp. 1286-1295. 2004.

[17] N. Santos, K. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," Proc. HotCloud. June 2009.

[18] Kai Hwang, Sameer Kulkareni, and Yue Hu, "Cloud Security with Virtualized Defense and Reputation-Based Trust Mangement," 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), Chengdu, pp.717-722. China 2009.

[19] S. P. Marsh, "Formalising Trust as a Computational Concept", Ph.D. Thesis, University of Stirling, 1994.

[20] T. Beth, M. Borcherding, and B. Klein, "Valuation of trust in open networks," In ESORICS 94. Brighton, UK, November 1994.

[21] H. Takabi, J. B. D. Joshi, and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," IEEE Security and Privacy, vol. 8, no. 6, pp. 24-31, Nov./Dec. 2010, doi:10.1109/MSP.2010.186.

[22] Huan-Chung Li, Po-Huei Liang, Jiann-Min Yang, and Shiang-Jiun Chen, "Analysis on Cloud-Based Security Vulnerability Assessment," 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE), pp. 490-494, 2010.

[23] K. Popovic and Z. Hocenski, "Cloud computing security issues and challenges," MIPRO, 2010 Proceedings of the 33rd International Convention, pp. 344-349, 24-28 May 2010 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5533317&isnumber=5533310.

[24] A. Jøsang and R. Ismail, "The Beta Reputation System," In Proceedings of the 15th Bled Electronic Commerce Conference, pp. 17-19. June 2002.

[25] A. Abdul-Rahman and S. Hailes, "A distributed trust model," In Proceedings of the 1997 New Security Paradigms Workshop, pp. 48-60, 1998.

[26] A. Jφsang and S. J. Knapskog, "A metric for trusted systems," Global IT Security, pp. 541-549, 1998.

[27] Zhao-xiong Zhou, He Xu, and Suo-ping Wang, "A Novel Weighted Trust Model based on Cloud," AISS: Advances in Information Science and Service Sciences, Vol. 3, No. 3, pp. 115- 124, April 2011.

 ISBN: 978-1-61208-153-3

Security Management of a Cloud-based U-City Management System

Sung Min Kim, Jun Oh Kim, Chang Ho Yun,
Jong Won Park, Yong Woo LEE (Corresponding Author)
School of Electrical & Computer Engineering
University of Seoul, Ubiquitous-City (Smart City) Consortium
Seoul, Korea
{smkim, kjo, touch011, comics77, ywlee}@uos.ac.kr

Hae Sun Jung
Ubiquitous-City (Smart City) Consortium
Seoul, Korea
holylife7@hotmail.com

***Abstract*— In this paper, we introduce a user authentication methodology for a cloud-based U-City management system to manage the U-City which includes ubiquitous resources and cloud computing resources. Cloud computing enables the integrated urban operation center of the U-City to provide limitless computing power without having its own computing center. However, because huge number of services and users use the U-City service and the cloud computing power and they should be carefully screened, we need a specially designed security management to protect the U-City and its facilities. For it, we propose the cloud-based U-City management system, UTOPIA which uses SAML-based Single-Sign-On (SSO) authentication for the security management to do user authentication and privilege management for the cloud computing in the U-City.**

Keywords-U-City; Cloud Computing; SAML; U-City Security Management; Single-Sign On.

I. INTRODUCTION

With ubiquitous computing, we are seeking the way to satisfy human beings' desire to enjoy IT services with any device, anytime and anywhere. In U-City, every possible information system such as residential, environmental, medical, business, governmental, social and the like is linked through ubiquitous computing technologies and the whole U-City acts as virtually one system or a global system which works for human beings and takes care of them.

U-City is usually centrally managed by the central operation center which processes the huge amount of the data and often needs huge computing power. The irregularity of need in computing power makes it very attractive that the U-City uses cloud computing since the cloud computing obviously save the cost of computing power [1].

Cloud computing is defined as "a style of computing where scalable and elastic IT-related capabilities are provided as a service to customers using Internet technologies." [2]. To use cloud service requires generally better security than to use private system. In order to include cloud computing in U-City, we should keep it in mind. As well-publicized cases of cloud computing vulnerability, we can think Amazon S3 malfunction over seven-hour on July 20, 2008 [3], Gmail outage over one-day in mid-October 2008 [4] and Google Docs vulnerabilities [5].

Security issues such as user authentication, information protection and access control should be carefully solved in order that we provide U-City services using the cloud computing to users. For it, we propose a U-City Management System named UTOPIA, which uses SSO authentication technology based SAML as a part of the security management [6]. Users who use the U-City Management System do not need to be bothered to login again and again whenever they use difference service of UTOPIA but login just once.

This paper is organized as follows. Section 2 briefly introduces related works and confirms that this work is the first and only work till now. Section 3 outlines our U-City management system, UTOPIA, which is composed of the three tiers. Sections 4 explains our SAML based SSO user authentication as a part of our security management. Section 5 describes the implementation of the SAML based SSO user authentication into UTOPIA. Finally, Section 6 gives conclusions and explains future works.

II. RELATED WORK

A. U-City Management

U-City management system enables citizens to easily use U-City services. There are many U-Cities in Korea as shown in Table 1.

Seoul Metropolitan Government Information Agency, Seoul in Korea has been building many U-Towns based on U-Seoul Master Plan which aims U-Care, U-Fun, U-Green, U-transport, U-Business and U-Government [7]. They have central operation centers, which use their own U-City management systems but they are different from UTOPIA which uses the three tier U-City Management System Paradigm based on U-City middleware and U-City portal. They have various kinds of user services, but they are not integrated together and do not provide SSO based user authentication.

U-Dongtan U-City provides many kinds of public services such as surveillance of public areas, environment pollution information, water leakage management, media board, traffic information and so on. It uses user authentication with just identity and password and does not use SSO authentication [8]. The U-city has central operation center which use a platform but does not use U-City middleware. Like these, we are the only U-City management system which uses three tier U-City paradigm, U-City middleware and the U-City

 ISBN: 978-1-61208-153-3

portal and uses SSO authentication. That is, currently, there is no U-City which is based on our concept.

TABLE I. SOME MAJOR U-CITY PROJECTS IN KOREA

U-city Project Name	*Period*	*Goal*
Digital Media City	2001~2010	The world best IT town. Northeast Asian IT hub.
U-Gangnam	2004~2007	Seamless Connectivity. Mobile Environment. Teleportation & Telework.
U-cheonggyecheon	2007	3D-based GIS. A U-City testbed.
U-Myengdong/ U-Ljiro	2007~2010	Digital media plaza. Digital media stree, Digital media gallery.
Eunpyeong New Town	2006~2011	A ubiquitous new-town.
U-Busan	2004~2010	The World first u-city. U-port, U-Traffic, U-convention.
U-Gwangju	2004~2012	Centered on U-home.
U-Daejeon	2004~2007	U-cluster. U-wellbeing.
U-Gyeongbuk	2004~2010	The largest U-City testbed. U-culture.
U-Pyong Chang	2006~2010	U-city for winter sports.
U-Chungbuk	2005~2009	3D GIS. U-cluster.
U-Jeju	2004~2006	Focused on telematics.
U-Sejong	2005~2030	U-government
Gwanggyo New-town	2005~2011	A ubiquitous well being town.
Pangyo New-town	2006~2010	A U-echo city.
U-Dongtan	2003~2007	GIS. ITS. BcN. IBS.
U-Jeonju	2005~2008	U-culture. U-tour. U-traffic.
U-Paju	2005~2009	Total Life-Card. Smart transport.
U-Bucheon	2010~2014	U-home network, U-traffic, U-tour. U-echo. U-safety.
U-Changwon	2004~2008	Digital broadcasting. Media center.
U-Ansan	2007~2012	U-Industry. U-tour.

B. Security management using personal authentication technology

Table 2 summarizes major personal authentication methods case by case. Currently SSO is popular and widely used [9]. We think SSO is one of the best solutions to the security management in U-City, since so many kinds of U-City services are provided, so many kinds of organizations such as public agencies, financial institutions, large corporations, educational institutions are integrated in U-City and the separately developed U-Cities can be merged into a larger U-City later. Currently no U-City uses SSO for their security management, and this work is the first case and the only research at the moment.

TABLE II. THE PERSONAL AUTHENTICATION

Security Technology	*Description*
ID/Password	It is a typical personal authentication method. It requires periodic renewal [10].
Public key certificate	It uses a digital signature to bind a public key with an identity. The private keys are stored in certificate storage location. Encryption / decryption processing, cryptography transmission method are usually used to protect them. By implementing programs to protect private key and certificate into client computers, the security can be improved. However, it requires users' agreement and actions and causes extra maintenance expenses. [11]
SSO (Single Sign On)	Users log in once and gain access to all systems without being prompted to log in again at each system [12].
MTM (Mobile Trusted Module)	It is a hardware-based authentication which was proposed by TCG (Trusted Computing Group). It is usually used for the authentication of mobile devices and is recently used for cloud computing authentication with SIM (Subscriber Identity Module) [13].
Finger Print and Identifier	It uses user's bio profiles such as finger print and etc. which are usually kept in the file system and are used to identify the user. But, it is weak to Trojan horse attacks, memory hacking and key-logging because users' profiles are store in the file system [14].
IP-Geographic location Identification	It uses user's IP location and is helpful to prevent MITM attacks [14].
Knowledge-based authentication	It asks the question about specific knowledge of user information. But, it is usually used with other methods because it is vulnerable to MITM attacks [14].
OTP(One-Time Password)	It uses a password which is valid during only one login session to avoid a shortcoming of static passwords. In order to deliver the OTP, text messaging or proprietary tokens or web-based method is used. It is vulnerable to key logging and MITB because it relies on a key input [14].
Out-of-Band authentication	Each time, it uses a different communication channel to verify a transaction request. It guarantees very high security but it requires initial registration. Thus it can be expensive [14] [15].
Internet Personal Identification Number (i-PIN)	It uses the Resident Registration Number for login. It is used in South Korea [16] [17].

III. CLOUD-BASED U-CITY MANAGEMENT

Our U-City management system, UTOPIA, supports the unified ubiquitous cloud environment by providing dynamic service deployment based on context-awareness, high performance and collaborative computing on Grid and cloud. It is based on three tiers paradigm as shown in Figure 1. The feeling tier is composed of U-City infrastructure such as buildings, bridges, loads, etc., and ubiquitous IT devices including sensors and video cameras and connected to the processing tier through Broadband Convergence Network (BcN) and Ubiquitous Sensor Network (USN). The processing tier plays a role of brain in human body, receives data from the feeling tier and processes them intelligently. Finally, the presentation tier receives the request of users and

 ISBN: 978-1-61208-153-3

sends it to processing tier and show the result from the processing tier to them. It acts as a window to the U-City system.

Our middleware in the processing tier, which we call SOUL, supports cloud computing and security management facilities including user authentication. It also supports Computational Grid so that it can smoothly satisfy applications which require real-time high performance computing. It supports computer supported cooperative work (CSCW) through Access Grid that is the best choice currently and a next generation CSCW.

It has the following additional characteristics. First, it provides common device interface which was designed to support variable sensor network data-sinks and protocols. Therefore, it can be used as the common gateway for various kinds of sensors and ubiquitous sensor networks which collect sensed data.

Second, it uses ontology-based intelligent inference engine which provides context-aware, that is, intelligent information using the sensed data through the common device interface. Third, it provides a user-transparent infrastructure that generates and provides intelligent services, which are invisible to users, to various applications. Fourth, it enables user to control remote devices in real-time mode so that remote control devices such as fire doors and other emergency devices can be controlled remotely in real-time mode. Fifth, it has the advantages of layered architecture since it is designed to have layer architecture. Lastly and sixth, it can directly be connected to easy-to-use, yet convenient, user interfaces, the U-City portal.

We believe that these advantages make SOUL possible to be used in various kind of U-city applications of our project and it can shorten the period and expense to develop the U-city applications. [18]

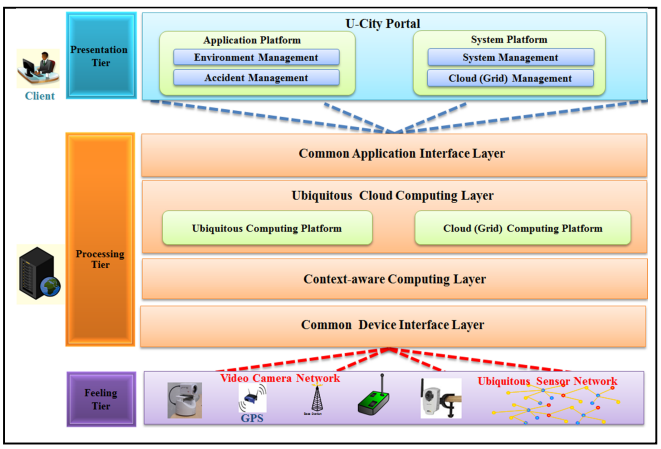


Figure 1. UTOPIA: A U-City Management System.

The U-City portal has the security management component and beyond it, it has two distinctive components as shown in figure 2. One is the application platform and the other is the system platform. The environment management application is one of the application services which belongs to the application platform and give services in the management of noise, air-pollution and water quality using GIS visualization technology. The cloud management belongs to the system platform [19][20].

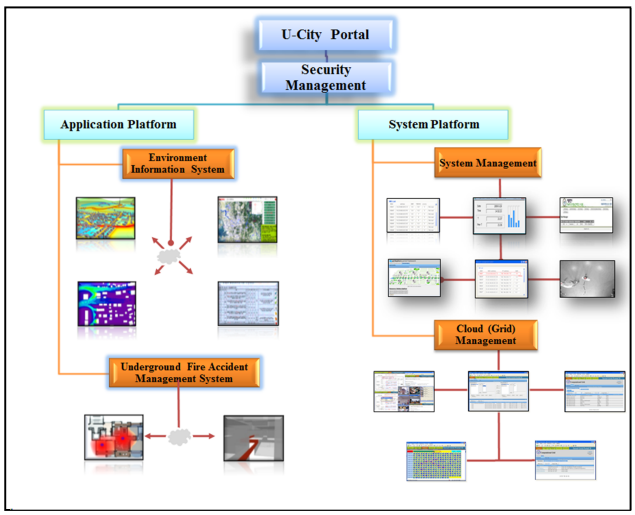


Figure 1. The architecture of the U-City portal.

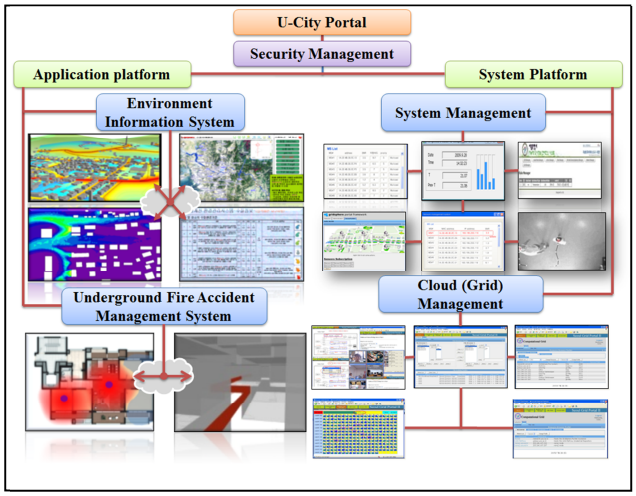


Figure 2. The architecture of the U-City portal.

IV. SSO AND SAML IN UTOPIA

The advantage of the adopting SSO in the U-City management system can be explained in following two viewpoints.

 ISBN: 978-1-61208-153-3

1) Users' viewpoint:

Without SSO, users are asked to log in each service or each organization when they want to use the U-City service and therefore users should remember their IDs and passwords in each login. With SSO, users just should log in one time. It is more convenient, easier and more efficient than without SSO.

2) Administrators' viewpoint:

With SSO, administrators can trace users activities and manage users security at the level of overall or total management of U-City, not at the level of individual organization or services. That is, U-City can have total solution for security management with easy administration and more consistant manner with SSO.

SSO in UTOPIA is operated as shown in Figure 3 and Figure 4. When a user accesses UTOPIA through the U-City portal to use the U-City services, the login facility starts. The SSO agent in the login facility for the security management in UTOPIA sends the user's information to the authentication manager using the SAML request. SAML messages at each step are encrypted using the SSL protocol. Then, the authentication manager does the security screening by asking the credential database and makes a decision. Since UTOPIA uses SSO, the user does not have to be bothered by login many times if he/she wants to use several services in UTOPIA.

.

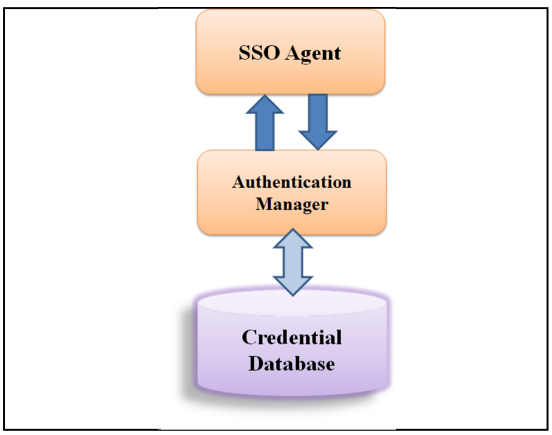


Figure 3. SSO management in UTOPIA.

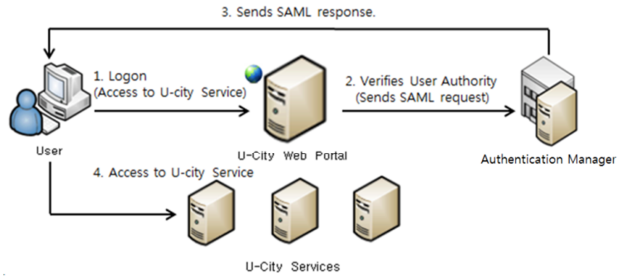


Figure 4. Single-Sign-On Service in UTOPIA.

V. IMPLEMENTATION

Figure 5 shows how UTOPIA processes SAML-based SSO for U-City. SAML supports implementing SSO. We use OpenSAML2 Library and openssl. The RSA key pair is generated with openssl for authentication. The RSA key pairs are UTOPIA_sso_private.der and UTOPIA_sso_public.der. SSO is implemented with original RSA public key. The user accesses the service provider, the U-City portal of UTOPIA, to use the U-City service. ServiceProviderForm in U-City portal is the access web page. The elements of ServiceProviderForm are loginForm, providerName, RelayState, acsURI. LoginForm is for authentication in identity provider. ProviderName is the name of service provider providing the service. RelayState is the redirected service page after ACS authentication. AcsURI is the URL to verify SAMLResponse in identity provider. The U-City service provider generates SAMLRequest in a XML format. SAMLRequest is sent to authentication provider, that is, Identity Provider, through the user's browser. The authentication provider parses the SAMLRequest and process user authentication. The authentication provider generates a SAMLResponse. The authentication provider sends SAMLResponse to the Assertion Consumer Service (ACS) through the user browser. The ACS in the service provider receives the SAMLResponse sent by the authentication provider and validates it. If it is o.k., then the service provider gives the user a permit to log in UTOPIA. Now, the user can successfully log in to UTOPIA and use the wanted U-City service.

 ISBN: 978-1-61208-153-3

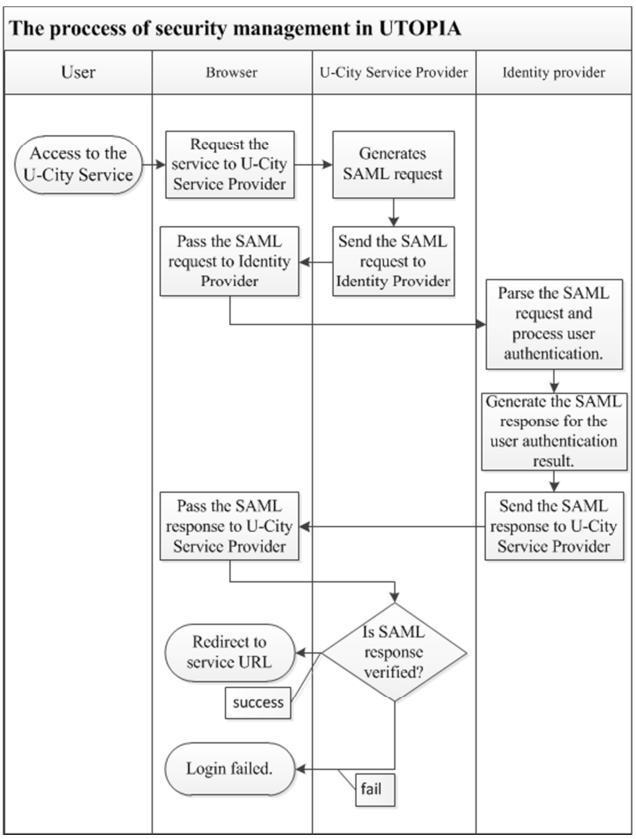


Figure 5. The sequence diagram of security management in UTOPIA.

VI. CONCLUSION

This paper explained the security management of U-City management system which uses cloud computing heavily. Our U-City management system, UTOPIA, adopted SAML-based SSO as a user authentication methodology for our security management facility. It is the first and the only U-City management system to use SSO currently. Users can use the services of UTOPIA through the U-City portal with unified authentication which uses one-time login for all UTOPIA services. The work is still in progress since we have been continuously adding many organizations and more services and in future work, we will continue to support a more fine-grained privilege management.

ACKNOWLEDGMENT

This study was supported by the Seoul Research and Business Development Program (10561), Smart (Ubiquitous) City Consortium, Seoul Grid Center. We would like to give thanks to Mr. Cheol Sang Yoon, Mr. Tae Ho Hong, Mr. Eui Dong Hwang, Mr. Kyoung-gyu Lee and the staffs of Seoul Grid Center and the members of Smart (Ubiqitous) City Consortium for their contribution to this research.

REFERENCES

[1] J. W. Park, C. H. Yun, S. Kim, H.Y. Yeom and Y. W. Lee, “Cloud Computing Platform for GIS Image Processing in U-City,” 13th International Conference on Advanced Communication Technology (ICACT), pp. 1151-1155, 2011.

[2] Gartner’s Cloud Computing website [online], May 2011, Available from: http://www.gartner.com/technology/research/cloud-computing/index.jsp.

[3] Amazon S3 Availability Event: July 20, 2008 [online], May 2011, Available from: http://status.aws.amazon.com/s3-20080720.html.

[4] Extended Gmail outage hits Apps admins [online], May 2011, Available from: http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9117322.

[5] Google’s response to Google Docs concerns. [online], May 2011, Available from: http://googledocs.blogspot.com/2009/03/just-to-clarify.html.

[6] T. Gross, “Security Analysis of the SAML Single Sign-on Browser/Artifact Profile,” Annual Computer Security Applications Conference, vol.19, pp. 298-307, 2003.

[7] U-City of Seoul website [online], May 2011, Available from : http://info.seoul.go.kr/.

[8] U-City of Hwasong Dongtan website [online], May 2011, Available http://www.udongtan.or.kr.

[9] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, Llanos Tobarra, “Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps,” the 6th ACM workshop on Formal methods in security engineering, 2008

[10] P. Beynon-Davies, “Personal identity management as a socio-technical network,” Technology analysis & strategic management, vol.22, no.4, pp. 463-478, 2010.

[11] G. Bick, M. C. Jacobson and R. Abratt, “The Corporate Identity Management Process Revisited,” Journal Of Marketing Management, vol.19, no.7-8, pp. 835-856, 2003.

[12] Y. Y. Chan, “Weakest Link Attack on Single Sign-On and Its Case in SAML V2.0 Web SSO,” Lecture Notes in Computer Science, vol. 3982, pp. 507-516 , 2006.

[13] Trusted Computing Group website [online], May 2011, Available from: http://www.trustedcomputinggroup.org.

[14] H. S. Kim, “Cloud Computing and Personal Authentication Service”, Information & Communications Magazine, vol. 20, no. 2, pp. 11-92, 2010.

[15] A. Litan, “Where String Authentication Fails and What You Can About It,” Gartner Research, 2009.

[16] Y. Oh, T. Obi, J. S. Lee, H. Suzuki, and N. Ohyama, “Empirical analysis of internet identity misuse: case study of south Korean real name system,” he 6th ACM workshop on Digital identity management (DIM '10), pp. 27-34, 2011.

[17] S. K. Un, N. S. Jho, Y. H. Kim and D. S. Choi, “Cloud Computing Security Technology,” Electrical Communication Trend Analysis, vol. 24, no. 4, pp. 79-88, 2009.

[18] H. S. Jung, C. S. Jeong, Y. W. Lee and P. D. Hong, “An Intelligent Ubiquitous Middleware for U-city: SmartUM,” Journal of Information Science and Engineering, vol. 25, no. 2, pp.375-388, 2009.

[19] S. W. Rho and Y. W. Lee, “U-city Portal For Smart Ubiquitous Middleware," 2010 The 12th International Conference Advanced Communication Technology (ICACT), pp. 609-613, 2010.

[20] S. W. Rho, C. H. Yun and Y. W. Lee, "Provision of U-city web services using cloud computing," 13th International Conference on Advanced Communication Technology (ICACT), pp. 1545-1549, 2011.

 ISBN: 978-1-61208-153-3

Evaluating a Distributed Identity Provider Trusted Network with Delegated Authentications for Cloud Federation

Antonio Celesti, Francesco Tusa, Massimo Villari and Antonio Puliafito
Dept. of Mathematics, Faculty of Engineering, University of Messina
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {acelesti, ftusa, mvillari, apuliafito}@unime.it

***Abstract*—Federation offers an affordable opportunity for small and medium cloud providers to become as competitive as the biggest counterparts. However, in order to establish a federated cloud ecosystem, it is needed to rely on an efficient security infrastructure enabling authentication among clouds. Assuming a scalable federated cloud environment, the management of security can become very hard due to the number of authentications and trusted relationships that have to be established. Nowadays, the latest trend in authentication is the Identity Provider/Service Provider model. This paper aims to investigate a distributed IdP/SP infrastructure based on the concept of delegated authentications, evaluating its possible utilization in a federated cloud scenario.**

***Keywords*-Cloud Computing, Federation, Distributed IdPs, Trusted Network.**

I. INTRODUCTION

By now, the cloud ecosystem has been characterized by the steady rising of hundreds of independent and heterogeneous cloud providers, managed by private subjects which yield various services to their clients. Using this computing infrastructure it is possible to pursue new levels of efficiency in delivering Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to clients (e.g., companies, organizations, end-users, and so on).

Despite such an ecosystem includes hundreds of independent, heterogeneous clouds, many business operators have predicted that the process toward interoperable federated Intracloud/Intercloud environments will begin in the near future [1], even involving standardization boards (i.e., IEEE [2]). Nowadays, small/medium cloud providers are becoming popular even though their virtualization infrastructures (i.e., deployed in their datacenters) cannot directly compete with the bigger counterparts, including mega-providers such as Amazon, Google, and Saleforce. The result is that frequently small/medium cloud providers have to exploit services of mega-providers in order to develop their business logic and their cloud-based services. This means that the role of market leader is intended to remain in the hands of bigger players in the near future. To this regard, a possible future alternative scenario is based on the concept of cooperating clouds constituting the federation. Federation has always had both political and historical implications: the term refers, in fact, to a type of system characterized by an aggregation of partially "self-governing" entities with a "central government". In a federation, each self-governing status of the component entities is typically independent and may not be altered by a unilateral decision of the "central government" [3]. Federation is also a concept which is adopted in many information systems. Considering small/medium independent self-governing cloud providers, federation means a cooperation enabling the sharing of part of their computational and storage resources with the purpose to provide new business opportunity. The advantage of a federated cloud scenario is twofold. On one hand, small/medium cloud providers, which rent resources to other providers, can optimize the use of their infrastructure, which often is under utilized, at the same time earning money for the use of their resources. On the other hand, external small/medium cloud providers can elastically scale their logical virtualization infrastructure, borrowing resources and paying other providers for their use. Therefore, cloud federation allows another form of pay-per-use economic model for ICT companies, universities, research centers and organizations that usually do not fully exploit the resources of their physical infrastructure. The benefits of cloud federation include provisioning of distributed cloud-based services, resource sharing, resource optimization and power saving [4].

However, several issues have to be faced from the point of view of security. Security is a wide topic in cloud computing and in this work we specifically focus on the establishment of trusted relationships between clouds, that can become very hard to be managed in scalable scenarios. Usually a trusted relationship among two or more systems is performed by means of authentication mechanisms.

In this paper, we discuss two possible authentication scenarios for the establishment of trust contexts between federated clouds: 1) Single Sign-On (SSO) Authentication using the traditional Identity Provider/Service Provider (IdP/SP) model; 2) Single Sign-On (SSO) authentication using a system of distributed Identity Providers (IdPs) with delegated authentications.

The paper is organized as follows: Section II describes the state of the art in authentication for distributed system, focusing on the IdP/SP model. In Section III, we analyze in detail the two authentication scenarios. A comparison between them is discussed in Section IV. Conclusions are summarized in Section V.

 ISBN: 978-1-61208-153-3

II. Related Works

A. Authentication Systems

With the term "authentication" we refer to any process by which it is possible to verify that someone is who claims to be. Username/password is the most widely used form of authentication. Another method is based on the private/public key cryptography. A stronger form of authentication is based on digital certificate [5], an electronic document which uses a digital signature to bind a public key with an identity described by information such as the name of a person or an organization.

Considering the evolving Internet scenarios, where entities need to access different services in a dynamic fashion, the requirement of interoperability among authentication technologies, also reducing the number of needed credentials and authentications is more and more compelling. To this regard, the latest trend in term authentication is represented by the Identity Provider/Service Provider (IdP/SP) model along with the Security Assertion Markup Language (SAML) [6], an XML-based standard that allows to exchange authentication and authorization data. The IdP/SP model allows to plan Single Sign-On (SSO) authentication scenarios when software/human entities can login once an IdP gaining the access to all SPs which rely on that target Idp.

Although SSO and SAML technologies are strictly related to the web context, some recent works are trying to employ the same approach on new scenarios where many entities that belong to different domains need to perform authentication [7], [8]. This is also the case of cloud federation [1], [9], where clouds cooperate together establishing trust contexts in order to provide new business opportunities. Recently, trust has been identified as a beneficial concept in large scale networks [10]. Considering trust relations when selecting service providers as partners leads to more efficient cooperation and composition of services [11].

SAML, offers the possibility of adding extensions in order to achieve dynamic federation in a generic way, regardless the specific scenario where it is applied. Considering federated cloud environments, in [12] it is discussed a new SAML profile named Cross-Cloud Authentication Agent SSO (CCAA-SSO) defining the steps needed for a secure cloud SSO authentication. However, the bottleneck of the IdP/SP model is represented by the presence of a central IdP per trust context. In order to overcome such a limitation, an approach [13] is proposed to minimize the dependence on central IdPs with a priori configuration, making entities more autonomous and capable of taking trust decisions. Another solution is exploiting the concept of delegation. Unfortunately, SAML natively lacks of delegation capabilities. Nevertheless, there are several works in Grid, Web Service, and Ubiquitous Computing environments where SAML is extended with the purpose to benefit of delegation capabilities [14], [15], [16].

B. Propagation of Trustiness

Scenario we are addressing can be defined as simplified context for trustiness in Cooperating Clouds, thus because Clouds may strongly leverage IdPs entities. Many works have been done in area of trustiness even in the propagation of trustiness. Of course our concept of delegation relies on some pre assumptions, those are: a) Each Cloud Provider uses well-know IdP (either its own or in the shared configuration). b) Each Cloud is able to decide if use/not-use the delegation against some specific IdPs (it may perform its filtering of IdPs existing in trustiness chains).

The complexity of evaluating the level of trust of a Subject insisting in the Internet determines to carefully face the topic, especially for large networks (i.e., Social Networks). Huang [17] developed a framework of trust propagation schemes evaluating them on a large trust network consisting of 800K trust scores expressed among 130K people. An interesting work has been conducted in [18] about the *Ontology of Trust* reporting a formal semantics and defining the concept of *Transitivity*. The authors highlighted that Trust Transitivity is not always an applicable concept at all. Chen et al [19] tried to determine a formula for expressing the trustiness. In particular they introduced the Mean operator (*Transitive mean degree*), that is the trust degree of a path (from source to destination considering the weights of edges existing in the between). It is calculated with the geometric or arithmetic mean of those weights of all edges along that path.

The security and trustiness in distributed environments are topics widely assessed. Our main aim is to investigate Clouds and adopt existing security solutions for overcoming issues related to the federation.

III. Authentication Between Clouds

Due to the high dynamism of federated cloud environments, a flexible method for building dynamic trust contexts should be provided.

According to [1], in this work, we assume that, regardless the adopted cloud middleware, the federation process is accomplished according to three different phases that is: Discovery, Match-Making and Authentication. In our solution a specific module named Cross-Cloud Federation Manager (CCFM) including three agents is able to perform such activities.

The *Discovery Agent (DA)* manages the discovery process of the resources and services made available by all the clouds belonging to the dynamic distributed environment. Once the clouds' service discovery has been performed, the *Match-Making Agent (MA)* will accomplish the task of choosing the more convenient cloud(s) wherewith to establish the federation according to requirements and policies. Finally, the *Authentication Agent (AA)* will perform the authentication with the selected clouds, creating a trust context, hence a federation. Once the security context has been created, a

 ISBN: 978-1-61208-153-3

cloud will be able to exploit resources and services offered by other federated cloud. In this Section, we focus on the authentication phase debating two different scenarios involving IdPs. In order to clarify the idea on using the IdP/SP model, we consider the following as a basic example: according to the IdP/SP terminology the AA of cloud A borrowing resources plays the role of "client", the AA of cloud B lending resources plays the role of "SP" (Relying Party), and IdP X plays the role of trust third party (Asserting Party) assuring to cloud B that cloud A is which claims to be. In order to allow cloud A to be authenticated by cloud B, it is needed that cloud A is enrolled in IdP X and that cloud B relies on IdP X. Once the authentication has been accomplished, cloud A will be able to log-in all clouds relying on IdP X without further authentications.

A. Traditional IdP/SP Scenario for Cloud Federation

Assuming an ecosystem with N clouds, the most obvious approach consists of using $M, M < N$ different IdPs. Basically, we can distinguish three main cases:

1) *Case 1.* $M = 1$. It is the simplest case. It consists of using a unique IdP for all federated clouds, thus enabling SSO authentications. In this way each cloud has to manage only one credential. However, this solution is out of place, because a unique central IdP would be a bottleneck for the whole authentication system.
2) *Case 2.* $M < N, M \neq 1$. It is the case in which a cloud, in order to perform authentications with the other $N-1$ clouds, has to perform an enrollment on M IdPs, thus managing M different credentials. For example, let us consider three different *IdPs X, Y, Z*, and clouds $1, 2, 3, 4, 5, 6, 7, 8, 9$. Clouds $1, 2, 3, 4$ rely on *IdP X*, clouds $5, 6, 7$ rely on *IdP Y*, and clouds $8, 9$ rely on *IdP Z*. In order to allow cloud 10 to be authenticated on all the other clouds, it has to perform enrollments on *IdP X, Y, and Z*, thus managing three credentials.
3) *Case 3.* $M = N - 1$. In this case each of the $N-1$ clouds rely on a different IdP. A cloud, in order to perform authentications with the other $N-1$ clouds, needs to perform enrollments on $N-1$ IdPs, thus managing $N-1$ different credentials.

In cases 2 and 3, if an IdP is corrupted, it will not affect the whole authentication system, however case 3 represents the worst case from the point of view of needed trust relationships, i.e, enrollments of clouds in IdPs. In this paper, we analyze this latter case. Considering a federation including N clouds, the number of trust relationships t_r needed to obtain the total overlay (i.e., each cloud is authenticated with each other) can be computed as:

$$t_r = N(N-1) \quad (1)$$

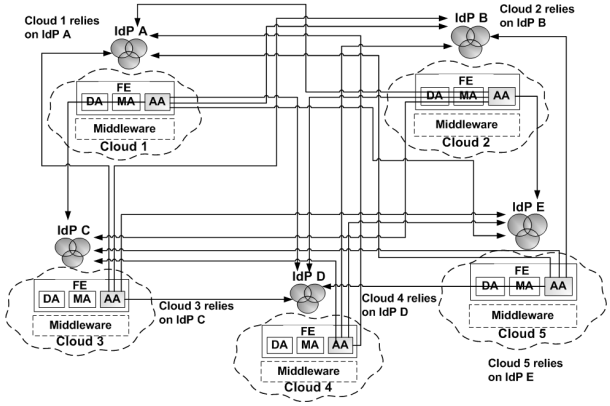


Figure 1. Authentication between clouds using the traditional IDP system where each cloud relies on a different IdP (worst case from the point of view of needed trust relationships).

In Figure 1, 5 clouds are depicted with their associated IdPs. Using eq. (1), $t_r = 5 * 4 = 20$. This means that the full overlay of the network can be reached after the establishment of 20 trust relationships (i.e., performed enrollments of clouds in IdPs). In Figure 1, the existence of a trust relationship is indicated by an arrow connecting the AA of the CCFM of each cloud with the corresponding IdP(s), where the cloud has an enrollment. For example, in order to allow cloud 1 to be authenticated in clouds 2, 3, 4, and 5, it as to perform enrollments in IdPs B, C, D, E. All the consideration we will assume in the following are based on the possibility of extending the SAML protocol as described in some recent works we have cited [13], [14], [15] and [16].

B. Distributed IdP Trusted Network (DIdP-TN) Scenario for Cloud Federation

As the authentication based on the traditional IdP system can imply high management costs especially in case 3, starting from the idea of delegation, we investigated an alternative authentication scenario able to reduce the number of required authentications in a federated cloud environment. We named such a system Distributed IdP Trusted Network (DIdP-TN). As depicted in Figure 2, the authentication system is based on the concept of delegation between IdPs. Cloud 1 has an enrollment on IdP A and therefore is able to perform a SSO authentication on cloud 2 and 3. As trusted relationships exist between IdP A, B, C, D, E, cloud 1 is also able to perform a SSO authentication on all the clouds of the federation. For example, as clouds 6, 7 rely on IdP E, cloud 1 is able to perform an authentication on cloud 7, because IdP E trusts IdP B and IdP B trusts IdP A. In this scenario, trust relationships have to be managed by the DIdP-TN and not by clouds themselves as in the traditional scenario. In this case the number of trust relationships t_r needed to obtain the full coverage of the network of clouds

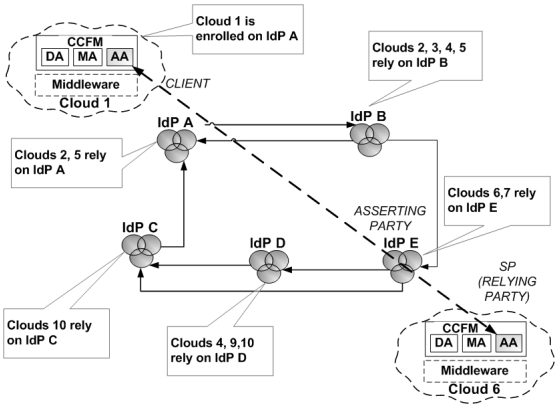


Figure 2. Authentication between clouds using the DIDP-TN system.

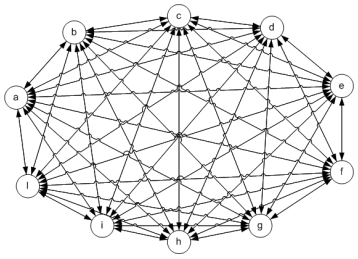


Figure 3. Example of digraph representing the traditional IDP-based authentication system for a federated cloud environment, where each cloud relies on a different IdP (worst case from the point of view of authentication management).

will be:

$$t_r \leq N(N-1) \tag{2}$$

IV. Evaluation of the DIdP-TN

In order to compare the DIDP-TN with a traditional IDP/SP authentication system for cloud federation, we modeled them using the graph theory and performed several experiments.

A. Modeling the two Authentication Systems

Let V and $E(V) = \{\{a,b\}$ *(for simplicity ab)* $|a,b \in V, a \neq b\}$ two finite sets. We define a pair $G = (V,E)$ with $E \subseteq E(V)$ the cyclic digraph (or directed graph) representing a cloud federation. The elements of V are vertices of G, and those of E the edges of G. Vertices a and b are adjacent if the edge $ab \in G$. The vertices set of the digraph G is denoted by V_G and its edge set by E_G. The number $\upsilon_G = |V_G|$ of vertices is called the *order* of G, and $\epsilon_G = |E_G|$ is the *size*. The E are oriented, that is, the edges are oriented: $E \subseteq V \times V$ where $ab \neq ba$. The digraph does not allow loops, that is, it is not allowed an edge aa. Let $e_i = v_i v_{i+1} \in G$ be edges of G for $i \in [1,k]$. The sequence $W_G = e_1 e_2 \dots e_k$ is a walk of length k from v_1 to v_k. Here e_i and e_{i+1} are compatible in the sense that e_1 is adjacent to e_{i+1} for all $i \in [1,k-1]$. We will write $a \to b$ if it exists at least one walk between a and b. We denote with $\omega_G = |W_G|$ the number of walks from a vertex a to a vertex b. A digraph will be named *complete* if $\forall a,b \in V_G$, a is adjacent with b. In this case, if $\upsilon_G = N$, it will be $\epsilon_G = N(N-1)$. Furthermore, a digraph will be named *connected*, if $\forall a,b \in V_G$ it exists at least one walk $a \to b$.

Let G_{IdP} a subgraph of the graph G, denoted by $G_{IdP} \subseteq G$, if $V_{IdP} \subseteq V_G$ and $E_{IdP} \subseteq E_G$. G_{IdP} represents the traditional IdP/SP authentication system in a federated cloud environment. G_{IdP} is built according to K events. An event represents the need of authentication of cloud a in cloud b, and each oriented edge $ab \in E$ represents a trust relationship, i.e., the enrollment of cloud a in the IDP on which cloud b relies, so that a will be read cloud and b IdP. Given an event with two equiprobable random vertices a,b, $a \neq b$, if a walk of length $k = 1$ exists, that is, if an edge ab exists, nothing is done; else an edge ab is created. Considering the set F with all the clouds belonging to the federation with $\upsilon_V = N$ each cloud $a \in F$, in order to be federated with the other $N-1$ clouds, must have a walk of length $l = 1$ toward all the other $N-1$ clouds of the federation. This implies that the digraph representing the federation has to be connected, so that $N(N-1)$ trust relationships (i.e., enrollments of cloud in IdPs) have to be performed. In this case, considering G_{IdP}, $\omega = \epsilon = N(N-1)$ is the number of needed trust relationships t_r in order that each clouds is able to be authenticated in each other. Figure 3 depicts an example of digraph representing the authentications in a federated clouds environment with total overlay using the traditional IDP-based system with $\upsilon = 10$ and $\omega = \epsilon = 90$.

Let $G_{DIdP-TN}$ a subgraph of the graph G, denoted by $G_{DIdP-TN} \subseteq G$, if $V_{DIdP-TN} \subseteq V_G$ and $E_{DIdP-TN} \subseteq E_G$. $G_{DIdP-TN}$ represents a DIdP-TN in a federated clouds environment. $G_{DIdP-TN}$ is built according to K events. An event represents the need to establish a trust relationship, i.e., an agreement between two IdPs, and each oriented edge $ab \in E$ represents the a trust relationship between IdP a and IdP b where delegated authentications take place. Given an event with two equiprobable random vertices a,b, $a \neq b$, if it exists one and at least one walk from the vertex a to the vertex b, nothing is done, else an edge ab is created. The meaning of each element $ab \in V$ of $G_{DIdP-TN}$ is the following: if we read ab, it will be read the IdP b is trusts IdP a. It is important to notice that as we are considering a digraph, if IdP a trusts IdP b it does not mean that IdP b trusts IdP a. This implies that the digraph representing the DIdP-TN has to be only complete (and not connected as in the previous scenario). In this case, a walk $a \to b$ of length $1 \leq l \leq N-1$ from IdP a to the IdP b represents a trust relationship between the two IdPs. Note that in this case considering $G_{DIdP-TN}$, $\epsilon \leq \omega \leq N(N-1)$, that is, the number of needed trust relationships between IdPs is less or equal to

 ISBN: 978-1-61208-153-3

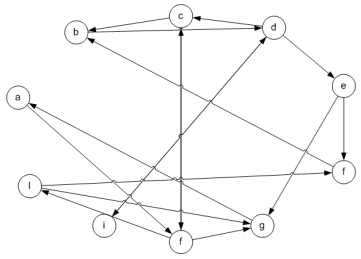


Figure 4. Example of digraph representing a distributed system of IdPs with delegated authentication in federated cloud environment.

t_r. Figure 4 depicts an example of a digraph representing the trusted relationships between IdPs, by means of each cloud is able to perform a SSO authentication on each other.

B. Comparison Between the two Authentication Approaches

For each authentication scenario we built a digraph creating edges according to the simulation of K events. For both graphs, we assumed an order $v = 25$. In simple terms, we considered a scenario including 25 Idps. For each event $i, i = 1, 2, \ldots, K$, we stored the number ϵ_i of edges created and the percentage X_i of total overlay on the whole digraph up to event i as:

$$X_i = \frac{\epsilon_i \cdot 100}{N(N-1)} \quad (3)$$

The total overlay is a parameter indicating how clouds cover the network of federated clouds from the point of view of authentications. The 100% of total overlay is obtained when each cloud of the federation is able to perform the authentication with all the other ones.

For simplicity, all the simulations have been performed with equiprobable events, and without the possibility of cancellation of a created edge, i.e., without the possibility to break trusted relationships. For each of the two authentication scenarios, we assumed 25 IdPs and 8000 events, repeating the simulations 50 times, picking up the mean values of both the created edges and the total overlay percentage for each $i - th$ event. For each simulation, we also calculated variances and confidence intervals at 95%. The goodness of our experiment is motivated by the fact that we have obtained confidence intervals rather small.

Figures 5 and 6 depict a comparison between the two authentication scenarios respectively considering the percentages of overlay on the whole cloud federation and the number of established trust relationships. In Figure 5, on the x-axis is reported the number of simulated events instead on the y-axis is reported the percentage of overlay on the whole cloud federation. Regarding the traditional IdP/SP authentication scenario, we obtained the 100% of overlay on the whole cloud federation after 6765 events (i.e, the need of establish authentications between clouds), instead in the case of the DIdP-TN we obtained the 100% of

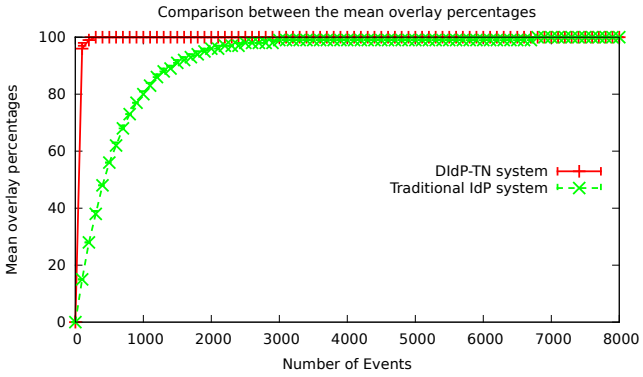


Figure 5. Comparison between the two authentication systems, considering the overlay percentages.

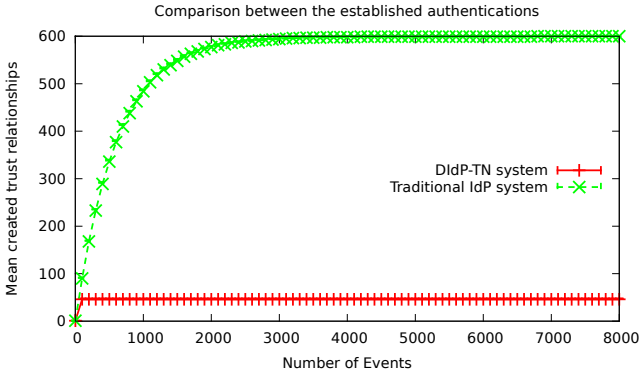


Figure 6. Comparison between the two authentication systems, considering the mean value of created authentications.

total overlay after 285 events (i.e., the need of establish agreements between IdPs). In Figure 6, on the x-axis is reported the number of simulated events, instead on the y-axis is the number established trusted relationships. We remark that for the traditional IdP/SP authentication scenario a trust relationship is an enrollment of a cloud in one IdP, and that in the case of DIDP-TN authentication scenario a trust relationship is an agreement between two IdPs. Regarding the traditional IdP system scenario, we can observe that we obtained a connected digraph after 6765 events. In fact, after 6765 events, we obtain $N(N-1) = 25 \cdot 24 = 600$ enrollments of clouds on IdPs. Instead, regarding the DIdP-TN system, we obtained a system in which each cloud is able to perform authentication on each other after 285 events, and 47,860 mean established agreements between IdPs. In both cases the variance had a Gaussian trend. This meant that the confidence intervals had their maximum amplitude around the midpoint of all the curves, before their saturation. Saturation is reached when each cloud is able to perform the authentication with each other, i.e., when the overlay percentage is 100%.

 ISBN: 978-1-61208-153-3

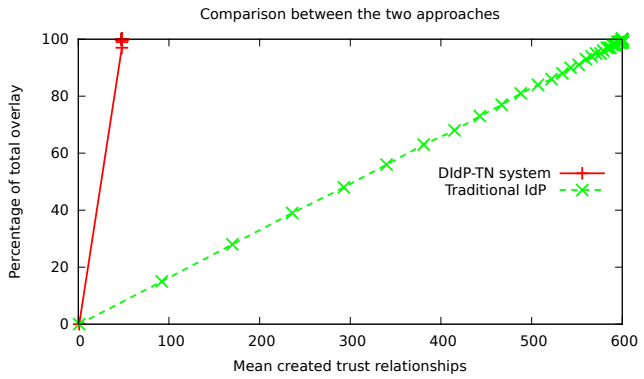


Figure 7. Comparison between the two approaches considering both the number of created authentications and the respective overlay percentages.

Figure 7 depicts a comparison between the two approaches considering both the number of created trust relationships on the x-axis and the respective overlay percentages on the y-axis. It is possible to notice how for the DIdP-TN system the 100% of overlay is obtained faster than the traditional IdP system.

C. Overcoming issues with Transitive Trust

The paper we are presenting is a preliminary work that needs to be refined. In the context of Transitive Trust, systems authentication performed along through the delegation mechanism might raise problems. In particular in our case a subset of IdPs that are nor recognized in the chain of trustiness of whatever cloud provider. To address such a problem we introduced the Access Control List (ACL) for preventing the involvement of untrusted IdPs no directly accesses but present in the list of delegation. Indeed there could be the possibility that even though a trust relationship exists from an IdP a to an IdP b through a cloud c, the cloud a decide to create a direct trust relationship with cloud b because it considers too much risky a delegated authentication through a cloud c. For example cloud a could consider cloud c not so reliable from the point of view of security. We are looking at a much more complex trustiness scenarios in which the links weight of trusting walks along with the IdP reputation must be taken into account (i.e., [20]).

V. Conclusions

In this paper, we focused on two authentication scenarios for federated cloud environments: the first based on the adoption of a traditional IdP system, and the second based on a DIdP-TN. From the simulations, it is evident how the DIdP-TN system allows to drastically reduce the needed operations for clouds, simplifying the management of accounts and enrollments. However, even if on one hand it is possible to reduce the number of needed authentications, on the other hand a few problems might rise. In this work, we assumed equiprobable events, but if we consider also the possibility of breaking the trust relationships, the scenario on one hand might be fault tolerant as alternative trust relationships (i.e., walks considering the digraph) might exist, whereas on the other hand the scenario might become more complicated.

References

[1] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *Proceedings of IEEE CLOUD '10*, pp. 337–345, IEEE, July 2010.

[2] April 2011. IEEE works towards cloud interoperability standards: http://www.cloudcomputingzone.com/2011/04/ieee-works-towards-cloud-interoperability-standards/.

[3] Forum of Federations: http://www.forumfed.org/en/index.php.

[4] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Munoz, and G. Toffetti, "Reservoir - when one cloud is not enough," *Computer*, vol. 44, pp. 44–51, 2011.

[5] "C. Adams and S. Farrell, Internet X.509 Public Key Infrastructure: Certificate Management Protocols, RFC 2510: http://tools.ietf.org/html/rfc2510."

[6] "Security assertion markup language, oasis, http://www.oasis-open.org/committees/security."

[7] K. Traw, S. Yang, and P. Comitz, "Federated identify management in service oriented architectures," in *Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pp. 1–6, May 2008.

[8] R. McKenzie, M. C. M, and C. Wallis, "Use cases for identity management in e-government," in *Security & Privacy, IEEE*, vol. 6, pp. 51–57, March-April 2008.

[9] Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," *Proceedings of IEEE Cloud '10*, pp. 123–130, 2010.

[10] D. Artz and Y. Gil, "A survey of trust in computer science and the semantic web," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 58 – 71, 2007.

[11] S. Florian, S. Daniel, and D. Schahram, "The cycle of trust in mixed service-oriented systems," in *Proceedings of SEAA '09*, pp. 72–79, 2009.

[12] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Three-phase cross-cloud federation model: The cloud sso authentication," *Second International Conference on Advances in Future Internet (AFIN)*, pp. 94–101, 2010.

[13] P. Arias Cabarcos, F. Almenárez Mendoza, A. Marín-López, and D. Díaz-Sánchez, "Enabling saml for dynamic identity federation management," in *Wireless and Mobile Networking*, vol. 308, pp. 173–184, Springer Boston, 2009.

 ISBN: 978-1-61208-153-3

[14] T. Komura, Y. Nagai, S. Hashimoto, M. Aoyagi, and K. Takahashi, "Proposal of delegation using electronic certificates on single sign-on system with saml-protocol," in *SAINT*, pp. 235–238, 2009.

[15] S. Shen and S. Tang, "Cross-domain grid authentication and authorization scheme based on trust management and delegation," in *CIS*, pp. 399–404, 2008.

[16] W. Jun, D. V. David, and H. Marty, "Extending the security assertion markup language to support delegation for web services and grid services," in *Proceedings of IEEE ICWS '05*, pp. 67–74, 2005.

[17] M. S. F. Jingwei Huang, "An ontology of trust: formal semantics and transitivity," in *ICEC*, pp. 259–270, 2006.

[18] J. Huang and M. S. Fox, "An ontology of trust: formal semantics and transitivity," in *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, ICEC '06, (New York, NY, USA), pp. 259–270, ACM, 2006.

[19] Y. Chen, T.-M. Bu, M. Zhang, and H. Zhu, "Max-minimum algorithm for trust transitivity in trustworthy networks," *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, vol. 3, pp. 62–64, 2009.

[20] Y. Chen, T.-M. Bu, M. Zhang, and H. Zhu, "Measurement of trust transitivity in trustworthy networks," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 4, 2010.

 ISBN: 978-1-61208-153-3

Testing the Suitability of Cassandra for Cloud Computing Environments

Consistency, Availability and Partition Tolerance

Felix Beyer, Arne Koschel,
Christian Schulz, Michael Schäfer
Faculty IV, Department for Computer Science
Applied University of Sciences and Arts
Hannover, Germany
{felix.beyer, christian.schulz2, michael.schaefer}@stud.fh-hannover.de, arne.koschel@fh-hannover.de

Stella Gatziu Grivas, Marc Schaaf
Institute for Information Systems
University of Applied Sciences Northwestern Switzerland
Olten, Switzerland
{stella.gatziugrivas, marc.schaaf}@fhnw.ch

Irina Astrova
Institute of Cybernetics
Tallinn University of Technology
Tallinn, Estonia
irina@cs.ioc.ee

Alexander Reich
BeEvolution GmbH
Hannover, Germany
alexander.reich@beevolution.de

***Abstract*—Since relational database management systems (DBMSs) are ill-suited to cloud computing environments, multiple efforts are now underway to offer a viable alternative to relational DBMSs. These efforts have led to the rise of a new kind of DBMSs called NoSQL. One of the most visible products in this rise is Cassandra. Cassandra is a NoSQL DBMS, which can also be used as a clustered file system. Cassandra was claimed to be particularly well suited for cloud computing environments. Our goal in this paper was to confirm or deny that claim. Towards this goal, we conducted tests on Cassandra to determine what levels of consistency, availability and partition tolerance can be achieved and if these can be achieved without sacrificing performance.**

Keywords—Cloud computing, Cassandra, consistency, availability, partition tolerance, experiments.

I. INTRODUCTION

Consistency, availability and partition tolerance are of great importance to cloud computing environments. These can be achieved by using relational or NoSQL database management systems (DBMSs). Since NoSQL DBMSs are still a new research area, various definitions exist that may even contradict each other. For this paper, we have chosen the following definition: NoSQL is a movement grouping all efforts, which intend to provide a viable alternative to (SQL-based) relational databases for storing and processing data [1].

Relational DBMSs [3] are 30 years old. They have been the dominant storage technology behind websites. The past few years have seen the emergence of cloud computing environments, which are going to be an increasingly common backbone for websites. But cloud computing environments and relational DBMSs do not fit well together [10]. In particular, relational databases can scale, but usually only when this scaling happens on a single node (i.e., vertical scaling). When the capacity of that single node is reached, relational databases need to scale horizontally and be distributed across multiple nodes over a network. This is when the suitability of relational DBMSs for cloud computing environments is reduced.

A. Consistency

Consistency guarantees that every node in the cluster has the same view on data. So once one node has written some data, all other nodes in the cluster will see those data.

The importance of consistency for cloud computing environments is perhaps best explained by example. Consider an airline company that provides a booking website. Assume that the airline company's database is distributed over a network, so data can be accessed from different nodes. Consistency is endangered now because one node may change data without knowing about the changes have been made by other nodes. In particular, assume that a customer opens a session on the booking website and a last available seat for the selected flight is displayed to the customer. This seat has already been booked, but the node serving the customer's session does not know about it yet. The result is that the customer can still book the last seat. Next time when the nodes synchronize each other, inconsistency shows up as there will be two bookings for one and the same seat.

To avoid a situation like the above, NoSQL DBMSs should provide consistency. Relational DBMSs typically use ACID (**A**tomicity **C**onsistency **I**solation **D**urability) transactions for this purpose. But ACID transactions are not distributed-system friendly. Therefore, NoSQL DBMSs typically either skip them entirely or comply with BASE (**B**asically **A**vailable **S**oft-state **E**ventual Consistency).

Compliance with BASE means that the latest version of data on one node might not match that on other nodes; so every node in the cluster is only guaranteed to see writes eventually. As a result, NoSQL DBMSs might not handle long running business processes [6] like booking flights, where the current state of data, e.g., seats availability on the plane, should be shown to all other customers while one

 ISBN: 978-1-61208-153-3

customer, who is booking a flight, has not finished the booking yet.

B. Availability

Availability guarantees that if one node fails, there will still be some copies of data on other nodes in the cluster, so the availability of the whole cluster will not be endangered by that node failure.

Continuing the previous example, assume that the node serving the customer's session experiences a failure during which the customer cannot book the last seat anymore.

To avoid a situation like the above, NoSQL DBMSs should provide availability. Relational DBMSs typically use replication for this purpose. The same technique is used by NoSQL DBMSs.

C. Partition Tolerance

Partition tolerance guarantees that the cluster remains operational even when communication between nodes in the cluster is lost.

Continuing the previous example, assume that the airline company's database is running on multiple nodes across a network. Also, assume that a network connection with the node serving the customer's session is lost due to a network failure. The database is now partitioned. If the database is tolerant of it, then the cluster can still perform read and write operations, i.e., the customer can still book the last seat. If not, the cluster will be completely inaccessible.

To avoid a situation like the above, NoSQL DBMSs should provide partition tolerance – they typically use quorum for this purpose. Being single-node, relational databases cannot be partitioned.

II. CONTRIBUTION

In this paper, we deal with using NoSQL DBMSs in cloud computing environments. Unlike many other papers, we do not focus on traditional approaches that use clustered file systems like Gluster [2] or relational DBMSs like MySQL and Oracle. Rather, we introduce a novel approach that uses Cassandra.

Cassandra [5] was claimed to be particularly well suited for cloud computing environments. Our goal was to confirm or deny that claim. For this purpose, we experimented with Cassandra. In particular, we built a test setup, developed a test application and conducted tests on Cassandra using this application.

III. CASSANDRA

Cassandra is a recently upcoming NoSQL DBMS that can also be used as a clustered file system [4]. It was originally developed as an open source by Facebook in 2007 to horizontally scale their internal application; viz. Inbox Search. Later in 2009 Facebook released Cassandra to Apache. This allowed Cassandra to move forward in the direction that is more general to the public than just to Facebook's in-house needs.

Recently, Cassandra has acquired great popularity and showed high potentials for cloud computing. This is because Cassandra offers a variety of possibilities to provide the desired levels of consistency, availability and partition tolerance.

A. Consistency

In Cassandra, every operation is assigned a consistency level, so that it can be decided whether the consistency should be guaranteed among all nodes in the cluster or it is acceptable if some node might not contain the latest version of data, e.g., in case of a node failure. In particular, Cassandra supports the following consistency levels:

ANY: $W + R > N$
ONE: $W = 1$ or $R = 1$
QUORUM: $W = Q$ or $R = Q$
ALL: $W = N$ or $R = N$,

where R is the number of records to read (i.e., the number of reads on a replica), W is the number of records to write (i.e., the number of writes on a replica), N is a replication factor and $Q = N / 2 + 1$.

Even though Cassandra complies with BASE, it is still possible to have ACID transactional consistency guarantees using ZooKeeper [7], a coordination service for distributed systems. For short running business processes, single path locking can be used (classes `ZkReadLock` and `ZkWriteLock`). However, in distributed systems with many interactions, the use of single path locking is not recommended since it often results in deadlocks. It is better to use multi-path locking (a class `ZkMultiLock`) since this class contains methods, which check for deadlocks and handle them before they occur. A downside of multi-path locking is decreased performance. For simple applications, both single and multi-path locking is sufficient to ensure consistency. More complex applications, however, require the use of a class `ZkTransaction`. This class works in conjunction with `ZkMultiLock`. It provides a simplified Thrift API, which allows for specification of a series of data mutation operations to be performed by a transaction. After the transaction has been specified, a method `commit` is executed with an instance `ZkMultiLock` passed it as a parameter. At this point, cages will add a reference to a transaction node, which is created by ZooKeeper. Next, the transaction can read the current values of the data, which are to be updated. At this point, the original state will be written into the transaction node [8]. Once this has been done, the data mutations will be performed. After that, all references to the transaction node from within the locks will be removed. The transaction node gets deleted and the transaction itself has been committed.

If the node fails during the execution of a sequence of individual data mutations, the cages will immediately be unlocked. The transaction, which has already been executed, will be rolled back to the "written before" state in the transaction node. So the state of the database will be identical to the original state before the node has performed its operations. This guarantees consistency of the database and complies with so-called relaxed ACID since changes one node makes during a long running business process will be seen by other nodes in the cluster [9].

 ISBN: 978-1-61208-153-3

B. Availability

In Cassandra, availability is achieved through replication. Every node in the cluster that needs access to data has its own replica, so a failure of one node will not make all replicas unavailable at the same time.

C. Partition Tolerance

In Cassandra, partition tolerance is achieved through quorum (e.g., if one node is separated from the other two nodes in the cluster, it stops processing).

IV. TEST SETUP

The test setup consisted of a cluster having two nodes: primary and secondary. Writes are directed at both nodes, while reads are directed to just one of the nodes, which is known as the primary node. Because the other node is kept updated, it is known as a secondary node. It is always ready to take over. If the primary node should fail or become inaccessible for any reason, Cassandra will redirect reads to the secondary node and processing will continue uninterrupted. Before the failed node comes back on line, any interim updates will be applied to synchronize it with the other node.

A. Cluster Infrastructure

To configure the first node, we adjusted some variables in the configuration file. In particular, we set both `ThriftAddress` and `ListenAddress` to the IP address of the first node to enable intra-cluster communication and data access. (The database was accessed using Thrift API.) Also, we set `ReplicationFactor` to a value that was equal to the number of nodes in the cluster (i.e., 2) to ensure that a failure of one of the nodes would not make both replicas unavailable at the same time. (In general, the cluster can be configured with more than two replicas, depending on the probability of failures and the requirements for availability.)

For the second node, we set both `ThriftAddress` and `ListenAddress` to the IP address of the second node. In addition, we set `Seed` to the IP address of the first node so that the second node would know to which server it had to connect for getting data when it was added to the cluster. Finally, we set `AutoBootstrap` to true. This resulted in the second node being added to the cluster automatically. (If a new node is added, only seed nodes in the cluster need to be configured, instead of adjusting all node configurations.)

After the cluster configuration had been completed, we checked if the two nodes would connect to each other. We did it by using a command `ring`, which returned a list of all available nodes. Although this check showed that the two nodes were available in the cluster, we analyzed entries in the log file generated by Cassandra to see if the cluster remained operational over some period of time.

The following listing shows an excerpt from the resulting log file:

```
INFO  16:50:25,966  Starting up server gossip
INFO  16:50:26,045  Binding thrift service to 192.168.5.132:9160
INFO  16:50:26,050  Cassandra starting up ...
DEBUG 16:50:26,132  attempting to connect to 192.168.5.134
INFO  16:50:26,160  Node 192.168.5.134 is now part of the
                    cluster
DEBUG 16:50:26,161  Resetting pool for 192.168.5.134
DEBUG 16:50:26,793  attempting to connect to 192.168.5.134
INFO  16:50:26,798  InetAddress 192.168.5.134 is now UP
INFO  16:50:26,800  Started hinted handoff for endpoint
                    192.168.5.134
INFO  16:50:26,811  Finished hinted handoff of 0 rows to
                    endpoint 192.168.5.134
```

As can be seen, the second node (192.168.5.134) was added to the cluster, and a synchronization process called `hinted handoff` was started and finished.

B. Test Database Schema

Cassandra supports a data model that is based on column families. A column family is a container for columns, analogous to a table in relational DBMSs; it holds the columns as an ordered list (a column family row), which can be referenced by the column name. There are two kinds of column families: simple and super. Simple column families consist of columns, which are grouped. Super column families can be viewed as a column family within another column family.

In Cassandra, a database is a distributed multi-dimensional map, which is indexed by a key. The top dimension is referred to as a key space and under this key space, column families follow. The key space is divided up by a cluster into ranges delimited by tokens.

In Cassandra, a database schema is flexible, meaning that we do not have to decide what columns we need in the records ahead of time. Rather, we can just add or delete columns on the fly. This is by contrast to relational DBMSs, where a database schema is fixed and pre-defined.

In the test setup, we used a simple database schema `Address`. There was only one key space `Keyspace1` containing a column family `Standard2`, which in its turn contained the following columns: `firstname`, `lastname`, `street`, `housenumber`, `zip`, `city`, and `country`. To populate the column family with data sets, we used the following statements:

```
setKeyspace1.Standard2["1"]["firstname"]="MyFirstname"
setKeyspace1.Standard2["1"]["lastname"]="MyLastname"
setKeyspace1.Standard2["1"]["street"]="MyStreet"
setKeyspace1.Standard2["1"]["housenumber"]="MyHouseNumber
"
setKeyspace1.Standard2["1"]["zip"]="MyZip"
setKeyspace1.Standard2["1"]["city"]="MyCity"
setKeyspace1.Standard2["1"]["country"]="MyCountry"
```

In this listing, the key value was set to 1. However, for any next data sets, this value was increased by one in order to differentiate the data sets from each other.

 ISBN: 978-1-61208-153-3

V. TEST APPLICATION

To experiment with Cassandra, we developed a test application in Java. This application took the following arguments as input: a node IP, a Cassandra port, a command to be performed (viz., `put`, `delete` or `get`), data for the command and optionally a key ID of the data. The test application consisted of the following classes.

A. SelectClient

This class was used to determine the time periods for every method execution.

B. CassandraClient

This class was used to open and close a connection to the database.

C. PutCassandraData

This class was used to insert data into the database. The class had a method `putDataIntoCassandra`, which defines the column names, generates new records and adds them to the database. The record generation was performed by a random generator, which combines data from the specified lists, and could be repeated any number of times using a loop.

D. GetCassandraData

This class was used to retrieve records from the database. Retrieving records was performed by the following methods:

- `getKeyList`, which sets a range for the specified key space and gets a key range from Cassandra.
- `getData`, which reads all records in the specified key range and returns the result.
- `getDataByKey`, which defines a slice range, reads one specific record identified by its key ID and returns the result.
- `printData`, which displays on the shell all records in the specified maximum range.
- `printDataByKey`, which displays on the shell one specific record identified by its key ID.

E. DeleteCassandraData

This class was used to remove records from the database. Removing records was performed by the following methods:

- `deleteCassandraData`, which creates a key range and deletes all records in the specified key range.
- `deleteCassandraDataByKey`, which deletes one specific record identified by its key ID.

VI. EXPERIMENTS

After setting up the cluster infrastructure, we performed the following test cases using the test application. After each test case, we analyzed the log file entries generated by Cassandra.

A. Test Case 1: Putting Data to Database

In this test case, we checked if records could be inserted into the database. For this purpose, we tried to add data to the first node.

The following listing shows an excerpt from the resulting log file for the first node:

```
DEBUG  16:52:47,373  insert
DEBUG  16:52:47,381  insert writing local key 1
DEBUG  16:52:47,383  insert writing key1 to 432@192.168.5.134
DEBUG  16:52:47,391  Processing response on a callback from
                     432@192.168.5.134
```

At first, an insert was executed, following by a local write. Then a remote write was executed, following by a response from the second node (192.168.5.134) to check if this node had received the data.

B. Test Case 2: Getting Data from Database

In this test case, we checked if records could be removed from the database. For this purpose, we tried to read data from the first node.

The following listing shows an excerpt from the resulting log file for the first node:

```
DEBUG  16:53:42,116  range slice
DEBUG  16:53:42,117  RangeSliceCommand{keyspace
       ='Keyspace1', columnfamily='Standard2',
       supercolumn=null, predicate=SlicePredicate(
       columnnames:[[B@1b7c76]),
       range=[0,0], maxkeys=1}<somerangesliceoutput>
DEBUG  16:53:42,191  get slice <somegetsliceoutput>
DEBUG  16:53:42,203  Reading consistency digest for 1
       from 606@[192.168.5.134,192.168.5.132]
```

At first, a range slice was executed; it set the key space, the column family and the range. It was followed by a get slice, which collected the requested data. An entry `reading consistency digest` in the log file indicated that the database was checked for consistency.

C. Test Case 3: Deleting Data from Database

In this test case, we checked if records could be removed from the database. For this purpose, we tried to delete data from the first node.

The following listing shows an excerpt from the resulting log file for the first node:

```
DEBUG  16:54:04,475  remove
DEBUG  16:54:04,476  insert writing local key 1
DEBUG  16:54:04,477  insert writing key 1 to 676@192.168.5.134
DEBUG  16:54:04,480  Processing response on a callback
                     from 676@192.168.5.134
```

At first, a remove was executed, following by a local write, which set the data values to null. Then a remote write was executed, following by a response from the second node (192.168.5.134) to check if this node set the data to null. Thus, deleting data was somehow similar to adding data.

 ISBN: 978-1-61208-153-3

D. Test Case 4: Consistency

In this test case, we checked if all nodes in the cluster had the same view on data even in the presence of updates. For this purpose, we added some data to the first node and tried to read the data back from the second node.

The following listing shows an excerpt from the resulting log file for the first node:

```
DEBUG 18:09:55,489 Adding hint for 192.168.5.134
      <some row mutation operation which adds new data on
       the first node>
DEBUG 18:11:29,284 Node 192.168.5.134 state normal, token
      115100908670755235738753006493737225538
INFO  18:11:29,284 Node 192.168.5.134 state jump to normal
INFO  18:11:29,284 Will not change my token ownership to
      192.168.5.134
INFO  18:11:29,284 Started hinted handoff for endpoint
      192.168.5.134 <some data mutation operation>
INFO  18:11:29,385 Finished hinted handoff of 2 rows to
       endpoint 192.168.5.134
```

At first, some data mutation was performed. Then a token was sent to the second node, following by starting and finishing a synchronization process with the second node (192.168.5.134) as the endpoint.

The following listing shows an excerpt from the resulting log file for the second node:

```
DEBUG 16:58:13,064 Node 192.168.5.132 state normal, token
      115100908670755235738753006493737225538
      <some row mutation operation which adds the changed
       data of the first node>
INFO  16:58:13,344 Started hinted handoff for endpoint
      192.168.5.132
INFO  16:58:13,351 Finished hinted handoff of 0 rows to
      endpoint 192.168.5.132
```

At first, the token was received from the first node. Then some data mutation was performed, following by starting and finishing another synchronization process with the first node (192.168.5.132) as the endpoint. After the synchronization process had finished, the data on the second node were one and the same as on the first node, thus indicating that the database was in a consistent state.

It should be noted that since we wrote data with a consistency level of `ONE` and wanted to get the same data back while reading, we read the data with a consistency level of `ALL`.

E. Test Case 5: Availability

In this test case, we checked if the database was available even in the presence of node failures. For this purpose, we disconnected the first node to simulate its failure and tried to read data from the second node to see if some copy of the data was still available.

Since data were replicated within a single cluster, they were available even after the first node had been disconnected. The performance for a read operation became half as fast as before. But this was fine for a two-node cluster.

F. Test Case 6: Partition Tolerance

In this test case, we checked if the database was tolerant to partitions in the presence of network failures. For this purpose, we disconnected the second node to simulate a loss of a network connection between the two nodes and tried to write data with a consistency level of `ONE` to the first node to see if that node could still process the write (even knowing that data on the second node could not be updated immediately).

The following listing shows an excerpt from the resulting log file for the first node:

```
DEBUG 18:11:29,116 range slice
DEBUG 18:11:29,117 RangeSliceCommand{keyspace
      ='Keyspace1', columnfamily='Standard2',
      supercolumn=null, predicate=SlicePredicate(
      columnnames:[[B@1b7c76]),
      range=[0,0], maxkeys=1}<somerangesliceoutput>
DEBUG 18:11:29,191 get slice <somegetsliceoutput>
DEBUG 18:11:29,460 Processing response on an async result
                   from 5678@192.168.5.134
```

As can be seen, the first node performed a write operation, thus favoring availability over consistency. An entry `async result` in the log file indicated that the second node would not know about interim updates until the network connection was restored.

In our next step, we repeated the same test but with a consistency level of `QUORUM`. Since the first node could not communicate with the second node to inform it about interim updates, the first node stopped processing the write, thus favoring consistency over availability. The cluster became read-only.

G. Test Case 7: Performance

In this test case, we checked if consistency could be achieved without sacrificing performance. For this purpose, we ran Test Case 1, Test Case 2 and Test Case 3 with 100, 1000, 10000 and 100000 data iterations.

We also experimented with different consistency levels to gain extra speed for read or write operations. For example, when we ran the tests with 10000 and 100000 data iterations, we were more concerned about write performance than read performance. Therefore, we wrote data with a consistency level of `ONE` (W=1) and read data with a consistency level of `ALL` (R=N). As a result, each read had to access all copies of data to determine which of them contained the latest version of data, whereas each write had to update only one copy of data. This time when we ran the tests with 100 and 1000 data iterations, we were more concerned about read performance than write performance. Therefore, we wrote data with a consistency level of `ALL` (W=N) and read data with a consistency level of `ONE` (R=1).

Figure 1 shows the result of our tests. As can be seen, consistency was achieved at expense of performance because

 ISBN: 978-1-61208-153-3

of the need for starting and finishing a synchronization process every time when the database was updated.

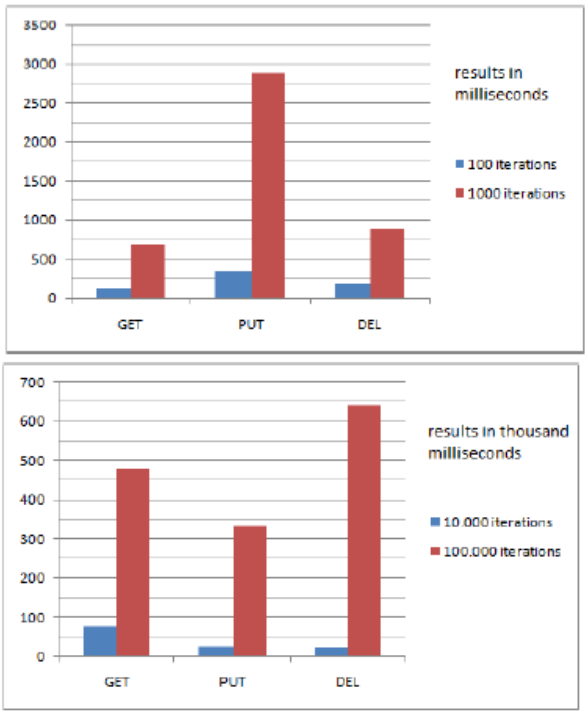


Figure 1. Performance test results.

VII. CONCLUSION

During many years clustered file systems like Gluster and (SQL-based) relational DBMSs like MySQL and Oracle have been the dominant technologies for providing an efficient and reliable data store in cloud computing environments. However, with the trend towards cloud computing, these systems get new competitors – NoSQL DBMSs. One of them is Cassandra, which was evaluated in this paper.

Cassandra was claimed to be particularly well suited for cloud computing environments. Our goal was to confirm or deny that claim. Towards this goal, we experimented with Cassandra. Our experiments showed that Cassandra did offer an efficient and reliable data store in cloud computing environments, either while favoring availability and partition tolerance over consistency or while favoring consistency and partition tolerance over availability.

The result of our experiments was in agreement with the CAP (**C**onsistency, **A**vailability and **P**artition tolerance) theorem [11]. This theorem simply states that out of consistency, availability and partition tolerance, a distributed system can choose to provide two but never three at the same time, as shown in Figure 2. For example, relational DBMSs typically provide both consistency and availability, but not partition tolerance. By contrast, NoSQL DBMSs typically provide both availability and partition tolerance, but not consistency.

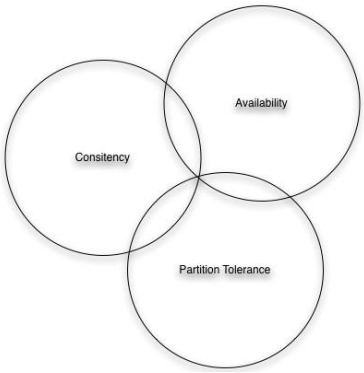


Figure 2. CAP theorem [12].

VIII. FUTURE WORK

In the future, we are going to increase a number of nodes in the cluster. Eventually applying the results of our tests to real-world applications is also part of our future work.

ACKNOWLEDGMENT

Irina Astrova's work was supported by the Estonian Centre of Excellence in Computer Science (EXCS) funded mainly by the European Regional Development Fund (ERDF).

REFERENCES

[1] Definition "NoSQL" term. http://data.story.lu/2010/11/16/definition-nosql-term, acc. 12.02.2011.

[2] Gluster. http://www.gluster.org/, acc. 12.02.2011.

[3] R. Elmasri and S. Navathe. Fundamentals of Database Systems (5th Edition). Addison Wesley, U.S.A, 2006.

[4] A. Lakshman and P. Malik. Cassandra - a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 44(2):35–40, April 2010.

[5] Cassandra. http://cassandra.apache.org/, acc. 17.04.2011

[6] U. Dayal, M. Hsu, and R. Ladin. Business Process Coordination: State of the Art, Trends, and Open Issues, 27th VLDB Conference, Roma, Italy, 2001.

[7] ZooKeeper, http://zookeeper.apache.org/, acc. 10.02.2011.

[8] P. Hunt, M. Konar, F. Junqueira, and B. Reed. ZooKeeper: wait-free coordination for internet-scale systems. 2010 USENIX conference on USENIX annual technical conference (USENIXATC'10). USENIX Association, Berkeley, CA, USA, 2010.

[9] ZooKeeper. http://ria101.wordpress.com/tag/zookeeper/, acc. 10.02.2011.

[10] T. Bain. Is the relational database doomed? http://www.readwriteweb.com/enterprise/2009/02/is-the-relational-database-doomed.php, acc. 21.10.2010.

[11] E. Brewer. Towards Robust Distributed Systems, PODC Keynote, July 19, 2000. http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf, , acc. 10.02.2011.

[12] M. Woodward. Caveats of Evaluating Databases. http://blog.mattwoodward.com/caveats-of-evaluating-databases-jan-lehnardt, acc. 21.10.2010.

 ISBN: 978-1-61208-153-3

Designing an Elastic and Scalable Social Network Application

Xavier De Coster, Matthieu Ghilain, Boris Mejías, Peter Van Roy
ICTEAM institute
Université catholique de Louvain
Louvain-la-Neuve, Belgium
{*decoster.xavier,ghilainm*}*@gmail.com* {*boris.mejias,peter.vanroy*}*@uclouvain.be*

***Abstract*—Central server-based social networks can suffer from overloading caused by social trends and make the service momentarily unavailable preventing users to access it when they most want it. Central server-based social networks are not adapted to face rapid growth of data or flash crowds. In this work we present a design for a scalable, elastic and secure Twitter-like social network application, called Bwitter, built on the top of a scalable transactional key/value datastore, such as Beernet or Scalaris. The application runs on a cloud infrastructure and is able to scale its resource usage up and down quickly to avoid overloading and resource wasting. We measure performance, scalability, and elasticity for our prototype and show it performs satisfactorily up to 18 nodes with realistic loads.**

Keywords-Scalability; elasticity; cloud application; social network; Twitter; Beernet; Scalaris; key/value store.

I. INTRODUCTION

Social networks are an increasing popular way for people to interact and express themselves. People can now create content and easily share it with other people. The servers of those services can only handle a given number of requests at the same time, so if there are too many requests the server can become overloaded. Social networks thus have to predict the amount of load they will have to face in order to have enough resources at their disposal. Statically allocating resources based on the mean utilisation of the service would lead to a waste during slack periods and overloading during peak periods. Twitter shows the "Fail Whale" graphic whenever overloading occurs [1]. This is a tricky situation as this load is related to many social factors, some of which are impossible to predict. For instance we want to be able to handle the high amount of people sending Christmas or New Year wishes but also reacting to natural disasters. This is why we want to turn towards scalable and elastic solutions, allowing the system to add and remove resources on the fly in order to fit the required load. This work focuses on the design of a social network with elastic and scalable infrastructure: Bwitter, a secure Twitter-like social network built on the transactional key/value store Beernet [2].

This paper summarizes the results of a master's thesis [3]. Section II defines the basic required operations for a Twitter-like social network. Section III explains why we chose a transactional key/value store, such as Beernet, for implementing Bwitter, and Section IV explains how to run multiple services on top of it. In this section we also discuss some possible improvements for DHTs in order to increase their security and offer a richer application programming interface. Section V presents the application design and Section VI gives our cloud-based architecture. Section VII describes the implementation of our prototype, and Section VIII evaluates its performance (including scalability and elasticity). We then conclude in Section IX.

II. A QUICK OVERVIEW OF REQUIRED OPERATIONS

Bwitter is designed to be a secure social network based on Twitter. Twitter is a microblogging system, and while it looks relatively simple at first sight it hides some complex functionalities. We included almost all of those in Bwitter and added some others. We will only depict the relevant functionalities here that will help us to analyse the design of the system and the differences between a centralised and decentralised architecture.

A. Nomenclature

There are only a few core concepts on which our application is based. A *tweet* is basically a short message with additional meta information. It contains a message up to 140 characters, the author's username and a timestamp of when it was posted. If the tweet is part of a discussion, it keeps a reference to the tweet it is an answer to and also keeps the references towards tweets that are replies to it. A *user* is anybody who has registered in the system. A few pieces of information about the user are kept in memory by the application, such as her complete name and her password, used for authentication. A *line* is a collection of tweets and users. The owner of the line can define which users he wants to associate with the line. The tweets posted by those users will be displayed in this line. This allows a user to have several lines with different topics and users associated.

B. Basic operations

1) Post a tweet: A user can publish a message by posting a tweet. The application will post the tweet in the lines to which the user is associated. This way all the users following her have the tweet displayed in their line.

 ISBN: 978-1-61208-153-3

2) Retweet a tweet: When a user likes a tweet from an other user she can decide to share it by retweeting it. This will have the effect of "sending" the retweet to all the lines to which the user is associated. The retweet will be displayed in the lines as if the original author posted it but with the retweeter's name indicated.

3) Reply to a tweet: A user can decide to reply to a tweet. This will include a reference to the reply tweet inside the initial tweet. Additionally a reply keeps a reference to the tweet to which it responds. This allows to build the whole conversation tree.

4) Create a line: A user can create additional lines with custom names to regroup specific users.

5) Add and remove users from a line: A user can associate a new user to a line, from then on all the tweets this newly added user posts will be included in the line. A user can also remove a user from a line, she will then not see the tweets of this user in her line anymore and will not receive her new tweets either.

6) Read tweets: A user can read the tweets from a line by packs of 20 tweets. She can also refresh the tweets of a line to retrieve the tweets that have been posted since her last refresh.

III. Why Beernet?

Beernet is a transactional, scalable and elastic peer-to-peer key/value data store built on top of a Distributed Hash Table (DHT) [2][4]. Peers in Beernet are organized in a relaxed Chord-like ring [5] and keep O(log(N)) fingers for routing. This relaxed ring is more fault tolerant than a traditional ring and its robust join and leave algorithm to handle churn make Beernet a good candidate to build an elastic system. Any peer can perform lookup and store operations for any key in O(log(N)), where N is the number of peers in the network. The key distribution is done using a consistent hash function, roughly distributing the load among the peers. These two properties are a strong advantage for scalability of the system compared to solutions like client/server.

Beernet provides transactional storage with strong consistency, using different data abstractions. Fault-tolerance is achieved through symmetric replication, which has several advantages that we will not detail here compared to a leaf-set and successor list replication strategy [6]. In every transaction, a dynamically chosen transaction manager (TM) guarantees that if the transaction is committed, at least the majority of the replicas of an item stores the latest value of the item. A set of replicated TMs guarantees that the transaction does not rely on the survival of the TM leader. Transactions can involve several items. If the transaction is committed, all items are modified. Updates are performed using optimistic locking.

With respect to data abstractions, Beernet provides not only key/value-pairs as in Chord-like networks, but also key/value sets, as in OpenDHT-like networks [7]. The combination of these two abstractions provides more possibilities in order to design and build the database, as we will explain in Section V. Moreover, key/value sets are lock-free in Beernet, providing better performance. We opted for Beernet because of these native data abstractions. But any scalable and elastic key/value store providing transactional storage with strong consistency could be used as well.

IV. Running multiple services on Beernet

Multiple services using the same DHT can conflict with each other. We will now discuss two mechanisms designed to avoid those conflicts.

A. Protecting data with Secrets

Early in the process, we elicited a crucial requirement. The integrity of the data posted by the users on Bwitter must be preserved. A classical mechanism, but not without flaws, is to use a capability-based approach. Data is stored at random generated keys so that other applications and users using Beernet cannot erase others values because they do not know at which keys these values are stored. But in Bwitter, some information must be available for everybody and thus keys must be known by all users, meaning that we cannot use random keys. For example, any user must be able to retrieve the user profile of another user, it must thus know the key at which it is stored. The problem is that Beernet does not allow any form of authentication so key/value pairs are left unprotected, meaning that anybody able to make requests to Beernet can modify or delete any previously stored data.

We make a first and naive assumption that services running on Beernet are bug free and respectful of each other. They thus check at each write operation that nothing else is stored at a given key otherwise they cancel the operation. Thanks to the transactional support of Beernet the check and the write can be done atomically. This way we can avoid race conditions where process A reads, the process B reads, both concluding that there is nothing at a given key and both writing a value leading to the lost of one of the two writes.

This assumption is not realistic and adds complexity to the code of each application running on Beernet. We thus relax it and assume that Beernet is running in a safe environment like the cloud, which implies that no malicious node can be added to Beernet. We allow any application to make requests directly to any Beernet node from the Internet. We designed a mechanism called "secrets" to protect key/value pairs and key/value sets stored on Beernet enriching the existing Beernet API.

Applications can now associate secrets to key/value pairs and key/value sets they store. This secret is not mandatory, if no secret is provided a "public" secret is automatically added. This secret is needed to modify or delete what is stored at the key protected. For instance we could have the following situation. A first request stores at the key *bar* the

 ISBN: 978-1-61208-153-3

value *foo* using the secret ASecret, then another request tries to store at key *bar* another value using a secret different from ASecret. Because secrets are different Beernet rejects the last request, which will thus have no effect on the data store. A similar mechanism has been implemented for sets, allowing to dissociate the protection of the set as a whole and the values it contains.

Secrets are implemented in Beernet and have been tested through our Bwitter application. A similar but weaker mechanism is proposed by OpenDHT [7].

B. Dictionaries

At the moment in Beernet, as in all key/value stores we know, there is only one key space. This can cause problems if multiple services use the same key. For instance two services might design their database storing the user profiles at a key equal to the username of a user. This means they can not both have a user with the same username. This problem cannot be solved with the secrets mechanism we proposed. We thus propose to enhance the current Beernet API with multiple dictionaries. A dictionary has a unique name and refers to a key-space in Beernet. A new application can create a dictionary as it starts using Beernet. It can later create new dictionaries at run-time as needed, which allows the developpers to build more efficient and robust implementation. Dictionaries can be efficiently created on the fly in O(log(N)), where N is the number of peers in the Beernet network. Moreover dictionaries do not degrade storing and reading performance of Beernet. If two applications need to share data they just have to use the same dictionary. This has not yet been implemented, but API and algorithms are currently being designed. An open problem is how to avoid malicious applications to access the dictionary of another application.

V. Design Process

We will now present our design choices and explain how we prevent machines hosting popular values from overloading.

A. Main directions

We will start by discussing the main design choices we made for our implementation.

1) Make reads cheap: While designing the construction mechanism of the lines we were faced with the following choice: Either push the information and put the burden on the write, making the "post tweet" operation add a reference to the tweet in the lines of each follower. Or pulling the information and build the lines when a user wants to read them, by fetching all the tweets posted by the users he follows and reordering them. As people do more reads than writes on social networks, based on the assumption that each posted tweet is at least read one time, we opted to make reads cheaper than writes.

2) Do not store full tweets in the lines but references: There is no need to replicate the whole tweet inside each line, as a tweet could be potentially contain a lot of information and should be easy to delete. To delete a tweet the application only has to edit the stored tweet and does not need go through every line that could contain the tweet. When loading the tweet the application can see if it has been deleted or not.

3) Minimise the changes to an object: We want the objects to be as static as possible to enable cache systems. This is why we do not store potentially dynamic information inside the objects but rather have a pointer in them, pointing to a place where we could find the information. For instance, Tweets are only modified when we delete them, if there is a reply to them, the ID of the new child is stored in a separated set.

4) Do not make users load unnecessary things: Loading the whole line each time we want to see the new tweets would result in an unnecessarily high number of messages exchanged and would be highly bandwidth consuming. This is why we decided to cut lines, which in fact are just big sorted set, into subsets, which are sets of x tweets, that can be organised in a linked list fashion, where x is a tunable parameter. This way the user can load tweets in chunks of x tweets. The first subset contains all the references to the tweets posted since the last time the user retrieved the line, it can thus be much larger than x tweets, it is not a problem as users generally want to check all the new tweets when they consult a line. The cutting is then done as follows: the application removes the x oldest references from the first set, posts them in an new subset and repeats the operation until the loaded first set is smaller than x.

5) Retrieve tweets in order: Due to the cutting mechanism and delays in the network we can not be sure that each reference contained in a subset is strictly newer than the references stored in the next subset. So we also retrieve the tweet references from this one and only select the first 20 newest references before fetching the tweets.

6) Filter the references: When a user is dissociated from a line we do not want our application to still display the tweets he posted previously. We decided not to scan the whole line to remove all the references added by this user, but rather remove the user from the list of the users associated with the line and filter the references-based on this list before fetching the corresponding tweets.

7) Only encrypt sensitive data: Most of the data in Twitter is not private so there would be no point in encrypting it. Only the sensitive data such as the password of the users should be protected by encryption when stored.

8) Modularity: Even if our whole design and architecture relies on the features and API offered by Beernet it is always better to be modular and to define clear interfaces so we can replace a whole layer by an other easily. For instance any other DHT could easily be used, provided it supports the

 ISBN: 978-1-61208-153-3

same data abstractions or they can be simulated.

B. Improving overall performance by adding a cache

1) The popular value problem: Given the properties of the DHT, a key/value pair is mapped to a node or f nodes, where f is the replication factor, depending of the redundancy level desired. This implies that if a key is frequently requested, the nodes responsible for it can be overloaded while the rest of the network is mostly idle and adding additional machines is not going to improve the situation. It is not uncommon on Twitter to have wildly popular tweets that are retweeted by thousands of users. In the worst case the retweets can be seen as an exponential phenomenon as all the users following the retweeter are susceptible to retweet it too [8].

2) Use an application cache as solution: Adding nodes will not solve the problem, because the number of nodes responsible for a key/value pair will not change. In order to reduce this number of requests we have decided to add a cache with a LRU replacement strategy at the application level. This solves the retweet problem because now the application, which is in charge of several users, will have in its cache the tweet as soon as one of its user reads the popular tweet. This tweet will stay in the cache because the users frequently make requests to read it. This way we will reduce the load put on the nodes responsible for the tweet.

We now have to take into account that values are not immutable, they can be deleted and modified. A naive solution would be to do active pulling to Beernet to detect changes to the key/value pair stored in the cache. This would be quite inefficient as there are several values, like tweets, that almost never change. In order to avoid pulling we need a mechanism that warns us when a change is done to a key/value pair stored in the cache. Beernet, as described in [2], allows an application to register to a key/value pair and to receive a notification when this value is updated. Our application cache will thus register to each key/value pair that it actually holds and when it receives a notification from Beernet indicating that a pair has been updated it will update its corresponding replicas. This mechanism has the big advantage of removing unnecessary requests. Notifications are asynchronous, so the replicas in the cache can have different values at a given moment, leading to an eventual consistency model for the reads. On the other hand writes do not go through the cache but directly to Beernet, this allows to keep strong consistency for the writes inside Beernet. This is an acceptable trade off as we do not need strong consistency for reads inside a social network.

VI. Architecture

Bwitter is designed as a cloud application in which both the Beernet and Bwitter nodes run on a cloud infrastructure and the users are purely clients. We can thus easily add or remove Bwitter and Beernet nodes to meet the demand,

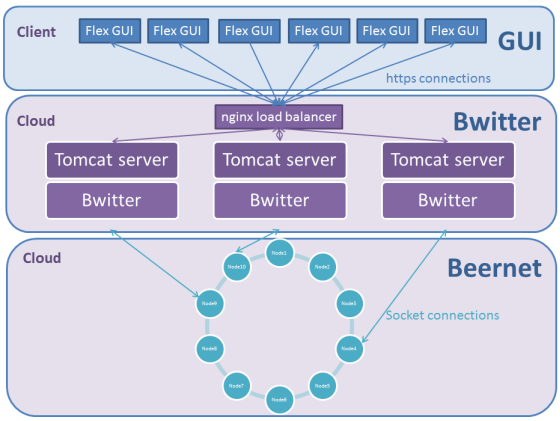


Figure 1. Architecture of the Bwitter social network application

increasing the efficiency of the network. A Bwitter node is a machine running Bwitter but generally also a Beernet node. This solution also allows us to keep a stable DHT as nodes are not subject to high churn as it was the case in the first architecture we presented. The Beernet layer is monitored in order to detect flash crowds and Beernet nodes are added and removed on the fly to meet the demand. We were not able to compare our system with the current Twitter architecure due to the lack of official documentation. But we know that Twitter is centralized, being able to handle only a limited number of concurrent request.

Our application consists of three loosely coupled layers. From top to bottom: the Graphic User Interface (GUI), the Bwitter layer which implements the operations described in Section II and finally the Beernet layer. The overall architecture is very modular and each layer can be changed assuming it respects the API of the layer above. The Beernet layer could be replaced by any key/value store with similar properties (in particular, with transactions and strong consistency). We recall that the data store must provide read/write operations on values and sets as well as implementing the secrets we described before.

The intermediate layer, also running on the cloud, is the core of Bwitter. It communicates both with Beernet and the GUIs. This layer can be put on the same machine as a Beernet node or on another machine. Normally there should be less Bwitter nodes than Beernet nodes. One Bwitter node is associated to a Beernet node but can be relinked to another Beernet node if it goes down. Each Bwitter node should be connected to a different Beernet node in order to share the load. In practice the Bwitter nodes are not accessible directly. They are accessed through a fast and transparent reverse proxy that splits the load between Bwitter nodes.

The top layer is the GUI, which runs on the client nodes and connects to a Bwitter node using a secure connection channel that guarantees the authenticity of the Bwitter node

 ISBN: 978-1-61208-153-3

and encrypts all the communications between the GUI and the Bwitter node. Multiple GUI modules can connect to the same Bwitter node.

A. Elasticity

We previously explained that to prevent the Fail Whale error, the system needs to *scale up* to allocate more resources to be able to answer an increase in user requests. Once the load of the system gets back to normal, the system needs to *scale down* to release unused resources. We briefly explain how a ring-based key/value store can handle elasticity in terms of data management.

1) Scale up: When a node j joins the ring in between peers i and k, it takes over part of the responsibility of its successor, more specifically all keys from i to j. Therefore, data migration is needed from peer k to peer j. The migration involves not only the data associated to keys in the range $]i, j]$, but also the replicated items symmetrically matching the range. Other NoSQL databases such as HBase [9] do not trigger any data migration upon adding new nodes to the system, showing better performance scaling up.

2) Scale down: There are two ways of removing nodes from the system: by *gently leaving* or by *failing*. It is very reasonable to consider gentle leaves in cloud environments, because the system explicitly decides to reduce the size of the system. In such case, it is assumed that the leaving peer j has time enough to migrate all its data to its successor, which becomes the new responsible for the key range $]i, j]$, being i the predecessor.

VII. Implementation

We implemented Bwitter using the cloud-based architecture of Figure 1. Source code is available at [10]. We made implementations both using Beernet [2] and Scalaris [11]. The architecture has three main layers: the GUI layer, the Bwitter layer, and the DHT layer. The GUI layer is implemented as a Rich Internet Application (RIA) using the Adobe Flex technology. The DHT layer is implemented using Beernet, built in Mozart v1.3.2 [12] enhanced with the secret mechanism. Beernet is accessible by the Bwitter layer through a socket API.

The Bwitter layer is connected to the DHT layer using sockets to communicate with an Oz agent controlling Beernet. The Bwitter layer is connected to the GUI layer with a Tomcat 7.0 application server using Java servlets from Java EE. The Bwitter nodes are accessible remotely via an http API that conforms to REST. The Tomcat servers are accessed indirectly through a reverse proxy server, in this case nginx. This nginx server is in charge of serving static content as well as doing load balancing for the Tomcat servers. This load balancing is performed so that messages of the same session are always mapped to the same Tomcat server. This is necessary as authentication is needed to perform some of the Bwitter operations and we did not want to share the state of the user sessions between the Bwitter nodes for performance reasons. The connection to the Web-based API is performed using https to meet the secure channel requirement of our architecture.

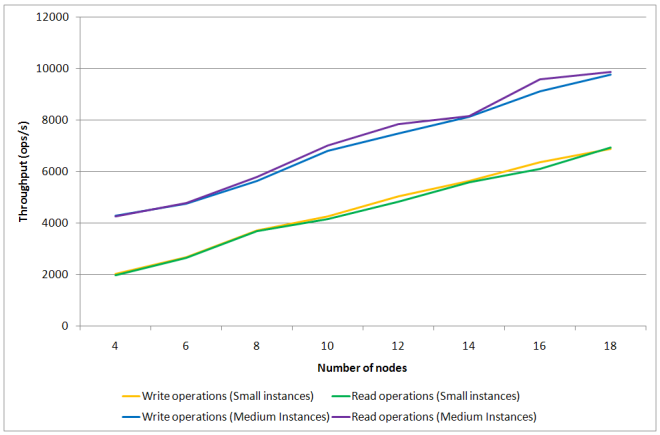


Figure 2. Scalability of the Scalaris transactional key/value store

VIII. Evaluation

We evaluated a prototype implemented with Scalaris v0.3 running on Amazon EC2 with up to 20 compute nodes. Note that we used Scalaris for the evaluation instead of Beernet, for technical reasons unrelated to Bwitter. This section summarizes our most important results; many more measurements and details can be found in [3]. Scalaris and Beernet both have very similar architecture and functionality: both provide a scalable transactional key/value store implemented on top of a replicated DHT and both use Paxos consensus for the transaction commit [2][11]. Since Scalaris underlies our Bwitter prototype (each Bwitter tweet requires many Scalaris operations), we first verified the performance and scalability of Scalaris. Figure 2 shows throughput for 20000 operations (reads or writes) as the number of compute nodes increases. This clearly shows that Scalaris is scalable for both reads and writes, on both Small and Medium size compute node instances in Amazon EC2.

For the Bwitter tests, we use one Large node for the dispatcher and many Small nodes for the Bwitter application. We simulated a network with two kinds of users, “Stars” and “Fans’, where Stars are followed by many Fans. We simulated two kinds of network: a Light network with 4000 users and 25 followers per user (each user follows 0.625% of the network) and a Heavy network with 2000 users and 50 followers per user (each user follows 2.5% of the network). Remark that both Light and Heavy networks have greater connectivity between users than the actual Twitter system, so that we can safely assume they are realistic loads.

Figure 3 shows aggregate throughput (number of successful operations per second) as a function of number of nodes. Here, an “operation” is defined in terms of what users do:

 ISBN: 978-1-61208-153-3

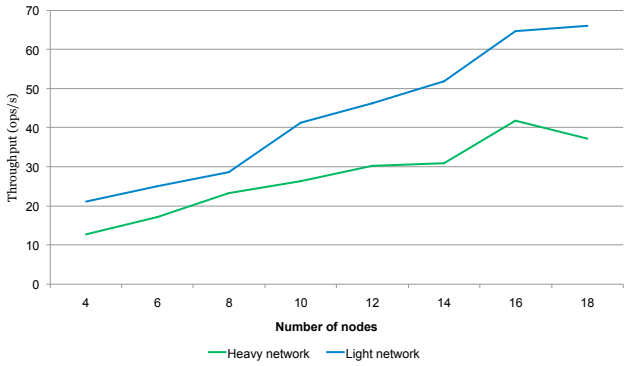


Figure 3. Scalability of the Bwitter application implemented with Scalaris

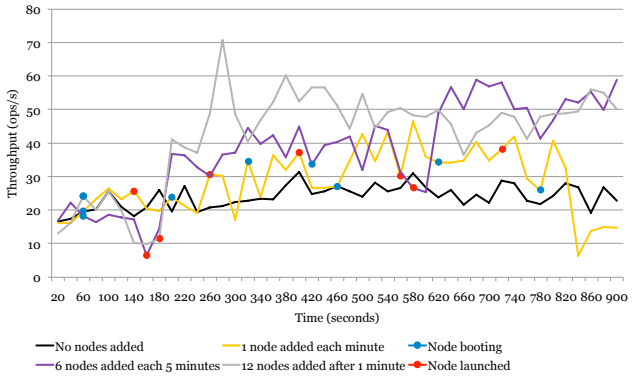


Figure 4. Elasticity of the Bwitter application implemented with Scalaris

it is either posting a tweet (20% of operations) or reading a set of recent tweets (reading all unread tweets counts as one operation; on average 20 tweets are read in one operation) (80% of operations). This means that Bwitter handles 66 operations/second with 18 nodes, which is slightly more than 1000 read/writes of individual tweets per second, in a network with 4000 users. Up to 18 nodes, the number of operations per second increases linearly with number of nodes for both Heavy and Light networks.

Figure 4 shows the elasticity behavior over a period of 15 minutes with four elasticity strategies, i.e., four ways of adding nodes to face increasing load. The black (lowest, almost horizontal) curve gives the baseline (no nodes added). The yellow (intermediate) curve shows the effect of adding one node every minute: the graph shows that this is not a good strategy. The best strategies are the gray and violet ones (highest throughput), in which larger numbers of nodes are added less frequently.

IX. Conclusion

The goal of Bwitter was to build a Twitter-like social network that is able to withstand flash crowds by using an elastic and scalable architecture. We used a scalable transactional key/value store, namely Beernet or Scalaris, as the data storage. We built an architecture on top of this store that is able to handle users with large numbers of followers and users following a large number of other users. We avoid overloading single nodes because we do not rely on any global keys and we use a cache to avoid the retweet problem. Scalability and elasticity tests performed on Amazon EC2 give encouraging results up to 18 nodes with realistic loads. During the implementation we came across two potentially important improvements for key/value stores, namely duplicating the key space using multiple dictionaries and protecting data via secrets (a form of capability). Secrets are now implemented in Beernet.

References

[1] Y. Lu, "What is Fail Whale?" www.whatisfailwhale.info, 2009.

[2] B. Mejías and P. Van Roy, "Beernet: Building self-managing decentralized systems with replicated transactional storage," *IJARAS: International Journal of Adaptive, Resilient, and Autonomic Systems*, vol. 1, no. 3, pp. 1–24, Jul.-Sep. 2010.

[3] X. De Coster and M. Ghilain, "Designing an Elastic and Scalable Social Network Application," pldc.info.ucl.ac.be, Programming Languages and Distributed Computing (PLDC) Research Group, Université catholique de Louvain, Tech. Rep., Aug. 2011.

[4] B. Mejías, "Beernet: pbeer-to-pbeer network, version 0.9," beernet.info.ucl.ac.be, 2011.

[5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, pp. 149–160.

[6] A. Ghodsi, "Distributed k-ary system: Algorithms for distributed hash tables," Ph.D. dissertation, KTH –- Royal Institute of Technology, Stockholm, Sweden, Dec. 2006.

[7] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A public DHT service and its uses," citeseer.ist.psu.edu/rhea05opendht.html, 2005.

[8] D. Boyd, S. Golder, and G. Lotan, "Tweet, tweet, retweet: Conversational aspects of retweeting on Twitter," in *Hawaii International Conference on System Sciences*, 2010, pp. 1–10.

[9] Apache, "HBase," hbase.apache.org, 2011.

[10] X. De Coster and M. Ghilain, "Bwitter source code," www.info.ucl.ac.be/~pvr/BwitterSources.zip, Aug. 2011.

[11] T. Schütt, F. Schintke, and A. Reinefeld, "Scalaris: reliable transactional p2p key/value store," in *ERLANG '08: Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*. New York, NY, USA: ACM, 2008, pp. 41–48.

[12] Mozart Consortium, "Mozart Programming System," www.mozart-oz.org, 2011.

 ISBN: 978-1-61208-153-3

A Social Network Approach to Provisioning and Management of Cloud Computing Services for Enterprises

Eric Kuada, Henning Olesen
Center for Communication, Media and Information Technologies
Aalborg University
Copenhagen, Denmark
kuada@cmi.aau.dk, olesen@cmi.aau.dk

***Abstract* - This paper proposes a social network approach to the provisioning and management of cloud computing services termed Opportunistic Cloud Computing Services (OCCS), for enterprises; and presents the research issues that need to be addressed for its implementation. We hypothesise that OCCS will facilitate the adoption process of cloud computing services by enterprises. OCCS deals with the concept of enterprises taking advantage of cloud computing services to meet their business needs without having to pay or paying a minimal fee for the services. The OCCS network will be modelled and implemented as a social network of enterprises collaborating strategically for the provisioning and consumption of cloud computing services without entering into any business agreements. We conclude that it is possible to configure current cloud service technologies and management tools for OCCS but there is a need for new approaches that view enterprises as both service providers and consumers to facilitate the easy implementation of OCCS networks**.

Keywords-cloud service brokerage; social networking; and opportunistic cloud computing services.

I. INTRODUCTION

Though faced with several challenges which are mostly security and risk management related, cloud computing adoption is gaining grounds with enterprises [1] because of the flexibility, scalability, elasticity, and potential cost savings that it offers to businesses [2]. With the support of industry analysts (e.g., Gartner, PricewaterhouseCoopers) and companies such as Amazon, Google, IBM, VMware, Microsoft, Sun, Dell, etc., this trend is not expected to change. Additionally, Vinod, et al. [3][4] suggest that instead of perceiving cloud computing simply as a way to make internal Information Technology services cheaper and efficient, businesses could take advantage of cloud computing to drive business growth by developing a new business model which is termed as the extensible enterprise.

The benefits of cloud computing has caught the attention of all stakeholders in research efforts to address its challenges to pave the way for an accelerated adoption of cloud computing services. There are therefore currently numerous research efforts by Information Technology industry giants, academic institutions, governments and union of countries (e.g., European Union) to promote the adoption of cloud computing services [5][6][7]. These efforts are resulting in diverse cloud computing service offerings from cloud service providers which have left enterprise consumers trying to make sense of the offerings of service providers. This situation is increasingly necessitating the services of a special group of cloud service providers that offer brokerage services for enterprise consumers on the more fundamental services such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) provided by cloud service providers.

This research work proposes a social network approach to the provisioning and management of cloud computing services termed Opportunistic Cloud Computing Service (OCCS) that has some resemblance to Cloud Service Brokerage (CSB). OCCS deals with the concept of enterprises taking advantage of cloud computing services to meet their business needs without having to pay or paying a minimal fee for the services.

This innovative approach of OCCS can facilitate the adoption process since enterprises will require no financial commitments to begin using cloud computing services, and discovery of services on an OCCS network will be easier in light of how information spreads on social networks. Commercial cloud service providers can benefit tremendously in the long run by introducing some of their services onto such a network; especially new services can be introduced onto the OCCS network for a period of time to gain popularity before being withdrawn later. Additionally OCCS can promote SaaS collaboration, scalability for resource aggregation for particular services when needed, fostering of business collaboration and further reduction of cost in Information Technology services. Since the idea of OCCS will be to provide a governance platform and its associated cloud management tools with which interested enterprises will provision SaaS, PaaS, IaaS and other resources that would be used by other interested enterprises, but not necessarily create new technologies, the platform is compatible with future cloud computing technologies and solutions.

The remainder of the paper is organised as followers: Section II explains the OCCS concept and outlines some of the background ideas and concepts that have inspired it. Section II also presents cloud service brokerage and outlines the similarities of OCCS in functionality with CSB. We present the research challenges that must be

 ISBN: 978-1-61208-153-3

addressed for the implementation of OCCS in Section III. Section IV discusses some unintended advantages that could be leveraged from OCCS implementation and Section V concludes the paper.

II. OPPORTUNISTIC CLOUD COMPUTING SERVICES

This section begins with an overview of the opportunistic cloud computing services concept, an outline of some background developments inspiring it, then a discussion of its Cloud Services Brokerage features and then presents detailed reference architecture for its implementation.

A. Overview

Opportunistic Cloud Computing Service (OCCS) is a social network approach to the provisioning and management of cloud computing services for enterprises. Previous works that link cloud computing with social networks such as [8], looked at leveraging the pre-established trust formed through friend relationships within social networking sites to enable friends to share resources; and most other examples use Cloud platforms to host social networks or create applications within the social network. There is however no literature on a social network infrastructure for enterprises currently; and this is where OCCS comes in. OCCS deals with the concept of enterprises taking advantage of cloud computing services to meet their business needs without having to pay or paying a minimal fee for the services. The OCCS network will form a social network of enterprises collaborating strategically (possibly selfishly or even maliciously) for the provisioning and consumption of cloud computing services without entering into any business agreements. Unlike social networking sites for individual use where users creates their own network of friends, in an OCCS network, members do not explicitly create ties with other members but these ties comes indirectly through the services and resource contribution and consumption mechanism.

This concept is derived from the combination of the concepts of peer-to-peer network services and the utility model of cloud computing. As in peer-to-peer networks where users are both resource providers and consumers, the idea will be to provide a governing platform that serves as the social networking platform for the enterprises and also consisting of interoperable Cloud management tools with which interested enterprises will provision SaaS, PaaS, IaaS and other resources that would be used by other enterprises interested in these services. A major challenge besides risk management and security issues that such a network will face is how to develop incentive schemes that ensure sustainability of the network.

It is anticipated that such a network will not always provide all the cloud service needs of an enterprise; hence OCCS will also seek to explore the utility model of cloud computing for enterprises to consume services provided by commercial cloud computing service providers at specific times, geographic locations, and Service Level Agreement (SLA) requirements for which a utility function defined by the enterprise is minimized. Here again the framework will try to employ open source brokerage tools instead of employing the services of a commercial Cloud Service Broker (CSB) for arbitrating between the cloud service providers and the enterprises.

Furthermore, preliminary investigations indicate that the OCCS network will not be most ideal for large corporation and financial institutions but will be well suited for small and medium sized enterprises. There have however been indications of larger corporations joining an OCCS network mainly as services and resource contributors in promoting their businesses.

Figure 1 shows an overview of the major parts in an OCCS network. It consists of two layers – the service layer and the management layer. The service layer consists of all the services contributed by members. These will normally be fundamental cloud services such as SaaS, PaaS, and IaaS; but, it can also include value added services normally provided by cloud service brokers. The management layer consists of two main components – the governance component that manages the services from members and CSB component that serves as an interface between the OCCS network and commercial cloud services providers and cloud service brokers.

OCCS is derived from two main concepts: peer-to-peer network services and the utility model of cloud computing. It however has also been inspired by equally important phenomenon such as social network theory, social networking, Web2.0, and the open source movement.

Social network theory has been used to examine how companies interact with each other, characterizing the many informal connections that link executives together, as well as associations and connections between individual employees at different companies. These networks provide ways for companies to gather information, deter competition, and even collude in setting prices or policies. It forms the basis of the OCCS feature of having no formal business agreements between the participating member enterprises. The other characteristics of OCCS stem from concepts and ideas such as user-generated content, harnessing the power of the crowd, architecture of participation, data on a epic scale, and openness [9] that characterises Web 2.0, social networking and the open source movement. OCCS however focus on corporate organisations instead of individual users and deals with replacing simple data and files as resources with cloud computing services that would normally have been provided by commercial cloud service providers.

 ISBN: 978-1-61208-153-3

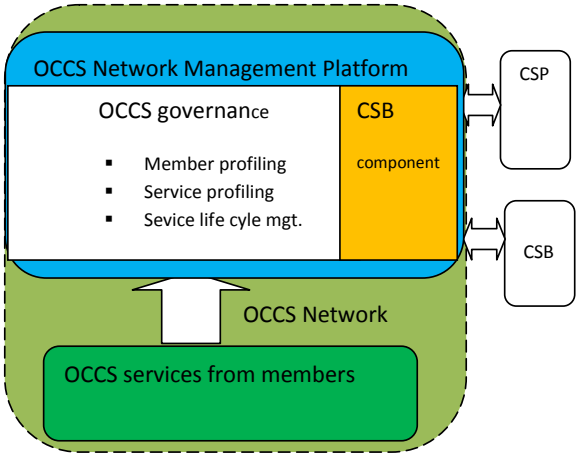


Figure 1. Major components of an OCCS network

B. Cloud Service Brokerage Functionalities

Cloud services brokerage is a business model where a company or other entity adds value to one or more (generally public or hybrid, but possibly private) cloud services on behalf of one or more consumers of those services [10]. The major functionalities that CSB provide to enterprises include service aggregation, customization, governance, new applications, services billing and arbitration, security, and insurance services. The services of CSB are becoming increasingly necessary to both enterprises and cloud service providers because of their different perspectives, objectives, and expectations from the cloud computing industry, coupled with the challenges enterprises have to deal with in selecting from cloud service providers and using the diverse cloud computing services.

An OCCS network consists of two main components - a platform for managing the services provisioned by members and a brokerage component for interfacing with commercial cloud service providers. The OCCS concept thus inherently provides new applications, service aggregation to multiple consumers, governance, and service arbitration and billing.

C. OCCS Architecture

In light with the principles on which the OCCS concept is built – namely: user-generated content, architecture of participation and openness; a successful implementation of an OCCS network will have to in the barest minimum provide the following features

- Support for the management of fundamental cloud computing services (SaaS, PaaS, IaaS)
- Support for the management of any arbitrary cloud computing service – anything as a service (XaaS)
- Interoperability with major cloud computing standards
- Interoperability with major cloud computing management tools
- Support for future cloud computing technologies

These factors have been considered in the design of the OCCS network reference architecture shown in Figure 2.

OCCS Services: these consist of all the services and resources that have been contributed to the network by members. These could be coming from contributing member's data center, private cloud, etc. Services are mainly fundamental cloud computing services such as SaaS, PaaS, IaaS; and other cloud computing services (XaaS) and resources.

Resource Manager: this together with the cloud computing deployment and management tools found in the Contributions Component and the Discovery & Utilization Component abstract the contributed services from the services layer and interface it to the OCCS management platform.

Contributions Component: it is responsible for handling the resource contribution process. Its main objective is to simplify and make it easy for members to contribute resources to the network. It performs two sub functions – providing cloud computing management tools and service life cycle management. It thus consists of cloud computing deployment and management tools for all types of services and resources. The service management involves service creation, service certification and service profiling which includes service review and ranking by users and service ranking by platform administrators.

Discovery & Utilization Component: its role is to simplify services and resources discovery and utilization process. It performs service recommendation by taking service requirements description by members and matching these with service properties description by contributors together with the profile rank of services. It also consists of cloud computing management tools for services and resources provisioning and utilization.

Categorization Component: this component is needed to ensure OCCS network supports arbitrary services while also ensuring easy management of these services. It is responsible for the categories creation process. It handles service category creation requests from members which is evaluated for approval by the platform administrators; and

 ISBN: 978-1-61208-153-3

also delegates privileges of categories creation given to some level of membership.

***Membership Manager**:* this is the main social network user management module for the OCCS network management platform. It is responsible for managing existing users and the registration of new members (enterprises, companies, institutions, etc.). It handles membership requests and in cooperation with the Governance Component performs company profile verification based on data provided by enterprises during registration to make decisions on membership approval or rejection.

Incentives Manager: Dynamic re-computation of cost in real time to be credited to service contributors and debited to resource users. Cost of service or resource utilization is dependent on demand.

QoS & Pseudo SLA Manager: it uses information from the Incentives Manager to provide service differentiation and pseudo SLA management to members.

Governance Component: it is the logical module that provides supervision for all the other components in the OCCS network management platform. It is implemented as the interfaces through which platform administrators interact with the platform to make governance decision.

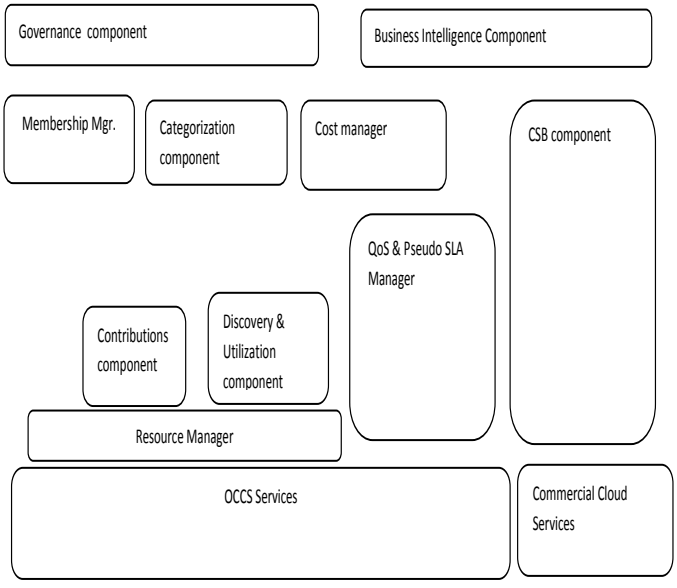


Figure 2. OCCS network Reference Architecture

Business Intelligence Component: this module is not essentially required for the operation of the OCCS network but provides means for the gathering of business intelligence from the platform and may include for example:

- Analysis of services contributed, their categories, utilization and their profile performance
- Analysis of member profiles with their contributed services and the enterprises that are utilizing these services
- Analysis of the services and resources requests that are not currently being provided by the platform

***CSB Component**:* it consists of cloud computing management tools and processes that interface the OCCS network to commercial cloud computing services and provide cloud brokerage services to members.

D. Implementation Strategy for OCCS

To ensure that the barest minimum features required for a successful implementation of a OCCS network is met, a typical OCCS network implementation will use the feature requirements of support for the management of fundamental cloud computing services, support for the management of any arbitrary cloud computing service, interoperability with major cloud computing standards and cloud computing management tools, and support for future cloud management technologies, in selecting a suitable cloud management tool (likely a non proprietary cloud management tool) which will form the base on which other functionalities can be added. The various components outlined in the OCCS reference architecture in Section III *C* above can then be developed on this base cloud management tool.

III. RESEARCH ISSUES WITH OCCS

Some of the major challenges of cloud computing receiving research attention currently include legal and compliance risk management, migration of applications, meeting SLA requirements, managing cloud services, and security concerns. The introduction of OCCS brings new research issues and adds a complexity dimension to some of the existing ones. This section outlines some of these research issues and the intuitive approaches of addressing them, which will have to be researched carefully for the successful implementation of OCCS networks.

A. Sustainability and Pseudo SLA

The sustainability of an OCCS network revolves around the concepts of architecture of participation and harnessing the power of the crowd. A potential problem that such a network will face is that of free-riding where member enterprises will want to only use services on the network without contributing [10]. The challenge here will be to develop appropriate incentive mechanisms for the sustainable operation of the network.

Another challenge is that of service differentiation and service quality management. Unlike conventional cloud computing service offerings by commercial service

 ISBN: 978-1-61208-153-3

providers, no SLA exist between the participating members in an OCCS network, hence such service quality differentiation must be handled through the incentive mechanisms that will be designed so that when limited resources are being contended for by multiple candidates those that have supported the system more can be given some form of preference. Additionally, there will be the need for transparency in dynamic demands and cost of service utilisation. Several research efforts have applied game theoretic approach to the modelling of incentives in peer-to-peer networks to solve the free-riding problem in peer-to-peer networks. [12] presents a resource allocation mechanism based on a distributed algorithm to enable service differentiation in peer-to-peer networks that also increases the aggregate utility in the whole network. Work on incentives for sharing in peer-to-peer networks by [13] analyzes several different payment mechanisms designed to encourage file sharing in peer-to-peer systems. The game theoretic approach can be explored in the design of incentive mechanisms for OCCS networks and the concept of pseudo SLA introduced for service differentiation and service quality management.

B. Reliability and Fault resilience

An OCCS network will need to provide a certain level of reliability to its members under normal operations and must be resilient enough to recover from faults. The reliability and resilience is however threatened by poor quality of services provisioned by members, failure and withdrawal of services from members, and the introduction of malicious services. Dynamic algorithms are required for detection, notification and responding to faults and poor quality services. Of particular importance is how to respond to faults in the network. A simple approach will be to notify service consumers of problematic events for them to take their own decisions; it may however be necessary to develop mechanisms that reassign alternative services to consumers based on certain usage policies and preferences indicated by the service consumers. The challenge here is the precise capturing of the properties of services in service descriptors and effectively matching these to the usage policies and SLA requirements of potential service consumers so that the entire process is transparent to them and their customers; and more so this transparency in fault handling must be achieved in the context of the fact that no SLA exists between the contributors of the services and the consumers of these services.

C. Network Governance

The purpose of the OCCS network governance will be to promote the overall quality of the system. Of particular research interest is the development of community management enabling technologies for profiling, service life cycle management and transparency in the pseudo SLA management. Both network members (enterprises) and the services they provision will have to be profiled to maintain trust in the individual services, member enterprises and the entire system platform. For example service provisioning will have to be in phases such as testing, and various levels of certification through continual ranking of services. Both central ranking by the platform administrators and peer review ranking by the members may have to be adopted. The service ranking and certification will need to promote new services from good profiled enterprises while quickly identifying malicious and poor quality services and revoking their certification.

D. Security

Security is the ability to protect information and information systems from unauthorized access, use, disclosure, disruption, modification or destruction and to respond and recover in case of a fault or incident. The implementation of OCCS will not bring any new technical demands on security in terms of confidentiality and data integrity apart from what is already necessary in ordinary cloud computing implementations. An area of research interest however is how to harness the available resources on the platform and the collaboration of members in combating security threats. If we consider the introduction of malicious services onto the OCCS platform, the OCCS network governance which includes member profiling, service profiling and life cycle management should prevent such occurrences. In the event of such an occurrence however, the system has to respond and recover quickly. It is therefore useful to research into mechanisms for harnessing the available resources on the platform and the collaborative efforts of members in dealing with such a threat.

E. Other Research Issues

Some other issues that are of importance and worth looking at are regulations and service provisioning. Current cloud computing vendor technologies and management tools assume distinct roles for the service providers and service consumers. But with some cloud management tools offering features such as delegated control and autonomous virtual enterprises [14]; and support for the technologies of most of the major cloud solution providers [15], it will be possible to configure these cloud management tools for OCCS. There may however be a need for new approaches for cloud management that view enterprises as both resource providers and consumers to facilitate the easy implementation of an OCCS network.

An issue with regulatory authorities for enterprises joining the OCCS platform could be that of tax evasion implications. This is because enterprises will be offering and using services, which are not being paid for and hence may not be subject to taxes depending on the country in which they are. Also most enterprises have internal policies that need adherence, and there may be industry specific laws and regulations that they need to comply with. Furthermore, different countries have their own laws concerning user data handling. Storing data in the Cloud therefore presents enterprises and service

 ISBN: 978-1-61208-153-3

providers with several risk management challenges. These challenges are further compounded by the concept of OCCS and hence can hamper its successful implementation.

IV. POSSIBLE FUTURE BENEFITS

This section gives brief discussions on some of the unintended benefits that can be leveraged from the implementation of an OCCS network. Some of benefits as discussed below include platform for new business models, promotion of SaaS collaborations, and promotion of cloud computing standardization.

A. Platform for new Business Models

OCCS can serve as a platform for enterprises to adopt new business models such as the extensible-enterprise model (deep B2B integration and highly modular web services). The adoption of cloud computing by any two companies in general reduces the complexities in business-to-business (B2B) integration. Companies can therefore leverage cloud computing by exposing their business processes to potentially large ecosystems of partners who often find ways of joining and integrating their business processes in the value chain. It is envisaged that OCCS will promote the adoption of cloud computing by enterprises and hence indirectly promoting such new business models. Secondly, enterprises on an OCCS network would already have been using similar services with similar cloud management tools; this should facilitate the integration of their business processes.

Additionally, the platform can foster the creation of new business that will provide commercial cloud brokerage services to members on the OCCS network.

B. Promotion of SaaS collaborations

The implementation of an OCCS network can promote SaaS collaborations. Enterprises on an OCCS network are very likely to participate in collaboration efforts in the development of software solutions that they deem useful to their own business. As an example, a construction company in need of a specialized software for design simulation that is currently not being provided by any member on an OCCS platform can initiate a SaaS project to involve other interested members in the development of the software which can then be contributed to the platform upon completion. Such SaaS collaborations could also come about by a member enterprise identifying an application of interest and providing the development platform with specific tools and providing it as a PaaS on the OCCS network; this could spark interest in the development of such an application by other members and can eventually lead to collaboration by interested members in its development.

C. Promotion of Cloud Computing Standardization

As already indicated in Section II C and Section II D, a successful implementation of an OCCS network must provide support for the management of fundamental cloud computing services, support for the management of any arbitrary cloud computing service, interoperability with major cloud computing standards and cloud computing management tools, and support for future cloud management technologies. Thus to start with, the OCCS concept must carefully follow cloud computing standards; the situation is however reversed if OCCS network implementations become successful. Thus those standards that are dominant on the OCCS platform will then be followed closely by cloud management tool developers and cloud service providers. This will further promote the success of the OCCS platform; and hence the promotion of cloud computing standardization and promotion of the OCCS implementations will be in a virtuous cycle.

V. CONCLUSION

Support for major hypervisors and role-based delegated control make it possible to configure current cloud computing technologies and management tools for OCCS even though they assume distinct roles for the service providers and service consumers. There is however a need for new approaches to cloud management that view enterprises as both resource providers and consumers which when complemented with standards for interoperability will facilitate the easy implementation of an OCCS network.

Successful implementation of OCCS networks can result is some unintended benefits such as serving as a platform for new business models, promotion of SaaS collaborations, and promotion of cloud computing standardization. These benefits together with providing a platform for enterprises to start using cloud computing services without any initial financial commitment will however be possible only if the research challenges identified in Section III (namely, developing appropriate incentive mechanisms and the associated quality of service differentiation, security, reliability and fault resilience, network governance and regulatory issues) are carefully dealt with.

REFERENCES

[1] Justin Pirie, "Setting the Standards," *European Communications*, pp. 30-31, Autumn 2010.

[2] Vinod Baya and Randy Myers, "How CFOs should audit the cloud balance sheet," *PricewaterhouseCoopers Technology Forecast*, no. 4, pp. 44-53, 2010.

[3] Vinod Baya and Galen Gruman, "Making the Extensible Enterprise a reality," *PricewaterhouseCoopers Technology Forecast*, no. 4, pp. 26-35, 2010.

 ISBN: 978-1-61208-153-3

[4] Vinod Baya, Bud Mathaisel, and Bo Parker, "The cloud you don't know: An engine for new business growth," *PricewaterhouseCoopers Technology Forecast*, no. 4, pp. 4-13, 2010.

[5] OPTIMIS: Optimized Infrastructure Services. (2011, March) OPTIMIS. [Online]. http://www.optimis-project.eu/project 11-07-2011

[6] Contrail consortium 2010. (2011, March) Contrail. [Online]. http://http://contrail-project.eu/objectives;jsessionid=E1961F3D98F3B602D34CFC9445D63DC7 11-07-2011

[7] RESERVOIR. (2011, March) RESERVOIR. [Online]. http://www.reservoir-fp7.eu/ 11-07-2011

[8] Kyle Chard, Simon Caton, Omer Rana, Kris Bubendorfer, "Social Cloud: Cloud Computing in Social Networks," in *2010 IEEE 3rd International Conference on Cloud Computing*, Miami, Florida, 2010, pp. 99 - 106.

[9] Paul Andeson, "What is Web 2.0? Ideas, technologies and implications for education," JISC , JISC Technology and Standards Watch Feb. 2007.

[10] Benoit Lheureux. (2010, December) Gartner Inc. [Online]. http://www.gartner.com/it/content/1461800/1461813/december_2_cloud_services_brokerage_blheureux.pdf 12-07-2011

[11] Markus Hofmann and Leland R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, Rick Adams and Karyn Johnson, Eds. San Francisco, USA: Elsevier, 2005.

[12] Richard T. B. Ma, et al, "A Game Theoretic Approach to Provide Incentive and Service Differenciation in Peer-to-Peer Networks," in *SIGMETRICS/Performance'04*, New York, NY, USA., June 12–16, 2004.

[13] Philippe Golle, Kevin Leyton-Brown, Ilya Mironov, and Mark Lillibridge, "Incentives for Sharing in Peer-to-Peer Networks," in *Springer WELCOM 2001, LNCS 2232*, Verlag Berlin Heidelberg, 2001, pp. 75-87.

[14] Abiquo, Inc. (2011, March) abiquo. [Online]. http://www.abiquo.com/products/features-and-benefits.php?lang=en 11-07-2011

[15] enStratus Networks LLC. (2011, April) enStratus. [Online]. http://www.enstratus.com/page/1/cloud-providers.jsp 11-07-2011

 ISBN: 978-1-61208-153-3

Competitive P2P Scheduling of Users' Jobs in Cloud

Beniamino Di Martino, Rocco Aversa, Salvatore Venticinque, Luigi Buonanno
Department of Information Engineering
S.U.N. (Seconda Università di Napoli)
Aversa (Italy)
[Beniamino.DiMartino,Rocco.Aversa,Salvatore.Venticinque,Luigi.Buonanno]@unina2.it

***Abstract*— Existing distributed solutions for distributed computing (Grid, Cloud, etc.) pose a high threshold for potential customers. The reason deals with the technical background and effort that are usually required in order to successfully access the computing facilities, thus limiting their massive adoption. By exploiting the features offered by different distributed paradigms (P2P and Cloud), we propose here an approach that reverses the role of resource requestors and resource providers, allowing potential customers to access the distributed infrastructures in a user-friendly fashion. In the proposed scenario, the task of retrieving the user's submitted jobs and configure accordingly the necessary resources is in charge of the providers, thus lowering the threshold required to successfully exploit the computing facilities. The experimental activities, described in the paper, validate the hypothesis that a competitive approach, in distributed scheduling environments, can decrease the threshold required to access the facilities and lead, if properly set up, to substantial performance gains.**

***Keywords*- P2P; cloud; competitive scheduling.**

I. INTRODUCTION

Cloud computing [26] is a recent model for enabling 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. On one hand, thanks to the virtualization technology, providers can rent their hardware resources in a very flexible way. On the other hand, users may have a dedicated data center as a service without the burden of buying and managing expensive hardware, but rather paying their utilization according to a pay-per-use business model.

Despite the benefits provided, many open issues have to be addressed with regard to this emerging computing paradigm. Some of them are portability of applications, lock-in proprietary solutions, negotiation and check of SLAs (Service Level Agreement) with Cloud providers. Among the others, an open issue has affected most of the distributed paradigms which have been spreading for the last few years: existing solutions require the customer to hold an advanced technical background in order to successfully exploit the computing facilities, thus limiting their massive adoption.

The list of issues a potential user has to deal with includes: the discovery of the architecture that is compliant with the application requirements, the setup of the execution environment, the research of the most convenient offer, the configuration of the acquired resources, the tuning of the applications, the uncertainty of execution time due to a best effort policy for resource sharing, etc.

Other distributed paradigms (e.g., inverted client-server systems [3]) do not pose such a high threshold to potential customers, but they do not encourage the intensive exploitation of resources.

P2P (Peer-to-Peer) [23] refers to logical organization of computing entities where each individual knows its neighbors and can behave both as a server and a client. There are some relevant examples of P2P systems, oriented to parallel and/or distributed computing, which have been successful in their exploitation.

In order to address the described issues, we propose a distributed paradigm that:

- Aims at implementing the same ease of use of P2P file sharing applications.
- Reverses the roles of requestors and providers, by charging the providers of all the overhead required to setup the execution environment, manage the job requirements, etc. In our model, clients just publish their jobs on the platform, specifying the software and hardware requirements, the application details, the deadline and the offered reward. Service providers, on the other hand, are in charge of discovering the published jobs and of addressing all the issues related to the jobs' requirements management;
- Adopts a competitive approach, where providers compete for satisfying the client's requests and are awarded with credits in case of successful elaborations, thus optimizing client's satisfaction and reducing the cost.

In the next section, we discuss related work. The third section introduces a comparison of policies for resource sharing in centralized and P2P networks. In the fourth

 ISBN: 978-1-61208-153-3

section, we present our competitive approach for job scheduling in P2P. In the fifth section, we provide a description of a prototype implementation and we show experimental results aimed at evaluating the effectiveness of the proposed solution. Finally, we present the conclusion.

II. RELATED WORK

Cloud computing is an on-demand distributed paradigm that refers to providers offering a large pool of easily usable and accessible virtualized resources in a pay-per-use model [1]. The services can be delivered within different layers, that are usually classified as SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) [24]. This paper mainly refers to SaaS and IaaS clouds. Cloud computing allows data centers to transparently offer services through the Internet by exploiting their computing and storage fabric of resources. In SaaS and IaaS clouds, applications and nodes are virtualized and dynamically provisioned on-demand as a personalized resource collection to meet a specific service-level agreement, which is established through a negotiation. A market-oriented resource management is necessary to regulate the supply and demand of Cloud resources [9], providing feedback in terms of economic incentives for both Cloud consumers and providers, and promoting QoS-based (Quality of Service) resource allocation mechanisms that differentiate service requests based on their utility [2]. Many research contributions aim at supporting the user with negotiation services based on Service Level Agreement that delegate to agents the discovery and agreement of the best offer from multiple providers [7], [11]. The main aim of this paper is to bring this mechanism a step further, by delegating this task to providers.

Security is still a big concern in cloud frameworks. While in Grid [25] environments, indeed, both resources and users need to be registered and to get a digital certificate for authentication and authorization purposes, before they are allowed to start a session. This mechanism is feasible when the number of nodes are not many. Security in cloud computing infrastructures, instead, is still, mostly, a work in progress. Nevertheless, some analysis on this topic have been performed [17]. Furthermore, it must be said that the very subject of security is what is slowing down the adoption of cloud computing over other forms of distributed scheduling [18].

Current P2P systems have the perk of allowing a very high number of users (hundreds of thousands is a common figure). They offer few services, without doing assumptions on the reliability of the peers themselves [14]. However, it is very complicated to ensure a given QoS level [13] without any sort of distributed scheduling. From a security point of view, P2P systems are, by definition, environments where it is difficult to be aware of the identity and trustability of hosts: the chance of exploiting a malicious resource is intrinsically high. While this risk is largely accepted for file sharing systems, in order to make it acceptable for distributed computing many issues must be addressed to ensure the safety of both the code owner and the code executor.

In [11], an architecture for the resource sharing on large scale networks has been described (CompuP2P). CompuP2P uses a protocol based on Chord [16] and detects a set of ”dynamic markets”, each of them groups all the peers that are willing to buy or sell the same ”amount” of computing power. The main bottleneck is represented by a special peer (”Market Owner), that is responsible for the association between requests and offers of computing power. In [15], a solution for the scheduling of multiple applications, in a concurrent fashion, is proposed. Authors propose a decentralized scheduling pattern and do a comparative analysis of different heuristic logics. Many Grid solutions for task scheduling and workload distribution exist. For example, Condor [12] is a high-throughput distributed batch computing system that provides job management mechanisms, scheduling policies, resource monitoring, and resource management. However, it can hardly be defined as a P2P system, cause of the presence of a central manager that accepts job submissions. Conversely, the objective of our research is to design a P2P infrastructure that is not relying on any centralized element and that enables a huge numbers of machines, which connect/disconnect dynamically to the network without any guaranties on their reliability, to easily access the resources offered by cloud providers.

III. CRITICAL COMPARISON OF DIFFERENT APPROACHES FOR RESOURCES SHARING

In the computational grids model, providers offer their services with a best effort policy and a collaboration pattern is usually adopted among different parties, which share their resources belonging to a virtual organization, in order to optimize the global performances. Grid clients compete to use the resources: this model exploits the competition of clients and the collaboration of servers.

According to a common opinion, the business Grid model was unsuccessful because providers are business competitors and, usually, do not collaborate. However, even if theoretically the market should rely on the competition of providers, often, in real-world scenarios, sellers cooperate rather than competing while, at the same time, trying to create competition among buyers.

It is a model similar to the one that is currently adopted in the automotive, where different companies share engines and other components, or in the insurance field, where prices are fixed above a threshold using a behavior that is, at least, at the edge of the law. Great companies have much interests and resources to organize themselves for collaborating. Even if powerful ones should give they usually take, by choosing to collaborate, rather than fighting, when it means a bigger return. Collaboration of providers is exploited to take.

In volunteer computing, clients are asked to donate CPU cycles when their computers are idle.

Users’ resources are then managed and exploited by powerful big organizations. In fact, they have the capacity to exploit all the limited resources shared by a huge number of distributed users. The lack of this kind of organization ability, and, at the same time, the great capability of users in terms of availability is evident in real life and in distributed computing. In volunteer computing, collaboration among

 ISBN: 978-1-61208-153-3

users is exploited *to give*. The most well-known case of volunteer computing is the SETI@Home project [3].

P2P is a successful example of decentralized resource sharing among clients. In P2P file sharing systems, users compete to download files from the available sources and are asked to share their data ("collaborate") in change of credits that can be spent for acquiring download privileges. Competition is easier to be implemented, because organization is not required. Competition among users is exploited *to take*, while the collaboration is used *to give.*

Cloud computing is a new paradigm that was born in a business context. The business model is pay per use and it is not based on resource sharing. A limitation of this approach is that, if a user chooses a solution from a particular provider, he will be locked by that choice because of the lack of portability.

In this scenario, it would be useful to design a technological solution that implements a business model aimed at optimizing the QoS at user side, and to maximize the incomes at server side.

A free market model, that exploits the competition of sellers to give computing power and collaboration of users to take it, could be the best solution.

Our approach proposes the utilization of a P2P model that allows the users to collaborate by publishing their jobs as "calls for proposal" (cfp), in the same way as it occurs in file sharing systems. On the other hand, the business model is based on the competition among servers, which seek shared proposals and try to answer as soon as possible in order to obtain the offered reward.

IV. P2P COMPETITIVE SCHEDULING OF USERS' JOBS

In this section, we propose a competitive approach for P2P distributed computing, whereas the roles of involved parties are inverted if compared to the Grid, Cloud computing or web services models: clients publish jobs on a P2P network overlay ("call for execution") while the servers look for these and compete to deliver the results. Calls for execution describe the requirements of the application, the credits the user would pay and, optionally, a deadline before which the results should be available.

While in the Grid model, and in traditional architectures for distributed scheduling, the job owner is in charge to choose the execution node, to check its compliance with the application requirements and to ask for the execution, in the proposed model these issues must be managed at server side. We think that the proposed approach would be very effective in the Cloud market, where providers can set up virtual, specialized environments for the execution of different jobs and use the idle ones to satisfy the user's request. Virtualization is commonly used by Cloud providers to improve the throughput of their hardware resources: thanks to the modern Cloud computing paradigms, the configuration of the task execution environment can be easily adapted to match the application requirements by exploiting the virtualization technology.

In our model, providers can exploit at the best their resources, and the Cloud IAAS, by managing both their overbooking and their smart scheduling. We try to design our model as much similarly as possible to current P2P systems for file sharing whose success in the Internet community has been bigger than the Grid.

It is evident, according to what has been discussed in the previous paragraphs, that many issues must be addressed in order to consider the P2P model as a viable relay for distributed computing at business level. This topic is out of the scope of this paper.

In the model, two kind of peers are defined: buyer and seller peers. User peers publish application descriptors, in the same way a file is commonly shared in P2P file sharing systems. The descriptor includes all the hardware and software requirements, as well as other constraints like the time within the results must be delivered and the offered reward. Clearly, it includes the info required for retrieving the task code and data. Seller peers crawl the network looking for published jobs, analyze the constraints, and choose to accept the proposals according to their own policy. For each retrieved request, the seller peer is able to evaluate its ability to fulfill the requirements and its convenience to accept the task. Multiple seller peers can accept the same task, and different patterns can be defined, e.g. the buyer could state that only the first business peer that delivers the results will be awarded, so that the seller peers will have to compete for being be the first one that completes the job. However, this is not the only possible pattern (e.g. in SETI@Home, multiple peers execute the same tasks and results are matched against each other).

Our model allows asynchronous mechanisms to be adopted: the user peer who published its job can disconnect, being aware that the results will be delivered according to what is specified in the job descriptor. This approach could be effective within today business scenario, where multiple providers exist and compete to promote their own services.. Furthermore, it allows for some flexibility (e.g. sellers could act as brokers that use resources provided by commercial Clouds providers).

Some keystones of the approach are:

- Client peers publish "calls for execution";
- Server peers discover and download calls for execution. Furthermore, they retrieve the code to be executed and the data;
- Server peers compete to complete as many as possible jobs to maximize their incomes;
- Clients can disconnect at any time: computation continues at server side;
- Workload balancing can be implemented.
- A business model is required to promote the execution of one's own applications.
- It is effective in an industrial environment.
- Configurations of virtual machines, or general computing resources, are set up according to the application's specific requirements.

 ISBN: 978-1-61208-153-3

V. IMPLEMENTATION

A prototype implementation of the above described model has been developed. In our implementation, the client and business peers are named, respectively, Buyers and Sellers, to highlight the market-like modeling of the system.

We have extended jKad [20], a publicly available open source implementation of the Kademlia [19] protocol released under GNU Lesser GPL. Each actor is composed by a set of different modules, each one performing a specific task. Figure 1 shows the modular architecture we have implemented.

The Buyer is composed of three modules:

- The *P2P GUI*, that implements a Graphical interface that allows the user to define the job properties according to the ontology.
- The *Job Sharing* is responsible for publishing the jobs submitted by the user into the P2P overlay.
- A *Network* module, that interacts with the Sellers, exchanging data files and results.

The Seller is composed of:

- A *Job Discovery* module, that is in charge of crawling the network in order to discover available CFEs (Call for Execution).
- A Parser, that analyzes the retrieved jobs. Furthermore, it interacts with the Buyer's Network module to retrieve the data required for the job's execution.
- A Job Queue Manager, that sequentially schedule the jobs.
- A Result manager, responsible for interacting with the Buyer's Network module in order to return the results.

A. P2P technology

The underlay system is a Kademlia-like P2P network. It has been chosen because of the major properties that DHT-based (Distributed Hash Tables) [27] P2P systems bring to applications (predictability of key research, robustness against node failures, etc.) and because of its simple protocol. In fact, only four messages are defined by the protocol:

- PING (node): to verify if a peer is still alive.
- STORE (key,value): to store a (key,value) pair in one or more nodes of the network.
- FIND NODE (node): to retrieve the k nodes that are closest (according to a XOR metric) to the node used as parameter.
- FIND VALUE (key): a node receiving this message returns the corresponding value if it has the requested key in his store. Otherwise, it will behave as upon receiving a FIND NODE.

As described in Figure 2, Buyers can publish jobs at any moment. Sellers look for shared CFEs and choose, for each job, whatever it is convenient or not to accept it. Once the job's descriptor is downloaded, the Seller can start the job execution. Results are then returned to the Buyer; consequently, the Seller can get its reward if the results and the timing are compliant to the job requirements.

As already mentioned in the previous paragraphs, multiple patterns are possible: the reward could be awarded to the first Seller who executes the task, or to all those able do deliver the results before a given deadline, or to the first "*n*", etc. The analysis of this topic is beyond the scope of this paper.

B. Prototype description

The P2P GUI module allows users to specify the job requirements and to publish CFEs. The input form (Figure 3) is dynamically drawn by the application according to an OWL template. A hash of the data entered by the user is then calculated and published into the P2P overlay network.

Publishing is performed by the Job Sharing module and it is implemented as a simple STORE message on the Kademlia network, using a special label as a key that identifies the shared job descriptor.

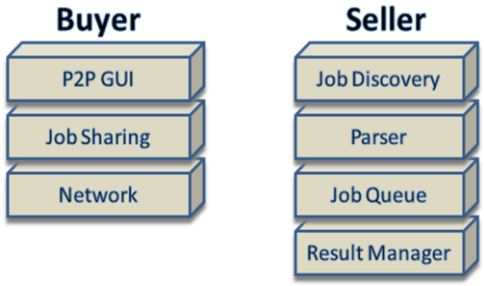


Figure 1. The architectural model

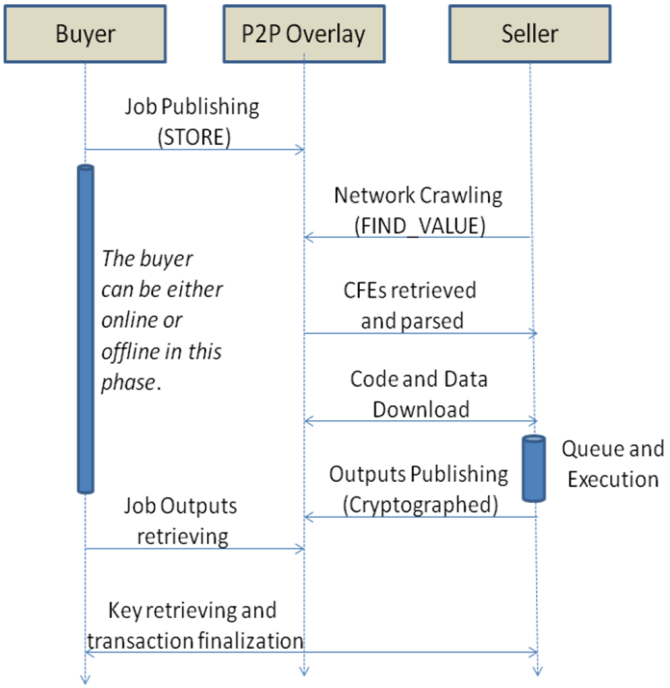


Figure 2. Sequence diagram describing the seller and buyer interactions.

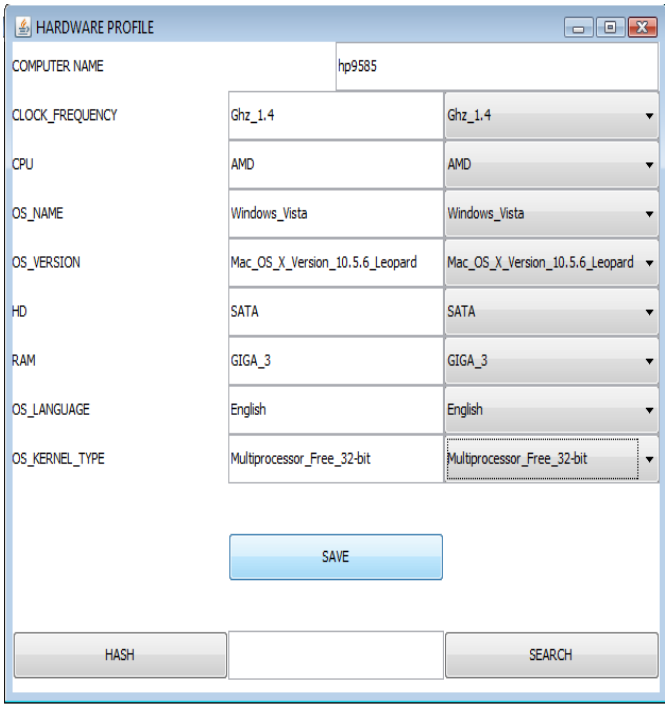


Figure 3. User interface for job publishing and discovery

Sellers, through the Job Discovery module, crawl the network looking for CFEs, using the FIND VALUE message. Once a job is retrieved, its descriptor is analyzed by the Parser module. The descriptor contains information about the type of executable, application requirements, the time constraints, the reward and further details that are summarized in Table I. The Parser, then, will interact with the Buyer's Network module to download code and data required by the job. Notice that, for simplicity, no negotiation mechanism has been implemented: once a job descriptor is retrieved, the Seller checks if the requirements can be fulfilled, then decides if it is worth to accept the task.

The decision-making mechanism does not take in consideration whatever any other Seller could be already on the same CFE. Accepted jobs are managed by the Job Queue module, that will sequentially schedule them. Finally, the Result Manager will interact with the Buyer's Network module in order to return the outputs.

An ontology has been created, that allows to define the application details and the application specific hardware/software requirements in a not ambiguous way. Some of the concepts are listed in Table II.

VI. EXPERIMENTAL RESULTS

In order to evaluate the behavior of the described prototypal implementation, under both functional and performance points of view, a testing environment has been set up. The developed software platform emulates the proposed approach, enabling the analysis of the system dynamics, including the overhead introduced by the adoption of the Kademlia protocol. In Figure 4, a communication diagram of the software platform is showed.

TABLE I. APPLICATION DETAILS

Variable	Meaning
Universe	Specific the kind of application that is been submitted (Exe file, java executable, etc.)
Unique ID	Identifier used to retrieve the code and data inputs on the overlay network.
Executable file	The name of the main executable file
Input	The url where the package can be retrieved
Output	The url where the results can be sent
Contract owner	A unique indentifier of the job submitter
Budget	The reward offered for the job execution
Deadline	The date by which the task must be completed
Owner email	Email contact of the owner

TABLE II. APPLICATION SPECIFIC HARDWARE AND SOFTWARE REQUIREMENTS

Variable	Meaning
CPU Architecture	Possible constraints on the CPU type
N.of CPUs	Number of required CPUs
RAM	Minimun amount of available ram required
Libraries	Possible required libraries – Optional
OS	Possible required OS – Optional
Storage	Minimum amount of free storage required

Different test cases have been defined. Each of them is characterized by a set of meaningful parameters, whose combination leads to a different statistical behavior of the system. The most relevant parameters that can be set for each test case are:

- Job Arrival rate.
- Number of peers in the system.
- Mean and standard deviation. It depend both on the computational requirement of the task and on the computing power of the seller peer. Times are modeled as Gaussian distributions.
- Cool down (time between subsequent network scans).
- Maximum allowed concurrency level (MAC). It is the number of peers that can simultaneously compete on a single task.

The test analysis has allowed us to detect interesting system dynamics. In particular, we evaluated the mean queue time that a job has been waiting inside a buyer's queue, the mean execution time and the mean time of permanence within the overlay network.

 ISBN: 978-1-61208-153-3

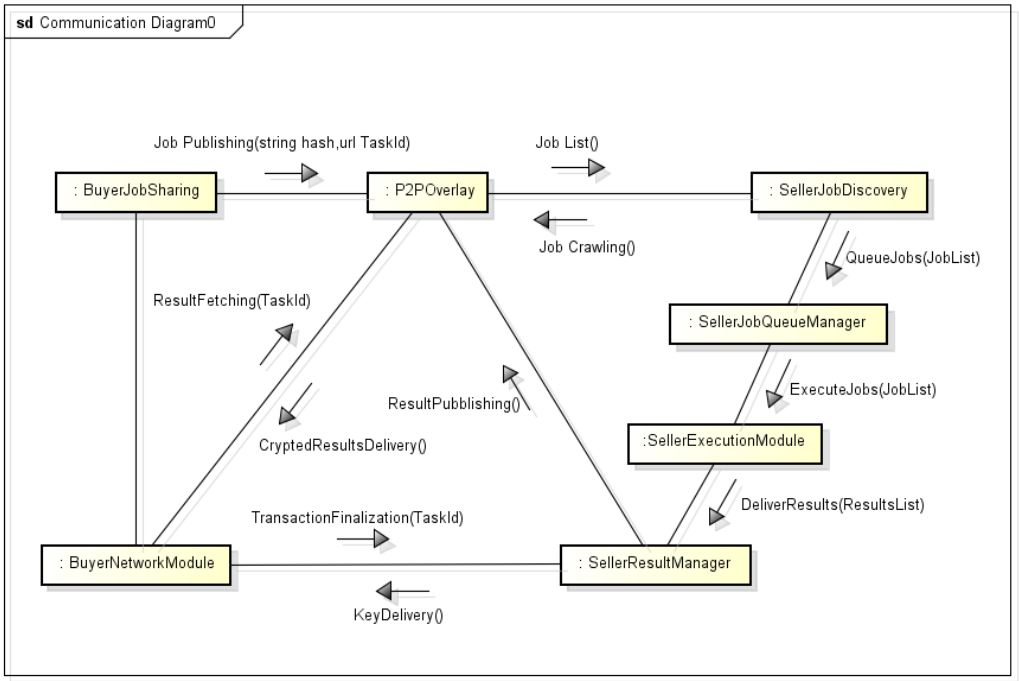


Figure 4. Testing platform: communication diagram

The results are compared with the values obtainable by running the reference job on a single dedicated server: we can model it as a scenario with reference mean execution time, and mean queue time equal to zero. Considering that a traditional PaaS environment allows the customer to autonomously manage the obtained server instance, the reference scenario correspond to what a user would expect by executing the same kind of tasks on a commercial PaaS cloud service (e.g., Amazon EC2 [21]).

For testing purposes, specific assumptions have been made: the first one is that the service providers have, globally, enough available resources to manage the overhead introduced by the competitive scheduling layer.

In other words, it means that, for the set job arrival rate, the global permanence time converges to a finite value. As long as this assumption is proved true, our tests show an improvement in mean system permanence times. Test set 3 describes a scenario where this assumption becomes untrue for one of the tested MAC values. The second assumption is that service providers consider cost-effective to commit resources to compete for job executions rather than keeping them idle.

A. *Test set 1*

The test has been performed with the following parameter settings:

- Job Arrival rate: 5 jobs per minute.
- Mean: 300 s.
- Standard deviation: 40 s.
- Number of peers: 300.
- Cool down: 120 s.

The test has been performed with two distinct MAC level values. As it is evident in Figure 5 and in Table III, in both cases, the mean permanence time of jobs in the system is lower than the mean execution time of the single job. This result is due to multiple peers competing to execute the job and deliver the results, so the actual execution time is definite by the peer that is the quickest one to perform the execution. Notice how an increase of the MAC value has lead to get a better mean of the system permanence times, despite the increasing queue time.

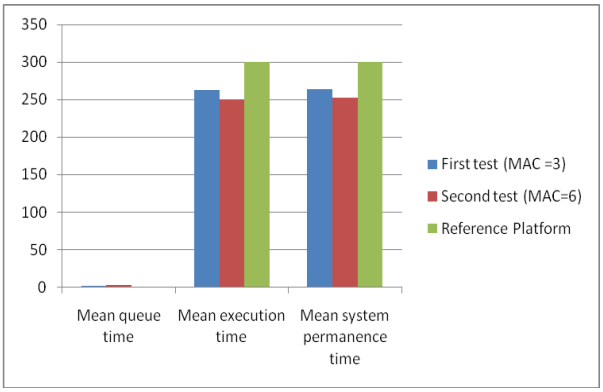


Figure 5. Performance results of test set 1

TABLE III. PERFORMANCE RESULTS OF TEST SET 1

	First test (MAC =3)	Second test (MAC=6)	Reference Platform
Mean queue time	1,339689655	2,348795559	0
Mean execution time	262,7362816	250,6696798	300
Mean system permanence time	264,2982312	253,0184754	300

B. *Test set 2*

The test has been performed with the following parameter settings:

- Job Arrival rate: 6 jobs per minute.
- Mean: 400 s.
- Standard deviation: 40 s.
- Number of peers: 2000.
- Cool down: 120 s.

As it has already been done in the previous scenario, this test has been performed with two distinct MAC level values. Figure 6 and Table IV summarize the results.

This test shows again, for both test cases, an improvement of the system permanence time compared to the baseline execution time. Notice, however, how the increased MAC value (test case 2) does not lead to better overall system permanence times: the increased queue time, due to having too many business peers fighting over each CFE and so less frequent network crawling.

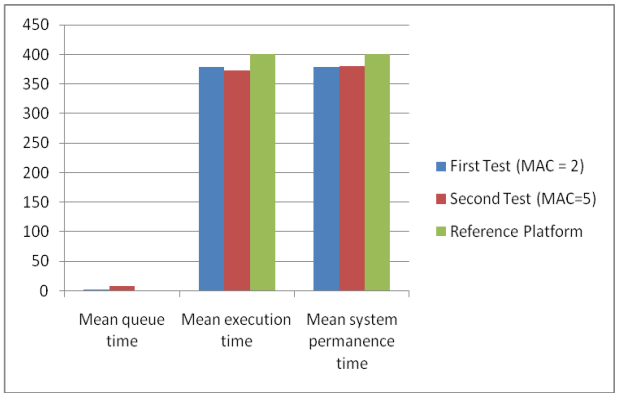


Figure 6. Performance results of test set 2

TABLE IV. PERFORMANCE RESULTS OF TEST SET 2

	First Test (MAC = 2)	Second Test (MAC=5)	Reference Platform
Mean queue time	0,878025466	7,532815631	0
Mean execution time	378,506824	372,9120812	400
Mean system permanence time	379,3848495	380,4448968	400

C. *Test set 3*

The test has been performed with the following parameter settings:

- Job Arrival rate: 6 jobs per minute.
- Mean: 600 s.
- Standard deviation: 40 s.
- Number of peers: 300.
- Cool down: 120 s.

Once again, this test has been performed with two different MAC level values. Results are summarized by Figure 7 and Table V. It is evident that, for a MAC = 6, the assumption of mean permanence time converging to a finite value is not proved: in this case, the reference platform would perform better than the system with the added competitive scheduling overlay.

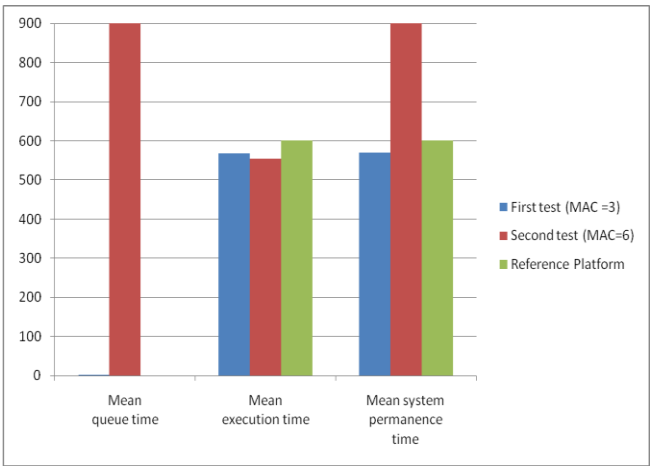


Figure 7. Performance results of test set 3

TABLE V. PERFORMANCE RESULTS OF TEST SET 3

	First test (MAC =3)	Second test (MAC=6)	Reference Platform
Mean queue time	2,2465506542	$\rightarrow+\infty$	0
Mean execution time	567,44207082	554,254621	600
Mean system permanence time	569,68862147	$\rightarrow+\infty$	600

VII. CONCLUSION

We presented a competitive approach for job scheduling in a P2P overlay of Cloud providers. Cloud technology is used for effective set-up of virtual resources which are compliant with application's requirements. The P2P overlay is used to publish and discover jobs' "calls for execution" and to overcome the complexity of negotiation mechanisms. Competition of providers is investigated to implement a business model where the cost is fixed by the users and providers try to respond and adapt.

We investigated the effectiveness of the proposed approach by implementing a framework that emulates the

 ISBN: 978-1-61208-153-3

network protocols and the peers behaviors. The experimental activities validate our hypothesis that a competitive approach, in distributed scheduling environments, does not only decreases the threshold required to access the facilities, but it can also lead, if properly set up, to substantial performance gains. More specifically, this objective can be achieved by setting an appropriate level of competition between the infrastructure managers. A fine balancing must be pursued: too many competitors increase the concurrence over each submitted job. As a result, we notice a degradation of the system performances due to longer queue times.

VIII. ACKNOWLEDGMENT

This research is supported by the grant FP7-ICT- 2009-5-256910 (mOSAIC [22]).

IX. REFERENCES

[1] e-IRG, "White paper 2009," June 2009, http://www.e-irg.eu/images/stories/publ/white-papers/e-irg_white_paper_2009_final.pdf, (last accessed 18/8/2011).

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, n. 6, June 2009, pp. 599-616, doi:10.1.1.144.8397.

[3] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky, "Seti@home: massively distributed computing for Seti," Computing in Science and Engg., vol. 3, n. 1, 2001, pp. 78-83, doi:10.1109/5992.895191.

[4] A. Forestiero, C. Mastroianni, and M. Meo, "Self-Chord: a Bio-Inspired Algorithm for Structured P2P Systems," 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, May 2009, pp. 44-51, doi:10.1109/CCGRID.2009.39.

[5] A. Forestiero, E. Leonardi, C. Mastroianni, and M. Meo, "Self-Chord: a Bio-Inspired P2P Framework for Self-Organizing Distributed Systems," IEEE/ACM Transactions on Networking, vol. 18, n. 5, October 2010, pp. 1651-1664, doi: 10.1109/TNET.2010.2046745.

[6] J. Linnolahti, "QoS routing for P2P networking," Helsinki University of Technology, Department of Computer Science, 2004, doi:10.1.1.58.7192, http://www.tml.tkk.fi/Studies/T-110.551/2004/papers/Linnolahti.pdf, (last accessed 18/8/2011).

[7] Y. Wang, L. Wang, and C. Hu, "A QoS Negotiation Protocol for Grid Workflow," Grid and Cooperative Computing (GCC 2006), Fifth International Conference, Dec. 2006, pp. 195-198, doi:10.1109/GCC.2006.14.

[8] G. Antoniu, M. Jan, and D. Noblet, "A practical example of convergence of P2P and grid computing: an evaluation of JXTAs communication performance on grid networking infrastructures," Proc. IEEE Symp. Parallel and Distributed Processing (IPDPS 2008), June 2008, pp. 1-8, doi:10.1109/IPDPS.2008.4536338.

[9] R. Buyya , D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," Concurrency Computat.: Pract. Exper., 2002, vol. 14, pp. 1507–1542, doi:10.1002/cpe.690.

[10] B. Cao, B. Li, and Q. Xia, "A Service-Oriented Qos-Assured and Multi-Agent Cloud Computing Architecture," CloudCom'09, LNCS, vol. 5931, Springer, 2009, pp. 644-649, doi:10.1007/978-3-642-10665-1_66.

[11] S. Venticinque, R. Aversa, B. Di Martino, and D. Petcu, "Agent based cloud provisioning and management: design and protoypal implementation," Proc. of Cloud Computing and Services Science (CLOSER), SciTePress, 2011, pp. 184-191.

[12] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," Concurrency and Computation: Practice and Experience, vol. 17, 2005, pp. 2-4, doi:10.1.1.6.3035.

[13] N. Drost, R. V. van Nieuwpoort, and H. Bal, "Simple Locality-Aware Co-allocation in Peer-to-Peer Supercomputing," Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06), 2006, p. 14, doi:10.1.1.78.1535.

[14] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," In 2nd International Workshop on Peer-to-Peer Systems (IPTPS03), 2003, pp. 118-128, doi:10.1.1.104.7210.

[15] A. Ghatpande, H. Nakazato, O. Beaumont, and H. Watanabe, "Analysis of divisible load scheduling with result collection on heterogeneous systems," IEICE Transactions, vol. 91-B, n. 7, 2008, pp. 2234-2243, doi: 10.1093/ietcom/e91-b.7.2234.

[16] I. Stoica, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking (TON), vol. 11, n. 1, Feb. 2003, pp. 17-32, doi:10.1109/TNET.2002.808407.

[17] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," USENIX, Proceedings of the 2009 conference on Hot topics in cloud computing (HotCloud'09), San Diego, CA, USA, 2009, doi:10.1.1.149.2162.

[18] Survey: "Cloud Computing 'No Hype', But Fear of Security and Control Slowing Adoption," July 2011, http://www.circleid.com/posts/20090226_cloud_computing_hype_security/, (last accessed 18/8/2011).

[19] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS'01), 2002, pp. 53-65, doi:10.1.1.18.6160.

[20] B. Penteado, "JKad: Java implementantion of the Kademlia Network," http://code.google.com/p/jkad/, (last accessed 18/8/2011).

[21] Amazon Elastic Compute Cloud (Amazon EC2), July 2011, http://aws.amazon.com/ec2/, (last accessed 18/8/2011).

[22] mOSAIC, July 2011, http://mosaic-cloud.eu/, (last accessed 18/8/2011).

[23] C. Gonzalo, "Peer-to-Peer (P2P) architecture: Definition, taxonomies, examples, and applicability," Internet Requests for Comment, RFC Editor, Fremont, CA, USA, Tech. Rep. 5694, Nov. 2009, http://www.rfc-editor.org/rfc/rfc5694.txt, (last accessed 18/8/2011).

[24] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, vol. 53, n. 6, 2009, p. 50.

[25] C. Kesselman and I. Foster, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann Publishers, Nov. 1998.

[26] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, and R. Katz, "Above the Clouds: A Berkeley View of Cloud Computing," Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2009-28, Feb. 2009, doi:10.1.1.149.7163, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf, (last accessed 18/8/2011).

[27] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," Communications of the ACM, 2003, vol. 46, no. 2, pp. 43-48, doi:10.1145/606272.606299.

 ISBN: 978-1-61208-153-3

Towards Green HPC Blueprints

Goran Martinovic, Zdravko Krpic
Faculty of Electrical Engineering
Josip Juraj Strossmayer University of Osijek
Osijek, Croatia
e-mail: goran.martinovic@etfos.hr, zdravko.krpic@etfos.hr

***Abstract*—Effectiveness and power consumption is becoming a major problem in high-performance computing. Numbers of researchers are working on methodologies in order to increase efficiency of these systems on hardware and software levels. Several "green" technologies are explained in this paper along with their pros and cons, with the aim of improving the power efficiency for high performance computers and cloud computing systems. Many of the aspects of green HPC are still in their initial stages, so this paper analyzes recent contributions in that respect, and proposes related work for every "green" improvement of HPC systems. It gives detailed blueprints for the Green HPC using state-of-the-art technologies from this field of research.**

Keywords - green computing; high performance computing; cloud computing; energy efficiency.

I. INTRODUCTION

Throughout the history of High Performance Computing (HPC), raw processing power was a primary concern. Various companies have had tendencies to build bigger computer systems in order to solve demanding computing tasks which could not be done on mainstream computer machines, or at least, in a reasonable time. As computer power grew, so did its "pat on the back" - heat dissipation, power consumption, production costs and software costs. A mere "petaflop race" became too expensive to participate in, although some institutions ignored the phenomenon of power consumption increase, claiming that this is a normal evolution of HPC. The "Green Destiny" Project [1] proved them wrong, and caught a tremendous amount of interest in both the computing and business industries. This revelation lead to different theories, but in the end, most computer scientists agreed that the energy footprint from the computers must be reduced, trying thereby to preserve performance. This being the case, various "Green" standards, such as EPEAT; for details see [2], Energy Star 5.0, [3], and RoHS directive [4], that were established as a guide for HPC equipment manufacturers and Cloud Computing (CC) vendors.

The second important reason for reducing HPC energy consumption is of a financial nature. HPC centers and CC system holders tend to exploit their resources in the most economic way, thus increasing the profit. In the world of ever-growing HPC systems, CC systems and service computing, the ability to offer more resources imply large expenses for maintenance, cooling and electric bills.

The purpose of this paper is to give a basic insight into available and proposed methods for the "Greening" of HPC and CC systems, as well as their positive and negative impact on performance and energy savings. Every method will be referred to related work.

The rest of the paper is organized as follows: Section 2 introduces possible "green" solutions for HPC and CC systems, while in Section 3, every solution is elaborated and supported by examples from available sources. Section 4 uses tiered HPC design to pinpoint objects for implementing proposed solutions. Finally, Section 5 concludes the paper and announces future research by authors.

II. SOLUTIONS AND RELATED WORK

Hardware manufacturers are constantly introducing lower power Integrated Circuits (ICs), which are the basis for reducing power consumption, and overall running costs of HPC/CC systems. Certain authors, as in [5] and [6], claim that advanced power management plays a key role in "greening" the computing systems. Some authors, e.g., [7] and [8] propose their vision of reducing the footprint of HPC energy footprint reduction through the use of advanced task management tools (high and low level schedulers and mappers) and frameworks. Other sources partially rely, amongst others, on "smarter applications", efficient programming and reconfigurable compilers, such as in [9] and [10]. However, a true energy-efficient HPC is an ideal combination of all the aspects mentioned. Careless disabling of the compute nodes while they are not in use, switching power states too often, reducing CPU frequency too much, incorporating bulky resource monitoring systems, using over-complicated scheduling systems and algorithms, can do more damage than good, resulting in even minor performance systems and bigger power consumption. The more "Green" technologies are used, the greater care should be taken to successfully balance their impact on reducing power consumption, while trying to keep acceptably high performance. The existing research in this area is based on testing the green methods for computing systems, as in [11] and [12]. In [11], a bit more technical approach is given, and the authors are based mostly on greening the data centers and

 ISBN: 978-1-61208-153-3

servers. In [12], a survey and taxonomy for green computing systems is given, but not all the green aspects are covered. Other topics are discussed in detail.

III. THE "GREENING" LEVELS OF AN HPC SYSTEM

A. Hardware General

Driven by large expenses of running HPC systems, vendors prompted computer manufacturers to build their systems in a more eco-friendly manner. Creating power-efficient hardware is undoubtedly the basis for building Green HPC/CC systems. A single PC consumes on average 200W, which is the power of two light bulbs, but HPC systems such as computer grids and clouds consume much more. Computer grids punched the MW limit in power consumption, and every upgrade demands a new approach for cooling systems and power sources. As the world's top supercomputing organizations found that mere increase in size and number of their systems has a negative impact on running costs, green supercomputing ideas were born. The process of greening computing resources started at hardware level. There were several key technologies that lead to significant energy savings. First, there was denser Very-large-scale Integration (VLSI) of chips, which ultimately led to multi-core chips and an increasing number of integrated components in the CPU die. Intel, for example, has recently announced a 22nm 3-D Tri-Gate technology, [13], which offers lower operating voltages and leaking currents in order to gain more performance per Watt (increasingly popular metrics used for measuring power-efficiency of a computer system). The same manufacturer has announced a 15nm technology by 2013 and 10nm by 2015, continuing to follow Moore's Law. In this way, unnecessary buses are removed from the system, cache and memory bandwidth are increased, providing at the same time solutions to a major problem of HPCs – memory bandwidth limitation. Since these modern HPCs have the memory bandwidth bottleneck, increased energy efficiency is granted. Also, a new era of low power chips emerged, providing less horse power, but with greatly increased power savings and reliability. Hardware technology advances also promise greater control over energy consumption of other components in the computer system. Leading CPU manufacturers, including AMD, Intel, Sun and IBM, offered their representative low power CPUs, which clearly states the importance of power savings, especially for HPCs. The importance of the given fact applies also to Graphics processing units (GPUs), which have recently evolved from a special purpose to an efficient high performance processing units. The Chinese Tianhe-1A is a good example of harnessing GPU power in order to achieve true supercomputing performance, but at lower energy footprint.

With respect to gains, manufacturers claim that the 22nm technology could bring up to a 37% performance increase at low voltage compared to actual 32nm, and a 50% power reduction at constant performance. An increase in VLSI density resulted in multicore processors, CPU-GPU integration and high performance GPUs. Even the other components such as chipsets, PSUs, disk drives and network systems, tend to have reduced energy footprint by using more advanced microcontrollers and ICs.

In relation with cons, as any other new technology, new production facilities should be built, which again questions the "greener process" of technology advance. But, considering many platforms based on the new technology, power saving benefits should be able to overcome production costs.

B. Power Management

The drive towards sustainable IT, which encompasses HPC and CC systems, has encouraged the creation of metrics claiming to quantify energy usage and apply objective math to the measurement of data center efficiency. Even different benchmarks for new age supercomputing systems are proposed, e.g., [14]. There are several metrics proposed for measuring data center power efficiency. The Green Grid, a consortium of IT industry experts has presented a series of proposals for IT facilities power measurement. The Green Grid proposes two key metrics for data center efficiency, and these are: Power Usage Effectiveness (PUE) and Data Center Efficiency (DCE), as in [15]. For example, the metrics of the former is based on the ratio between Total Facility Power (TFP) and IT Equipment Power (IEP):

$$PUE = \frac{TFP}{IEP}. \quad (1)$$

The Green Grid consortium, along with some others, including the work from [5], offer proposals for a complete power assessment of IT installations, which consist of analyzing present states, pinpointing weak points of systems which cause power inefficiency and giving propositions for improvements. Establishing a good Green metrics gives organizations valuable guidelines to reduce their costs by utilizing power management.

In addition, advanced power states have been incorporated in systems for years now. In green HPC, the reason for using multiple power states is to adapt HPC power consumption to real needs. The Advanced Configuration and Power Interface (ACPI) replaced old Advanced Power Management (APM), and introduced new techniques for more thorough power consumption suppression. The ACPI has the ability not only to reduce the processor speed, but also to monitor other components, thus providing greater versatility in a disabling system which is not used. The biggest power consumers in the computer system can be seen in Fig. 1, based on the survey in [11]. The low power states (also called the S-states) are used at the node level. The S-states which have the best power saving/wake up time ratio as suggested by authors, are S3 ("Suspend to RAM"), S4 ("Suspend to disk") and S5 ("soft off"). By using these states, nodes are deactivated when they are not needed, and woken up or turned on when the running HPC/CC system demands more performance. This approach can greatly reduce power consumption, if the time and energy needed to power down or wake up the node do not affect the overall revenue. That is, changing power states can be performance and energy consuming, as highlighted in [16].

 ISBN: 978-1-61208-153-3

Gains can vary depending on the level of power plan adaptation, power state changing frequency, power changing cost of a system, etc. Researchers in [6] predict about 10 – 13% of power saved, up to 32% with energy proportional devices with deviance less than 5%.

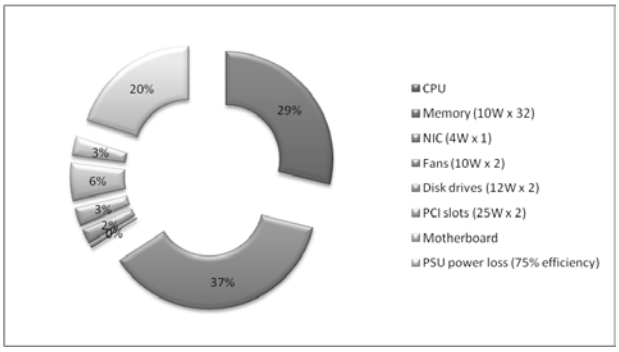


Figure 1. Power consumption in a typical HPC node, based on data from [14]

Power saving policies can significantly improve system scalability.

Cons are mainly based on the power changing costs (wake up time and energy), reduced reliability (especially for hard drives, because of their limited power on/off cycles - 40,000 on-off cycles claimed by HP for their machines), financial burden of adaptation to power requirements, etc.

C. Dynamic Voltage and Frequency Scaling, CPU and Memory Throttling

The primary power saving technology at CPU level is DVFS (Dynamic Voltage and Frequency Scaling), which enables current, voltage and frequency reduction of CPU when its utilization is below some threshold. Since energy consumption is proportional to frequency squared, DVFS offers a promising approach to reduce energy usage. Another benefit of lowering the frequency is the reliability. In the time of the "GHz race" (in 1990s and early 2000s), processors became more and more subject to failure, which from the perspective of a multi-processor system, such as Grid or Cloud, is an important issue to be reckoned with. However, larger voltage ranges do not improve power efficiency, as shown in [17]. They demonstrated that for sub-threshold supply voltages, leakage energy becomes dominant, making "just in time completion" energy inefficient. They also showed that extending voltage range below half *Vdd* will improve energy efficiency for most processor designs while extending this range to sub-threshold operations is beneficial only for specific applications. Supply voltage can be reduced if frequency of operation is reduced. If reduction in supply voltage is quadratic, then an approximately cubic reduction of power consumption can be achieved. However, it should be noted that frequency reduction slows the operation.

Memory throttling is similar to CPU throttling, but instead of lowering the frequency, it is basically the limitation of memory bandwidth based on the current memory bandwidth request. There are several memory throttling technologies already in use, e.g., Intel's Closed Loop Thermal Throttling (CLTT), and Open Loop Throughput Throttling (OLTT).

The benefits of memory throttling are studied in [18], where authors managed to achieve up to 35% reduction of the total memory power consumption. There were some limitations, however, because the results have shown that 40% of memory power consumption is not controllable, also, a memory bandwidth was limited to 75%. In the system with the memory throughput up to 100%, as much as 60% of memory power consumption can be saved if memory is not needed.

Authors in [18] offer an insight into the types of applications where memory throttling does not improve power savings. These are the applications with low memory requirements, or highly optimized applications with heavy cache usage.

D. Power Aware OS

Low power operating systems are mainly researched for mobile platforms, and embedded systems. But the advances in these operating systems can be applied to the world of HPC. HPC/CC operating systems play the key role in resource efficiency. First, the OS should be aware of the current load of the node, and when the node is loaded, other non-essential tasks should be treated as low priority tasks and be given only a portion of resources. The OS should be aware of the task priority and adjust the resources accordingly. Other features of Green computing such as DVFS, advanced power states, throttling, task scheduling, load balancing and a dynamic adaptation of the HPC environment to the current need should be all issued by the HPC/CC operating system.

Greening level: as much power saving as a combination of installed power saving technologies (if the OS contains routines which successfully exploit them).

Cons: The OS with many processes (which manipulate the power saving technologies) can degrade performance of the HPC.

E. Virtualization

Virtualization is one of the fundamental software technologies that leads to a development of CC systems. Even though virtualization mostly applies to CC systems, migration of this technology to HPC is almost inevitable, so its energy savings can be taken into account when sketching "greener" HPC. Virtualization enables more thorough use of CC systems' resources, because it provides an abstraction of real resources and resource transparency. If a node is to be used, a Virtual Machine(s) (VM) takes possession of the nodes' resources up to VMs maximum. If not all of nodes' resources are taken, another VM or VMs can occupy the rest of the same physical node, which enables efficient use of the resources and application scaling. There are several virtualization technology vendors active, such as Xen, VMWare, KVM and Virtualbox, and a lot of work is based on virtualization in the HPC world (e.g., [12], [19], [20]), thus the term can be regarded as one of the postulates of Green HPC/CC. VMs are the media which can hold other

 ISBN: 978-1-61208-153-3

"green" technologies, such as dynamic power policies, managing Source Level Agreements (SLAs) in cloud computing systems, DVFS, and workload balancing.

Gains are the combination of gains from other "green" technologies contained in VM, especially excellent scalability and resource provisioning.

Cons: VMs can be inert, so resource scaling can be slow. Every VM contains resources for its internal functionality (by OS), so if there are several VMs on one physical node, a great deal of resources is dedicated to it (mostly memory and disk space).

F. Task Scheduling, Application Granulation

A lot of work was conducted in this area even before green computing ideas were born, cf. [21]. The high application granularity level is a key for economical resource provisioning and application scalability, especially on heterogeneous computer systems. That includes optimal parallel programming in the way of making independent tasks that can be executed concurrently. After the application is submitted to HPC resources a high level scheduler takes care of running the application instances, and a low level scheduler/mapper assigns tasks to resources (nodes, processors, cores) in HPC environment, more in [7]. The tasks should be kept as small as possible, so that the applications could be more scalable, and efficiently use up computing resources. There are newer approaches, which already implement energy aware scheduling, for details see [8].

Gains: An intelligent scheduling/mapping system can lead to significant utilization efficiency, good resource provisioning and increased scalability. The exact energy savings can vary.

Problems in this area are caused by different task sizes, which refer to highly heterogeneous jobs for running on computer systems, data dependencies among tasks, scheduling duration and Amdahl's law.

G. Power Aware Compilers

Not only that power aware compilers should be present in embedded systems to reduce power consumption while compiling an application, they could also improve power efficiency of applications running on HPC nodes. These compilers would have direct control over DVFS, DCD (Dynamic Component Deactivation) and power management routines for resources in the HPC system. At the moment of compiling a code, a compiler should adapt the application for efficient execution on the proposed systems, and be aware of the past executions of similar applications and adjust resource usage accordingly. Automatic parallelization compilers could serve as a potent platform for additional power, voltage and frequency aware compilation, and could, in addition to efficient parallelization, improve usage of HPC system resources.

An interesting example, see [19, p. 72], shows that by using the ifort 9.1 compiler for compiling a simple matrix times a matrix (MxM) operation on Pentium 4 running at 2.8GHz, the two loops deliver 0.97GF/s, and by using ifort 10.0 performance rises to 2.6GF/s. From the aforementioned example it can be seen that a certain code can be several times faster on a more efficient compiler. Some of the compiler optimization methods are automatic loops exchange, automatic loop enrolling, replacing the subroutine calls with direct kernels, ignoring the "if" statements in the loops, etc.

Cons: sometimes a more advanced compiler has a trouble understanding the programmer's intention, which can output wrong results.

H. Resource Utilization Monitors

Resource Utilization Monitors (RUMs) should contain up-to-date information about resource utilization in the systems. The information about system load would then be transferred to other vital components of HPC, such as OS, Virtual Machine Managers (VMMs), power aware compilers, prediction policies and task schedulers. Such monitor is the basis for a highly promising green technology – the prediction. Various logs about application runs, errors, power state changes, voltage and frequency changes, VM migrations, CPU and memory load, and other components usage over time can enhance statistical analysis of resource use. An example of such proposed system can be found in [22], describing the OVIS project – an attempt to exploit HPC resources over cloud services. The approach addresses a scalable collection and analysis of resource metrics from both component-health and resource utilization perspectives, and hence it can contribute to the application-tailored resource allocation of hardware and the subsequent allocation and/or migration of virtual resources on the hardware.

Pros: works concurrently with prediction technologies and task scheduling. RUMs can enhance statistics for investing in a HPC upgrade.

Cons: a resource monitor and load balancer uses resources as well, and in the case of a highly heterogeneous environment it can lead to periodic slowdowns.

I. Prediction, Analysis of Past Executions

By monitoring the HPC system load, efficient tuning and adjustments can be made in order to improve future application executions. That involves smart application mapping, removing unnecessary resources, disabling error-prone nodes, analysis of power wastage sources, etc. Significant improvements can be made to reduce power consumption in HPC systems by applying knowledge gained by resource monitoring systems. It is essential that the latest information collected has the greatest priority in future run-time decisions because this technique ensures acceptance of every system change, hardware and/or software-wise without the need to reset the logs.

Pros: most benefits of prediction can be felt in large systems, where statistics is less prone to errors.

Cons: The "bed in" time of such system can make it inoperable in the first period of use. Also, in smaller systems there is more influence of special cases in the statistics of the load, which can lead to false analysis.

 ISBN: 978-1-61208-153-3

J. Component-based Software Engineering

Based on [10], Component-Based Software Engineering (CBSE) is an approach that deals with making different software by combining reusable software components. Making reusable/recyclable software lowers the software production costs significantly, by reducing the equipment and programmer utilization, what indirectly leads to power savings. However, an impact of CBSE to Green HPC is not entirely known, and it is part of future research that will be done by authors.

Pros: Lower software production costs.

Cons: Components can often be very large, leading to unnecessary overhead and diminished gains.

K. Programmer's Contribution

All the research in making "greener" HPCs in the world would not suffice if the programmer's role in efficient use of resources is ignored. This is the final frontier for a battle against the energy spill in HPC/CC systems, and thus the most important one. Programmer's role gives utter importance to conducting research in performance optimizations in programming, and making the right tools for parallel programming, debugging and compiling. After all, if all of the technologies mentioned in this work are inaccessible by the programmer, they cannot be utilized.

Pros: A high level of power adaptation for every aspect of HPC/CC systems.

Cons: Programmer's role is becoming more automated, and error-prone.

L. Cyber-physical Approaches

Several researches regarding usage of physical characteristics of HPC nodes in order to reconfigure the system or reschedule tasks are conduced, but they are still in their infant phase. In [20], thermal-aware scheduling was proposed with the purpose of reducing hot air recirculation among nodes in the HPC system. The load is balanced by means of scheduling tasks according to the node temperature. Other propositions involve various real-time power meters attached to nodes with the purpose of scaling the load corresponding to current power consumption.

Pros: Many physical characteristics can complement the information needed to increase energy efficiency. These characteristics often carry true information about the node state, and bring valuable information on how to upgrade energy efficiency.

Cons: Cyber-physical approaches demand additional systems, which inject supplementary problems into HPC environment. There is also limited physical information which is relevant to power efficiency improvement.

IV. GREEN HPC BLUEPRINT

If the taxonomy from [9] is adopted, the HPC or CC systems can follow the multi-tiered hardware architecture shown in Fig. 2. Every method/solution from Section 3 can be applied to (a) certain HPC components tier(s).

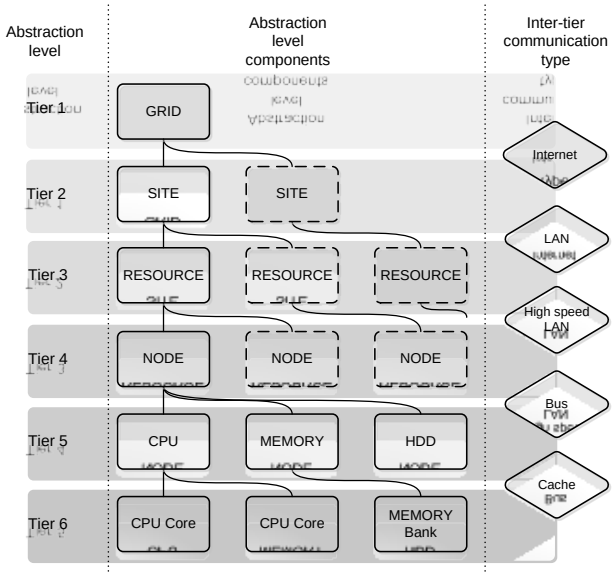


Figure 2. Tiered architecture of a typical HPC system

Fig. 2 in conjunction with Table 1 forms a picture for green HPC blueprints. Table 1 represents corresponding tiers for "greening" method implementations. Tiers are deliberately arranged from the highest to the lowest, because the purpose is to show the highest tier of implementation first, and also because without power efficiency of high tier components, power gains in the lower ones cannot be efficiently exploited. The first level of improvement is undoubtedly the manufacturing technology of hardware. The improvements at these levels stretch through the whole HPC systems, and present the basis for power production. Power management can be applied to tiers 6, 5 and 4, as well as to the network. Changing power states at tier 3 level is not convenient. DVFS and throttling can achieve power savings at tiers 6 and 5. Lower tiers can gain much and improve greening technologies at these levels. A power-aware OS is implemented in tier 4; these are nodes and virtual machines. Virtualization integrates at tier 5 or tier 4, depending on the SLA in the CC systems. Scheduling can be applied to multiple tiers of the HPC system. Scheduling abstraction is based on the current needs and application properties. Power-aware compilers are implemented at tier 4, but they can directly benefit from all the power-saving technologies of higher tiers. Resource monitoring systems and prediction based methods share the same tier of implementation. These are often tiers below 4, depending on the amount of control over HPC. CBSE, as the majority of software power saving techniques, is implemented at tier 4. Programmer's contribution can be indirectly installed at every tier, but most frequently controllable tiers today are tier 4 and 3. With the advances of future systems, high tier components are becoming more tightly coupled, so the hybrid green methods are going to be introduced. These will comprise several green technologies combined in power efficient fashion, so either of which does not diminish gains of other ones.

 ISBN: 978-1-61208-153-3

TABLE I. POWER EFFICIENT TECHNOLOGIES – TIERS OF APPLICATIONS

"Greening" technology	The tier of implementation
Hardware production technology	tier 6, tier 5
Power management	tier 6, tier 5, tier 4
DVFS, Throttling	tier 6, tier 5
Power-aware OS	tier 4
Virtualization	tier 5, tier 4
Scheduling	tier 6, tier 5, tier 4, tier 3
Power-aware compilers	tier 4
Resource monitoring systems	tier 4, tier 3, tier 2
Prediction	tier 4, tier 3, tier 2
CBSE	tier 4
Programmer contribution	tier 4, tier 3

V. CONCLUSION AND FUTURE WORK

Future HPC and CC systems will have to deal not only with the performance upgrades, but also with the increasingly present green standards. Energy costs of running these systems can be overwhelming even for large institutions. Numerous projects try to address various issues regarding the reduction of power consumption in these systems. These are primarily hardware advances, which cannot be exploited without appropriate software. Prior to applying green techniques for HPC and CC systems their overall revenue should be investigated, because not all of them bring power savings and high performance. This is the reason which implies that such attempts should be thoroughly modeled first. We presented different technologies for "greening" the HPC/CC systems, their pros and cons, and a level of HPC/CC system where these can be applied. Every "green" technology was referred to related work and an example was given. In future work, we plan to extend modeling of "green" technologies in HPC and CC environments, and to investigate new means of enhancing power savings in these systems. The possible combinations of green technologies and hybrid green technologies are going to be researched, as well as their gains compared to contemporary ones. Also, the impact of reusable software in terms of software cost reduction, and thereby also in terms of green HPC systems, is going to be evaluated.

ACKNOWLEDGMENT

This work was supported by research project grant No. 165-0362980-2002 from the Ministry of Science, Education and Sports of the Republic of Croatia.

REFERENCES

[1] M. Warren, E. Weigle and W-C Feng, "High-Density Computing: A 240-Node Beowulf in One Cubic Meter", Proc. ACM/IEEE High-Performance Networking and Computing Conf., Baltimore, MD, USA, Nov. 16-22. 2002, pp. 1-11.

[2] IEEE Standard 1680.1, Section 4, "Environmental Performance Criteria for Desktop Personal Computers, Notebook Personal Computers and Personal Computer Displays", http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1633760, accessed: April 2011.

[3] ENERGY STAR Program Requirements for Computer Servers, www.energystar.gov/ia/partners/product_specs/programreqs/Computer_Servers_Program_Requirements.pdf, 2010, accessed: March 2011.

[4] Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment Directives, www.rohs.gov.uk, accessed: March 2011.

[5] J.R. Stanley, K.G. Brill and J. Koomey, "Four Metrics Define Data Center Greenness", http://uptimeinstitute.org/wp_pdf/(TUI3009F)FourMetricsDefineDataCenter.pdf 2007, accessed: Feb 2011.

[6] T. Minartz, J. M. Kunkel and T. Ludwig, "Simulation of Power Consumption of Energy Efficient Cluster Hardware", Computer Science - Research and Development, Vol. 25, Numbers 3-4, 2010, pp. 165-175.

[7] Ripal Nathuji, Canturk Isci, and Eugene Gorbatov, "Exploiting Platform Heterogeneity for Power Efficient Data Centers", Proc. IEEE Int. Conf. on Autonomic Computing, Jacksonville, FL, USA, 11-15 June 2007, p. 5.

[8] 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. IEEE/ACM Int. Conf. on Cluster, Cloud and Grid Computing, Melbourne, Victoria, Australia, 17-20 May, 2010, pp. 368-377.

[9] R. Gruber and V. Keller, HPC@Green IT, Springer, 2010.

[10] J. Sametinger, Software Engineering with Reusable Components, Springer-Verlag, 1997.

[11] L. Minas and B. Ellison, "Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers", Intel Press, 2009.

[12] A. Beloglazov, R. Buyya, Y.C. Lee and A. Zomaya, "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems", Advances in Computers, Vol. 82, 2011, pp. 47-111.

[13] J. Bruner, "Intel 22nm 3-D Tri-Gate Transistor Technology", Intel Newsroom, http://newsroom.intel.com/docs/DOC-2032, May 2011, accessed: May 2011.

[14] Graph 500, http://www.graph500.org, accessed: Jan. 2011.

[15] Recommendations for Measuring and Reporting Overall Data Center Efficiency, www.thegreengrid.org/~/media/WhitePapers/RecommendationsforMeasuringandReportingOverallDataCenterEfficiency2010-07-15.ashx?lang=en, 2010, accessed: May 2011.

[16] N. Vasić, M. Barisits, V. Salzgeber and D. Kostić, "Making Cluster Applications Energy-Aware", Proc. 1st Workshop on Automated Control for Datacenters and Clouds, Barcelona, Spain, June 19, 2009, pp. 37-42.

[17] B. Zhai, D. Blaauw, D. Sylvester and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling", Proc. 41st Ann. Design Automation Conf., San Diego, CA, USA, June 7-11, 2004, pp. 868-873.

[18] H. Hanson and K. Rajamani, "What Computer Architects Need to Know About Memory Throttling", Proc. IBM Workshop on Energy Efficient Design, Saint Malo, France, May 14, 2010, pp. 44-49.

[19] D.A. Menasce and M.N. Bennani, "Autonomic virtualized environments", Proc. Int. Conf. Autonomic and Autonomous Systems, Silicon Valley, CA, USA, July 19-21, 2006, p. 28.

[20] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S.K.S. Gupta and S. Rungta, "Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers", Computer Networks, Vol. 53, Issue 17, 2009, pp. 2888-2904.

[21] H.J. Siegel and S. Ali, "Techniques for Mapping Tasks to Machines in Heterogeneous Computing Systems", Journal of Systems Architecture, Vol. 46, Issue 8, 2000, pp. 627-639.

[22] J. Brandt, A. Gentile, J. Mayo, P. Pébay, D. Roe, D. Thompson and M. Wong, "Resource Monitoring and Management with OVIS to Enable HPC in Cloud Computing Environments", Proc. IEEE Int. Symp. Parallel & Distributed Processing, Anchorage, AK, USA,16-20 May 2009, pp. 1-8.

 ISBN: 978-1-61208-153-3

A Risk Assessment Framework and Software Toolkit for Cloud Service Ecosystems

Karim Djemame
School of Computing
University of Leeds
Leeds, UK LS2 9JT
scskd@leeds.ac.uk

Django J. Armstrong
School of Computing
University of Leeds
Leeds, UK LS2 9JT
een4dja@leeds.ac.uk

Mariam Kiran
School of Computing
University of Leeds
Leeds, UK LS2 9JT
scsmk@leeds.ac.uk

Ming Jiang
School of Computing
University of Leeds
Leeds, UK LS2 9JT
scsmj@leeds.ac.uk

***Abstract*—As the realization of Cloud computing environments advances from a simple and single private Cloud towards a more complex Cloud Service Ecosystem consisting of multiple coexisting public or hybrid Clouds, there are emerging high level concerns such as risk, trust, ecological, security, cost and legal factors that underpin the non-functional properties of the ecosystem. These concerns are beyond the traditional focus of providing functionalities at levels close to a single Cloud infrastructure such as hardware resource virtualization. In this paper we present ongoing research work to analyze and address the risk factor in such a Cloud Service Ecosystem for the purpose of optimizing Cloud service. The main contributions of the work are the design and implementation of an effective and efficient risk assessment framework (methodologies of risk identification, evaluation, mitigation and monitoring) for Cloud service provision. Together with the corresponding mitigation strategies, the framework provides technological assurance that will lead to a higher confidence of Cloud service consumers on one side and a cost-effective and reliable productivity of Cloud Service Provider (SP) and resources organized by individual Infrastructure Provider (IP) on the other side. The design of the risk assessment framework and its software toolkit implementation is part of the research and development work of the OPTIMIS (Optimized Infrastructure Services) project whose objective is to enable an open and dependable Cloud Service Ecosystem that delivers IT services that are adaptable, reliable, auditable and sustainable both ecologically and economically.**

Keywords-risk assessment; Cloud services; service provider; infrastructure provider; optimization.

I. INTRODUCTION

The current model of a single Cloud service infrastructure mainly focuses on providing functionalities at levels close to the infrastructure, e.g., improved performance for virtualization of all, compute, storage, and network resources, as well as necessary raw functionality such as virtual machine migrations and server consolidation. However, for a Cloud Service Ecosystem that consists of multiple coexisting Cloud architectures, there are higher level concerns (e.g., risk, trust, ecological and legal factors) that should be addressed for the purpose of an optimized Cloud service provision. The purpose of this research work is to analyze and address the risk factor in a Cloud Service Ecosystem. Although in its most general sense, risk can be defined as the combination of the probability of an event occurring and its consequences and constitutes both "opportunities" for benefit (upside) and "threats" to success (downside) [20], in the context of this work, only those undesirable events with negative consequences are considered and need to be mitigated.

One of the hurdles that prevent a Cloud service consumer from adopting Cloud services is the lack of adequate confidence of those services in term of the uncertainties associated with their qualities and levels in the ecosystem. Although the provision of a zero-risk service is not practical, if not impossible, an effective and efficient risk assessment of service provision, together with corresponding mitigation mechanisms, may at least provide a technological insurance that will lead to a higher confidence of Cloud service consumers on one side and a cost-effective and reliable productivity of Cloud Service Provider (SP) and resources organized by individual Infrastructure Provider (IP) on the other side. In this research, confidence is defined as the expectation of a successful fulfillment of a Service Level Agreement (SLA) agreed between a Cloud service consumer and an SP. The notion of "cost-effective and reliable productivity" is defined as a provider's capability of fulfilling an SLA through the entire cycle of the service provision, and at the same time realizing its own business level objects of an SP (e.g., make a certain amount of profits) and high resource utilization efficiency of an IP. By aiming this win-win target, this research work proposes a general risk assessment framework of Cloud service provision in term of assessing and improving the reliability and productivity of fulfilling an SLA in a Cloud. Based on this framework, a software toolkit is being designed and implemented, as a basic risk factor related optimization module, which is able to be integrated into other high level Cloud management and control software systems for both SP and IP.

Although risk factor related assessments for deciding risk levels are the main concerns of this work, we also consider that the decision making procedure of how to apply corresponding mitigation solutions to already identified risks in a Cloud Service Ecosystem may involve considerations on other higher level factors such eco-efficiency, cost, security and trust. In case such factors constrain the application of mitigation solutions in one way or another, certain mitigation strategies should be identified to optimize the executions of these mitigation solutions.

 ISBN: 978-1-61208-153-3

The main objective of OPTIMIS (Optimized Infrastructure Services) project [6] is to enable an open and dependable Cloud Service Ecosystem that delivers IT services that are adaptable, reliable, auditable and sustainable both ecologically and economically. The key goal of OPTIMIS is to allow organizations to automatically and seamlessly externalize services and applications to trustworthy and auditable Cloud providers. In the context of OPTIMIS, risk assessment will be applied at the Cloud service construction, deployment and operation phases supporting a wide range of scenarios such as Cloud bursting and Cloud brokerage that will be present in a fully developed Cloud Service Ecosystem of the future. Such mechanisms for managing risk for Cloud-based services which consider inherent aspects of Clouds such as energy consumption, the cost of reconfiguration and migration, and the reliability and dependability of the provided services will maintain secure, cost-effective and energy-efficient operations.

The main contributions of this paper are the design and implementation of an effective and efficient risk assessment framework (methodologies of risk identification, evaluation, mitigation and monitoring) for Cloud service provision. Together with the corresponding mitigation strategies, the framework provides technological assurance that will lead to higher confidence in Cloud providers for Cloud service consumers on one side and cost-effective, reliable and productive Cloud service provider's resources on the other side.

The rest of the paper is structured as follows: in Section II, related work on applying risk management methodologies into utility computing areas, such as Grids and Clouds is surveyed. The risk assessment framework for Cloud Service Ecosystems proposed by this research work is described in Section III; the corresponding software toolkit for the implementation of this risk assessment framework is discussed and introduced in Section IV; in Section V, use cases of the framework and software toolkit in the context of the OPTIMIS project are introduced. Finally, the conclusion of current work in progress is presented in Section VI, in which future work is also introduced and discussed.

II. RELATED WORK

In recent years, the principles and practices of risk assessment/management were being introduced into the world of utility computing such as Grid and Clouds either as a general a methodology [5][7]14] or focusing on a specific type of risk, such as security and SLA fulfilment [13] and [19]. In this section, we conduct a balanced introduction to cover these two aspects.

In [1], an extended Confidentiality Risk Assessment and Comparison (CRAC) method [2], CRAC++, is proposed to assess confidentiality risk in IT outsourcing. The aim of this method is to enable the specification of confidentiality requirements in an SLA between a client and IT resource provider. The method claims that it is able to satisfy six criteria of confidentiality level specification approach: specified confidentiality level is not based on percentages of data loss; assessment is not based on monitoring incidents, no disclosure of confidential information is required to a provider, ease of use; it is repeatable and will increase the client's understanding of confidentiality risks in this outsourcing relationship. The most unique feature of the method is that it tackles two hard problems regarding the specification of confidentiality requirements: 1) confidentiality incidents cannot be monitored, since attackers who breach confidentiality try to do this unobserved by both client and provider, and 2) providers usually do not want to reveal their own infrastructure to the client for monitoring or risk assessment.

In [3], the design, implementation and evaluation of separate and integrated risk analysis methods for a commercial computing service to support successful utility computing model is introduced. By departing from two new challenges facing a commercial computing service in order to support a utility computing model: (i) "what are the objectives or goals it needs to achieve in order to support the utility computing model", and (ii) "how to evaluate whether these objectives are achieved or not", the paper identifies four essential objectives that are required to support the utility computing model: (i) manage wait time for SLA acceptance, (ii) meet SLA requests, (iii) ensure reliability of accepted SLA, and (iv) attain profitability. Based on the analysis on the nature of these objectives, "risk assessment on resource management policy" is identified as the key evaluation methodology to examine whether resource management policies are able to achieve the objectives. Both the separate and integrated risk analysis methods evaluate a policy using two indicators: performance, as the value measure of the policy, and volatility, as the risk measure, that is able to "reflect how performance values fluctuate and thus the consistency of the policy in returning similar performance values". The separate risk analysis analyses the performance and volatility involved in a single objective for a particular scenario and the integrated risk analysis assesses a combination of multiple objectives with different weights used to denote the importance of each objective. These weights for various objectives provide a flexible means for the service provider to easily adjust the importance of an objective and determine its level of impact on the overall achievement of a combination of objectives. Most importantly, the crucial impact of the integrated risk analysis method is emphasised by simulation results that "an objective that is not achieved can severely impact on the overall achievement of other objectives. Thus, it is essential to examine the achievement of all key objectives together, rather than each standalone objective to correctly identify the best policy that can meet all the objectives."

In [4], a novel "insurance" mechanism is proposed as a risk management method that is "primarily used to hedge against the risk of a contingent loss due to unfavourable and uncontrollable events". According to this mechanism, a service insurer in the Cloud is established to decide and collect insurance premium from a service provider, send

 ISBN: 978-1-61208-153-3

compensation to a service consumer; a service provider negotiates an insurance contract with the service insurer; a service consumer submits a claim to the service insurer. Since a service consumer is not the payer of premium but able to claim compensation in case a loss was caused by the service provider, it will be relatively "risk free" for the consumer to use the service confidently. A Cloud Risk Assessment and Management (Insurance) Reference Model, is established based on the extended Zachman framework [9] with the service/information assurance, integrity and analysis, and also the layered reference Service Oriented Architecture (SOA) security reference model [10].

In [13], a quantitative risk and impact assessment framework (QUIRC) is presented to assess the security risks associated six key categories of security objectives (SO) (i.e., confidentiality, integrity, auditability, multi- party trust, mutual auditability and usability) in a Cloud computing platform. The quantitative definition of risk is proposed as a product of the probability of a security compromise, i.e., an occurring threat event, and its potential impact or consequence. The overall platform security risk for the given application under a given SO category would be the average over the cumulative, weighted sum of *n* threats which map to that SO category. In addition, a weight that represents the relative importance of a given SO to a particular organization and/or business vertical is also necessary and their sum always adds up to 1. This framework adopts a wide-band Delphi method [18], using rankings based on expert opinion about the likelihood and consequence of threats, as a scientific means to collect the information necessary for assessing security risks. The advantage of this quantitative approach of risk assessment is that it enables vendors, customers and regulation agencies the ability to comparatively assess the relative robustness of different Cloud vendor offerings and approaches in a defensible manner. However, the challenge and difficulty of applying this approach is the meticulous collection of historical data for threat events probability calculation, which requires data input from those to be assessed Cloud computing platforms and their vendors.

In [5], a SEmi-quantitative BLO-driven Cloud Risk Assessment (SEBCRA) approach that is aware of the Business-Level Objectives (BLOs) of a given Cloud organization is presented. The approach is designed for a Cloud Service Provider (CSP) to improve the achievement of a BLO, i.e., profit maximization, by managing, assessing, and treating Cloud risks. The core concept on which this approach is based is that "Risk Level Estimation for each BLO is proportional to the probability of a given risk and its impact on the BLO in question". Once risk has been assessed, the Risk Treatment sub-process defines potential risk-aware actions, controls, and policies to conduct an appropriate risk mitigation strategies, such as, avoid the risk, by eliminating its cause(s), reduce the risk by taking steps to cut down its probability, its impact, or both, accept the risk and its related consequences or transfer or delegate the risk to external organizations. In an exemplary experimentation, the risk assessment approach demonstrates that it enables a CSP to maximize its profit by transferring risks of provisioning its private Cloud to third-party providers of Cloud infrastructures. This risk assessment approach can be extended to tackle scenarios where multiple BLOs are defined by a CSP and also work as an autonomic risk-aware scheduler, which will be based on business-driven policies and heuristics that help the CSP to improve its reliability.

The work in this paper focuses on a framework that supports risk assessment at the Cloud service deployment and operation phases. It supports not only service and infrastructure providers, but a wide range of scenarios such as Cloud bursting and Cloud brokerage as well.

III. A RISK ASSESSMENT FRAMEWORK

Risk assessment allows improving the foundations of the Cloud infrastructure to help manage and anticipate the risks or opportunities:

- Helping to provide a framework for identifying the risks that present threats to the Cloud.
- Facilitating discussion among the various partners during the development process.
- Foresee potential dangers or risks before they occur and implement mitigation strategies to compensate for them.
- Building an infrastructure for monitoring these risks over time and identifying new risks when they arise.

In Cloud computing, risk needs to be considered at all phases of interactions and investigated at each service stage in relation to the *assets* which need to be protected. Two stakeholders are involved: Service Providers (SP) during the service deployment and operation, and the Infrastructure Providers (IP) during admission control and internal operations. In OPTIMIS, various use cases will be considered for depicting a Cloud scenario as discussed in Section V. These use cases will affect the assets involved as well as the kind of interactions taking place presenting new challenges for risk assessment.

In addition to the different use cases and interactions, risk will be assessed based on *categories* which will help to manage it and the mitigation strategies to be applied. For instance all risks associated with service level agreements (SLAs) can be identified as legal issues and would thus need mitigation strategies from the legal realm.

In addition to identifying the risk categories, each risk item will be assessed thanks to a level of impact and likelihood. For simplicity, the risk level can be labeled in the range from 1 to 5 to show its intensity (1-very low, 2- low, 3-medium, 4- high, 5-very high). The risk level will help manage the risk items from most threatening to the least impact helping with the mitigation strategies to be adopted later. This information will be available in the risk inventory.

A. Service Provider

A service provider is responsible for matching the end-user requirements with the correct IPs to ensure the required demand is met. To achieve this, the SP needs to be risk aware of each IP and ranks them accordingly.

 ISBN: 978-1-61208-153-3

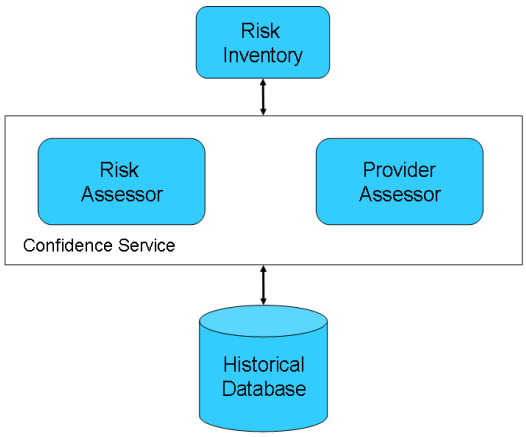


Figure 1. Service Provider – Risk Assessment Components.

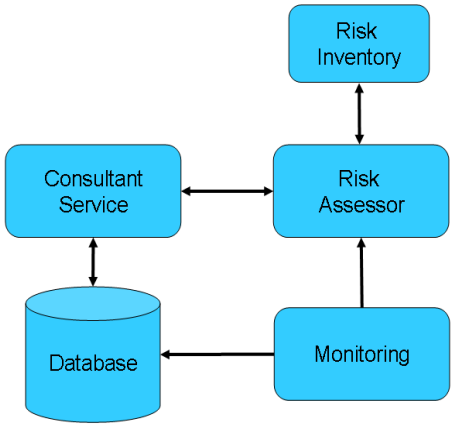


Figure 2. Infrastructure Provider – Risk Assessment Components.

Figure 1 shows the various components the SP will use to fulfill its purpose - a Confidence Service (comprising a Risk Assessor and a provider assessor), a risk inventory and a Historical Database for recording past SLA transactions. The confidence service will take into account the various risks of working with the different IPs accessing the providers. This will be part of the Service Deployment Optimizer (SDO). The SDO will make these decisions based on a stored database of history of working with the different IPs and the risk inventory associated with the different assets involved. A risk inventory is a simple database of risks associated with each asset, their vulnerabilities and threats. This would also contain risk mitigation strategies following risk assessment. All these factors will be accessed by the SDO to choose an efficient selection of the infrastructure provider to run the deployed service.

B. *Infrastructure Provider*

Performing risk assessment at this level increases the performance and quality of the IP. When the SP assesses the IP, the IP would also be assessing the service to be deployed. It will determine an estimated risk if it were to accept the SLA taking into account fault tolerance mechanisms and actions following an SLA violation, in turn improving the IP's reliability and quality of service.

The SP would send a service *manifest* request to the IP containing the feasibility of admitting the new service, with respect to current infrastructure load, predicted future capacity, as well as risk. This helps the IP to determine where to place the virtual machines (VMs) by combining its local management policy with the functional and non-functional requirements.

Figure 2 depicts the structure of the IP risk assessment components. The consultant service takes into account the risk assessor and the database to estimate the risk. This may use data mining tools on the previous history of events of running similar services or working with the same SP. The consultant service can also have access to all the monitoring information keeping the IP on track with the changes. This data can be static or dynamic in nature about its resources and the current service execution.

Examples of such information are the current workload, system outages, temporary performance shortages, monitored network traffic, experts' availability, or general information regarding the number of services to operate. The monitored data helps to determine bottlenecks in the IP's infrastructure so that the provider can improve its capacity planning, administration, and management of its resources. This leads to higher, cost-effective productivity of virtualized resources [21][22].

C. *Risk Inventory*

Various research areas such as business have developed risk inventories for determining how certain risks can be managed and evaluated to be brought up to an acceptable level. Most of the steps towards creating and refining of a risk inventory differ in relation to their purpose and context in which they are applied. A set of processes are identified to create and manage a risk inventory for the implementation of the framework:

1. Determine which use case scenario to focus on.
2. Determine the areas of interaction in the Cloud. Interaction takes place at various levels such as end-user to service provider or service provider to infrastructure provider. During each of these levels an SLA is agreed between parties and its fulfillment monitored.
3. Identify the assets involved which need to be protected from external or internal dangers (risk), as well as the vulnerabilities and threats these assets may have during operation.
4. Identify the risk triggering factors for these assets.
5. Identify the relationships between assets and various factors or events which may lead to risk mitigation.

Therefore the risk mitigation strategy would depend on the use case, asset at risk, and the event which may lead to activate the risk mitigation strategy to reduce it. Risk may also be dynamic and change depending on the situation and activities in the Cloud. These could be changes in policies, transactions etc. This introduces an additional dimension to the risk mitigation strategies which may vary with time.

 ISBN: 978-1-61208-153-3

D. Risk Assessment Models and Risk Categories

Risk assessment also depends on the time of operation during the Cloud service lifecycle. This allows the risk level to change over time. Various risk models can then be introduced to choose relevant mitigation strategies related to concrete situations and recognized threats. The risk models being investigated for this purpose are as follows:

- Probabilistic Risk Model - Risk is a compound of the probability of a problem occurring and the impact of the problem occurring. The probability would depend on the frequency of past problems over time.
- Possibilistic Risk Model – using stochastic processes such as Gamma distributions to predict the failure of a physical machine, Virtual Machine (VM) etc.
- Hybrid Risk Model – A combination of the two above models to predict and assess the risk on the probability of occurring events. Hybrid risk models allow different kinds of risks to be measured. This is because certain aspects can have a numerical probability attached to it for the risk actually occurring, but some events may have a dynamic nature to them, as certain exposures may lead to various relationships among the variables to actually propagate the risk.

Such models have been the focus of the work in [19] to enable a Grid provider to identify infrastructure bottlenecks (considering physical machines only) and mitigate potential risk, in some cases by identifying fault-tolerance mechanisms to prevent SLA violations. Moreover, a Grid broker provides the functionality to evaluate the risk associated with such provider by incorporating provider reliability into the risk models in order to verify the expected integrity of a provider's guarantees when they make any SLA offer [23].

Risk Category: **Technical**
Asset identified: Hardware
Vulnerability of asset: Poor maintenance
Threat to asset: Unresponsive system
Resulting risk item: Reduction in availability
Risk Likelihood: Low (2) [Range 1-5]
Risk Impact: Medium (3) [Range 1-5]
Resulting Risk level: Product of risk likelihood and risk impact [Range 1-25]
Risk event: Hardware failure
Resulting risk mitigation: Duplicate data, maintain hardware

Risk Category: **Policy**
Asset identified: SLA
Vulnerability of asset: Lack of jurisdiction information
Threat to asset: Breach in data confidentiality
Resulting risk item: Changes in jurisdiction
Risk Likelihood: Very high (5) [Range 1-5]
Risk Impact: High (4) [Range 1-5]
Resulting Risk level: Product of risk likelihood and risk impact [Range 1-25]
Risk event: Redeployment of data
Resulting risk mitigation: Seek legal advice

Risk Category: **General**
Asset identified: Security
Vulnerability of asset: Unprotected password
Threat to asset: Unrestricted access to data
Resulting risk item: Data leaks
Risk Likelihood: High (4) [Range 1-5]
Risk Impact: High (4) [Range 1-5]
Resulting Risk level: Product of risk likelihood and risk impact [Range 1-25]
Risk event: System hacks
Resulting risk mitigation: Encrypting data

Risk Category: **Legal**
Asset identified: SLA
Vulnerability of asset: Illegal clauses in the contract
Threat to asset: Sued
Resulting risk item: Ongoing legal dispute
Risk Likelihood: Low (2) [Range 1-5]
Risk Impact: High (4) [Range 1-5]
Resulting Risk level: Product of risk likelihood and risk impact [Range 1-25]
Risk event: Negligence
Resulting risk mitigation: Audit SLAs

Figure 3. Examples of Risk Categories.

The risk models under investigation will be applied to assess the risk on a number of groups of risks or categories. The various risk categories identified, with an example of an associated risk are:

- *Technical* – Hardware, VM failure
- *Policy* – Data jurisdiction policies or other issues which match requirements and considerations (prior to deployment).
- *General* – Various general issues such as security, data applications or processes (as assets to be protected during the different phases of the cloud lifecycle).
- *Legal* – SLA issues

An example of each of category is presented in Figure 3.

E. Risk Mitigation Strategies

Following the assessments on various risk factors and identification of associated mitigation solutions, where possible, appropriate mitigation strategies will be decided to implement these solutions. In general, mitigation strategy can be risk avoidance, limitation, retention**,** transfer and acceptance [11]. Within the context of our work, risk avoidance and limitation are the main strategies to be applied and the selection and execution of mitigation solutions will be considered as an optimization problem.

Since the nature of mitigation is to take precautionary actions before the occurrence of risk, time constraint and cost of a mitigation solution are key factors for deciding which mitigation strategies to choose and how to deploy them. When multiple risk factors need to be mitigated at the same time, it will be more complex to make an optimized decision under time and cost constraints. One example is that a set of

 ISBN: 978-1-61208-153-3

risk mitigation tasks with known, arbitrary execution times, need to be implemented by some identical high level risk mitigation solution executers by a given deadline. The problem is to schedule all of the mitigation tasks onto the least number of executers so that the deadline is met. This is a classic One-Dimensional Bin Packing problem in particular and combinatory optimization problem in general. Hence, this work is investigating optimization algorithms to help make decisions for scenarios as illustrated in these examples.

IV. A RISK ASSESSMENT SOFTWARE TOOLKIT

One of key design principles of a risk assessment software toolkit is to make it a self-contained independent functional component that is able to perform for Infrastructure Providers (IPs) and Service Providers (SPs) and be adopted, in either full or in part, by higher level Cloud management and control software system for higher level optimization purposes such as SP's brokerage for multiple IPs.

Following the logical structure of the risk assessment framework described in Section III, the toolkit is designed to physically consist of two independent parts: SP Risk Assessment Tool (SPRAT) and IP Risk Assessment Tool (IPRAT). For the SPRAT, its high level functions (e.g., evaluate the reliability of a specific IP offer) are mainly exposed by its external interfaces defined in its Confidence Service sub-component. Other lower–level functions such as the evaluation of the risk associated with an IP's offer and evaluation of IP's profile is provided by the external interfaces of the Risk Assessor sub-component and the Provider Assessor sub-component respectively. The Risk Inventory and Historical Database sub-components are private to the SPRAT and no external interfaces are provided by them. The Risk Inventory is designed as a knowledge base to consist of facts, scenarios, and reasoning rules for risk assessments related decision-making activities of the SPRAT.

For the IPRAT, its high level functions (e.g., evaluate the risk fulfilling a given service manifest of a specific SLA) are mainly exposed by its external interfaces defined in its Risk Assessor sub-component. Other lower–level functions such as data-mining of past failure events in an IP are provided by the Consultant Service sub-component. These lower-level functions are not purely private for the IPRAT. The Risk Inventory and Historical Database sub-components are also private to the IPRAT and no external interfaces are provided by them. For the IPRAT, its Risk Inventory is designed as a knowledge base to consist of facts, scenarios, and reasoning rules that are related to lower level hardware and software resources. The Historical Database sub-components is also private to the IPRAT. In addition, IPRAT's Monitoring sub-component includes two parts: one is the risk event detection and alarm part, and the other one is the lower-level hardware and software runtime status collectors. From the implementation perspective, the second part can be based on a third-party data monitoring and collection software, such as Nagios [12], as a plug-in, and will depend on the scalability and efficiency of it.

V. USE CASES IN THE CONTEXT OF OPTIMIS

In the OPTIMIS toolkit, risk is analyzed in the context of three dimensions: use case, actor and time. The toolkit tackles five Cloud uses cases that are in various stages of realization in the current Cloud ecosystem. They are: i) Private, ii) Bursting, iii) Multi-Cloud, iv) Federated and v) Brokerage [6]. These use cases have various implications for OPTIMIS as the differing goal of each contribute to what vulnerabilities an asset may have and thus its associated risk factors. The different Business Level Objectives of the SP and IP actors play a part in deciding the importance of risk because the execution of high-level strategies alter the importance and applicability of risk in a given situation. In addition, the lifecycle of a Cloud service adds a temporal aspect to risk assessment. Cloud Service Lifecycle is comprised of three phases: Service Construction, Service Deployment and Service Operation.

At Service Construction a service is developed, composed and configured. This entails packaging the core elements of a service and its dependencies together, the configuration of the service manifest that describes the functional parameters of each core element within the service and preparation of the VM images used to run the service. The Service Deployment phase sees the deployment of a service onto an IP. An IP is selected using a filter mechanism to decide, using Trust, Risk, Eco-efficiency and Cost (TREC) factors, which IP is most suitable to use for a given service manifest.

Finally at Service Operation, a service begins execution on a selected IP and is continually monitored.

A. Optimis Cloud Use Cases

The use cases are outlined in the following subsections and illustrated in Figure 4 which provides the vision of the OPTIMIS Cloud ecosystem.

1) Private Cloud

In the Private Cloud use case an SP and IP within the same administrative domain cooperate to provision resources for one or more services using internal infrastructure.

2) Cloud Bursting

In the Cloud Bursting use case an IP at some point during the operation of a service may require additional capacity to manage increases in demand above that which its local infrastructure can accommodate. This requires an IP to initiate the SLA negotiation process with another IP.

3) Multi-Cloud

The Multi-Cloud use case is an extension of the Cloud Bursting use case where by an IP may make use of multiple IPs to provision additional resources. The use case can be distinguished from bursting in regards to the IP selection mechanism used, which evaluates the functional and non-functional requirements of the service manifest and chooses the most appropriate IP for a given component of a service.

4) Federated Cloud

 ISBN: 978-1-61208-153-3

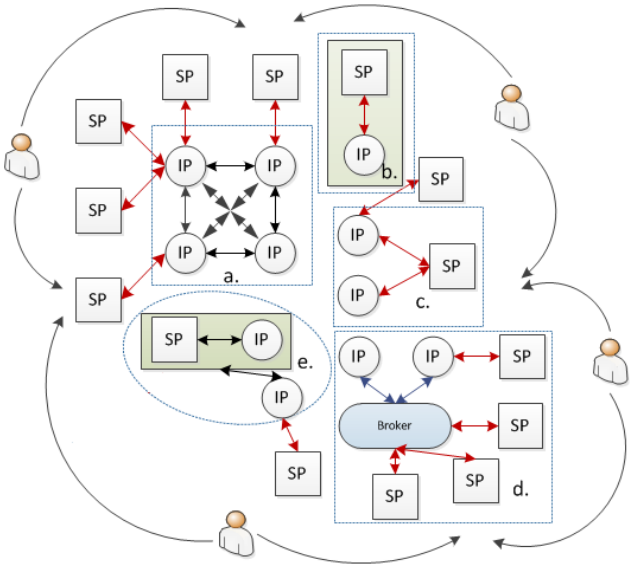


Figure 4: Interactions between Actors on a per Use Case Basis: (a) Federation, (b) Private, (c) Multi-Cloud, (d) Brokerage, and (e) Bursting.

In the Federated Cloud use case an IP provides resources for an SP on behalf and across a collective of IPs working in collaboration. This use case differs from the Multi-Cloud use case as the IPs have previously entered into a mutual SLA between all members of the federation before coming into contact with the SP.

5) Cloud Brokerage

The Cloud Brokerage use case sees the addition of a third actor into the Cloud ecosystem the Broker. The broker acts as an intermediary that facilitates the Cloud Lifecycle and adds value through maintaining a historic database of its encounters with SPs and IPs providing a mechanism to gauge the past performance of an actor and its ability to adhere to SLAs.

B. Stages of Risk Assessment in the Use Cases

Taking into consideration the Cloud Service Lifecycle in the context of the Risk Assessment Tools, assessment will be performed at many stages and will be reliant on the specific use case. Figure 5 depicts the general view of risk assessment in all the different use cases in OPTIMIS. The risk assessment stages will be dependent on the use cases being represented. The different use cases will influence the different actors allowing similar risk assessment between them. In the case of the private cloud, the actors involved were the Service Provider and the Infrastructure Provider (as shown in Figure 5). In the cases of Cloud bursting, federated and multi-Cloud, this will allow further actors to be involved depicting infrastructure provider and infrastructure provider interactions.

There are six action stages which are dependent on the interaction of an SP and IP and what tasks it is performing and will dictate what risk models and input data are utilized in the assessment.

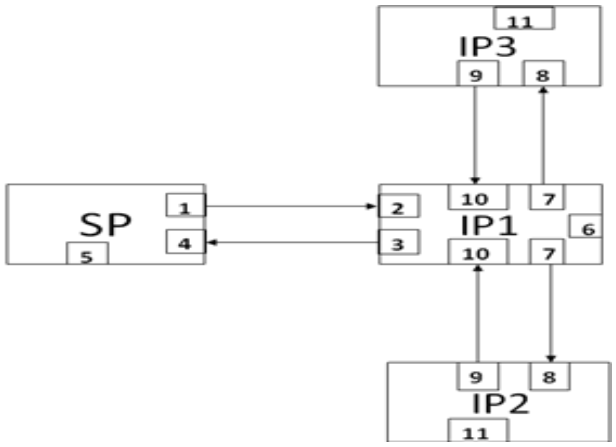


Figure 5: Risk Assessment Steps 1-11 in the Different Use Cases.

The six action stages are as follows:

- Action 1: The sender, before sending an SLA request to an IP, assesses the risk of dealing with all known IPs.
- Action 2: An IP receives an SLA request and assesses the risk of dealing with the SP from which the request came from.
- Action 3: The IP assesses the risk of the SLA from the sender and evaluates the risk associated with the service manifest.
- Action 4: The sender then receives the IPs SLA offer and assesses the risk associated against other IP SLA offers.
- Action 5: The sender performs continual risk assessment at Service Operation, monitoring service level non-functional QoS metrics such as response time.
- Action 6: The receivers perform continual risk assessment at Service Operation, monitoring low level events from the infrastructure such as risk of VM failure.

For the private cloud the 6 stages in Figure 5 will be from steps 1-6 in which each of the 6 actions take place. The order in which each of the action stages is (Step 1-Action 1), (Step 2 -Action 2), (Step 3-Action 3), (Step 4-Action 4), (Step 5-Action 5) and (Step 6-Action 6).

In Cloud Bursting use case four further stages of risk assessment occur between the IP1 and IP2 that replicate the risk assessment performed by the SP in the Private Cloud use case, where by IP1 takes on the negotiation roles of the SP to facilitate the acquisition of additional resources. The additional number of action stages is (Step 7-Action 1), (Step 8 -Action 2), (Step 9-Action 3), (Step 10-Action 4), (Step 11-Action 6).

In the Federated Cloud use case, due to the collaborative nature of the IPs and the assumed prior SLA between the members of the federation, this use case is a simplification of Cloud Bursting with the exception that any number of IPs

 ISBN: 978-1-61208-153-3

can be burst to and a single IP resumes the role of being the point of entry into and controller of the federations. This means no risk assessment is necessary in regards to risk assessment steps 7 to 10 of the Cloud Bursting use case. Therefore there are only Steps 1-6 with an additional Step 11.

Finally, in the Multi-Cloud use case the missing steps of risk assessment in the Federated Cloud use case are necessary as IP1 is required to select and negotiate with several IPs. Therefore it will use all the steps from Step 1-11 for its risk assessment in multi-cloud scenario.

VI. CONCLUSION AND FUTURE WORK

This paper presents various methodologies being designed and developed for performing risk assessment on both SP and IP levels. The main contributions of the work are the design and implementation of an effective and efficient risk assessment framework (methodologies of risk identification, evaluation, mitigation and monitoring) for Cloud service provision. Four risk categories, namely legal, technical, policy, and general have already been identified. SP and IP risk models are being investigated in conjunction with a risk inventory for Cloud computing specific to OPTIMIS through various use cases: private cloud, cloud bursting, multi-clouds, federated cloud, and cloud brokerage. This inventory is populated with Assets, Incidents/Risk Scenarios and Impact/Consequences, as well as associated mitigation strategies. The novel risk assessment models will be built and developed as a combination of probabilistic, possibilistic and hybrid models to suit each risk category identified in the risk inventory.

ACKNOWLEDGMENT

This work has been partially supported by the EU within the 7th Framework Programme under contract ICT-257115 - Optimized Infrastructure Services (OPTIMIS).

REFERENCES

[1] A. Morali and R. J. Wieringa, Risk-Based Confidentiality Requirements Specification for Outsourced IT Systems, pp. 199-208, Proceedings of the 18th IEEE International Requirements Engineering Conference, 2010, DOI 10.1109/RE.2010.30.

[2] A. Morali and R. J. Wieringa, Risk-Based Confidentiality Requirements Specification for Outsourced IT Systems (ex-tended version), Technical Report TR-CTIT-10-09, Centre for Telematics and Information Technology, University of Twente, 2010.

[3] C. S. Yeo and R. Buyya, Integrated Risk Analysis for a Commercial Computing Service in Utility Computing, Journal of Grid Computing, Volume 7, Number 1, pp. 1-24, ISSN: 1570-7873, Springer, Germany, March 2009.

[4] M. Luo, L. J. Zhang, and F. Lei, An Insurance Model for Guaranteeing Service Assurance, Integrity and QoS in Cloud Computing, pp. 584-591, Proceedings of the 2010 IEEE International Conference on Web Services, DOI 10.1109/ICWS.2010.113.

[5] J. O. Fitó, M. Maćas, and J. Guitart, Towards Business-driven Risk Management for Cloud Computing, pp. 238-241, Proceedings of the 2010 International Conference on Network and Service Management – CNSM 2010.

[6] A. J. Ferrer, F. Hernandez, J. Tordsson, E. Elmroth, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgo, T. Sharif, and C. Sheridan, OPTIMIS: a Holistic Approach to Cloud Service Provisioning, in the Proceedings of the 1st International Conference on Utility and Cloud Computing (UCC 2010), Chennai, India, December 2010.

[7] K. Djemame, I. Gourlay, J. Padgett, K. Voss, and O. Kao, Risk Management in Grids, In R. Buyya and K. Bubendorfer, editors, Market-Oriented Grid and Utility Computing, pp. 335–353. Wiley, 2009.

[8] R. Alsoghayer and K. Djemame, Probabilistic Risk Assessment for Resource Provision in Grids, pp. 99-110, in the Proceedings of the 25th UK Performance Engineering Workshop, Leeds, UK, July 2009.

[9] J. A. Zachman, "A Framework for Information Systems Architecture", IBM SYSTEMS JOURNAL, VOL 26. NO 3, 1987.

[10] J. Heaney, D. Hybertson, A. Reedy, S. Chapin, T. Bollinger, D. Williams, and M. Kirwan, Jr. Information Assurance for Enterprise Engineering, in Proceedings of PLoP, Monticello, Illinois, 8-12 September 2002.

[11] G. Stoneburner, A. Goguen, and A. Feringa, Risk Management guide for Information Technology Systems, NIST Special Publication 8-00-30

[12] Nagios <http://www.nagios.org> 30.06.2011

[13] P. Saripalli and B. Walters, QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security , pp. 280-288, In the Proceedings of the IEEE 3rd International Conference on Cloud Computing, 2010

[14] X. Zhang, N. Wuwong., H. Li, and X. J. Zhang, Information Security Risk Management Framework for the Cloud Computing Environments, pp. 1328-1334, in the Proceedings of the 10th IEEE International Conference on Computer and Information Technology, 2010 (CIT 2010)

[15] IDC Cloud Computing Survey <http://blogs.idc.com/ie/?p=210> 30.06.2011

[16] ENSIA Report on Cloud Computing Security Risk Assessment <http://www.enisa.europa.eu/act/rm/files/deliverables/Cloud-computing-risk-assessment> 30.06.2011

[17] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, Above the Clouds: A Berkeley View of Cloud Computing, Technical Report. University of California at Berkeley, 2009.

[18] H. A. Linstone, The Delphi Method: Techniques and Applications. Addison-Wesley, 1975.

[19] K. Djemame, J. Padgett, I. Gourlay, and D. Armstrong, Brokering of Risk-Aware Service Level Agreements in Grids, Concurrency and Computation: Practice and Experience, 2011.

[20] The Risk Management Standard, Institute of Risk Management, The Association of Insurance and Risk Managers, National Forum for Risk Management in the Public Sector, Volume 2008, 21st August, 2002.

[21] Optimis Consortium, Architecture Design, WP 1.1: Requirements Elicitation, 2010 <http://www.optimis-project.eu/publications> 30.06.2011.

[22] Optimis Consortium, Architecture Design, WP 1.2: Reference Architecture, 2010 <http://www.optimis-project.eu/publications> 30.06.2011.

[23] C. Carlsson, Risk Assessment for Grid Computing with Predictive Probabilities and Possibilistic Models, in proceedings of the 5th International Workshop on Preferences and Decisions, Trento, Italy, April, 2000.

 ISBN: 978-1-61208-153-3

A Linear Programming Approach for Optimizing Workload Distribution in a Cloud

Vadym Borovskiy, Johannes Wust, Christian Schwarz, Alexander Zeier
Hasso-Plattner-Institut, Potsdam, Germany
{*vadym.borovskiy, johannes.wust, christian.schwarz, alexander.zeier*}*@hpi.uni-potsdam.de*

Wolfgang Koch
SAP AG, Walldorf, Germany
wolfgang.koch@sap.com

***Abstract*—Cloud computing's usage-based pricing model creates an incentive for subscribers to optimize the utilization of the rented resources. The goal of the current work is to devise a formal approach for distributing workload among a minimum number of servers. The paper models this problem as a set partitioning problem and describes two solution approaches. The first one generates a set of candidate blocks and then composes an optimal partition by solving an integer programming problem. The second approach solves the set partitioning problem with column generation technique. Both methods were implemented and evaluated. The experiment results led to a conclusion that the second approach delivers the best results.**

***Keywords*-Workload distribution; Set partitioning; Column generation**

I. Introduction

Cloud computing continues to gain momentum due to its ability to provide on-demand computing resources in both an economically and computationally efficient manner. The economic benefit of cloud computing from a provider's point of view comes from economies of scale [1]. The more resources a data center has, the lower the cost of individual resource. From a resource consumer's point of view, the benefit derives from converting the fixed cost of owning and maintaining on-premise infrastructure into the variable cost of renting it on demand. On average, on-premise infrastructure is underutilized, because its capacity is driven by the system's peak load. But peak loads only account for a small part of a systems' operating time. Companies make large investments in their infrastructure only to find it idle for a majority of the time. By subscribing to cloud services companies pay only for the resources they actually use, whereas with on-premise hardware, the amount of resources they pay for is driven by peak workload [2].

Cloud computing's usage-based pricing model creates an incentive for subscribers to optimize the utilization of the rented resources. This is especially relevant for multi-module systems, because of many possible deployment options. Selecting a particular number of servers and the distribution of the system's modules among the servers produces visible effects. If few modules are installed on each server, the overall number of servers is bigger than absolutely necessary, which implies extra cost. On the other hand, when too many modules are deployed on each server, bottlenecks appear, which implies lower throughput and lower quality of service. Thus, by choosing a proper deployment configuration subscribers can [1]: (i) avoid resource over-provisioning; (ii) maintain the desired quality of service in the face of increasing workload by provisioning on-the-fly additional resources.

The paper is structured as follows. Section III presents a formal model of the workload distribution problem. Section IV describes a straightforward solution procedure based on the suggested model. Section V describes how a column generation technique can improve the solution procedure. Section VI presents computational results of the suggested algorithms. Section VII concludes the paper.

II. Related Work

Even though load balancing has received much attention in the research community [3], [4], [5], [6], no conventional techniques can be applied to the discussed problem. A fundamental obstacle limiting the applicability of the conventional techniques is the violation of the requirement that any server in a cluster can handle any request coming from any client (i.e., the servers must be interchangeable). In our case servers are not interchangeble, because they perform different tasks (i.e., run different modules). The lack of existing approaches motivated us to apply knowledge from other areas. In particular, our work has been inspired by research in airline crew scheduling [7], [8], where the set partitioning approach has been successfully applied for resource allocation problems. With regards, to column generation as a method of dealing with large linear programs, many researchers have observed that it is a very powerful technique for solving a wide range of industrial problems to an optimum or to a near optimum. Ford and Fulkerson, for example, suggested column generation in the context of a multi-commodity network flow problem [9]. Gilmore and Gomory then demonstrated its effectiveness in a cutting stock problem [10]. More recently, vehicle routing, crew scheduling, and other integer-constrained problems were successfully solved with column generation [11].

III. Mathematical Model of the Problem

The goal of the current work is to devise a formal approach for distributing modules of a system among a number of servers. Given the workload of each module, we want to assign it to one of the available servers with given

 ISBN: 978-1-61208-153-3

capacity. The workload of a module and the capacity of a server must be measured in the same units that represent the amount of a resource consumed or provided. The resource can be, for example, CPU time, memory, storage or network bandwidth. Measuring the exact workload of a module may be impossible, due to its dynamic nature. However, we believe that with the help of profiling tools, a reasonably precise workload estimate is feasible to obtain.

A simpler way of figuring out the workload of a module is to measure it relatively to the capacity of a server. For that, install multiple instances of a module on the same server as long as the service level of each module satisfies requirements. If a server can handle four instances of a module, then the module's workload is 25% of the server's capacity. Assuming server capacity is 100, the workload of an item is 25. If the workload of modules changes significantly over time, then no distribution can be optimal for a long time. In this case the workload distribution procedure must be carried out more frequently.

In set theory terms the workload distribution problem is stated as follows: divide a set into one or more disjoint subsets called blocks. This problem is called set partitioning and is well-known in computer science [12]. Through partitioning a set of workload items and assigning each block of a partition to one processing unit the workload distribution is carried out. The following example demonstrates the idea. Suppose there are four workload items, denoted as $w_i, i = \overline{1..4}$. In order to distribute them among processing units, a partition of the set $W = \{w_1, w_2, w_3, w_4\}$ must be generated. The set W can be partitioned in 15 different ways.

$$\begin{array}{lll} w_1w_2w_3w_4, & w_1w_2w_3|w_4, & w_1w_2w_4|w_3, \\ w_1w_2|w_3w_4, & w_1w_2|w_3|w_4, & w_1w_3w_4|w_2, \\ w_1w_3|w_2w_4, & w_1w_3|w_2|w_4, & w_1w_4|w_2w_3, \\ w_1|w_2w_3w_4, & w_1|w_2w_3|w_4, & w_1w_4|w_2|w_3, \\ w_1|w_2w_4|w_3, & w_1|w_2|w_3w_4, & w_1|w_2|w_3|w_4 \end{array} \quad (1)$$

Each of the partitions represents a possible workload distribution. The seventh partition, for instance, consists of two blocks: w_1w_3 and w_2w_4. Therefore, the corresponding distribution will require two processing units (one per block). Multiple ways of partitioning a set create the possibility of choice and the task of finding the best partition. This leads to an optimization formulation: *Given a set of workload items find its feasible partition that has minimum number of blocks.* A partition is called feasible if the workload created by any of its blocks is less than or equal to the capacity of a processing unit. For simplicity reasons we assume all processing units have the same capacity.

The next step is the formalization of the above statement. As with any optimization problem, the set partitioning problem must have three parts: (i) Decision variables: the representation of possible partitions; (ii) Objective function: a criterion of evaluating the "quality" of a partition; (iii) Constraints: feasibility restrictions on possible partitions.

In the current work, we use the classic integer programming formulation of the set partitioning problem. The formulation assumes a two-step solution procedure:

1) Generation of a set $B = \{b_j : j = \overline{1..N}\}$ of feasible candidate blocks $b_j = \{w_l : l \in \overline{1..n}\}$, where n is the number of workload items and N is the number of candidate blocks.
2) Construction of an optimal partition out of the previously generated blocks with the help of the following integer program:

$$\sum_{j=1}^{N} x_j \to min \quad (2)$$

$$\sum_{j=1}^{N} a_{ij} \cdot x_j = 1 \quad i = \overline{1..n} \quad (3)$$

$$x_j \in \{0,1\} \quad j = \overline{1..N} \quad (4)$$

where a decision variable x_j equals 1 if the j^{th} block is included in the optimal partition and 0 otherwise; a_{ij} is an element of matrix A of size $n \times N$ and calculated as:

$$a_{ij} = \begin{cases} 1 \text{ if } w_i \in b_j \\ 0 \text{ if otherwise} \end{cases} \quad (5)$$

The objective function (2) favors partitions with the smallest number of blocks. The constraints (3) force each workload item w_i to appear only in one block of the optimal partition. For convenience these constraints can be expressed in matrix form:

$$A \cdot x = \mathbb{1} \quad (6)$$

where A is defined by the expression (5), x is the vector with N decision variables and $\mathbb{1}$ is a vector of size n with all elements equal 1.

The reason for choosing the two-step procedure and the formulation (2) – (5) of the set partitioning problem is their wide and successful application in other areas, in particular airline crew pairing and stock cutting. Research results from these areas form a solid foundation for our own effort. Subsequent sections of the paper discuss different aspects of the solution procedure that influence its computational characteristics and the quality of the solution it produces.

To illustrate the usage of this set partition problem formulation, we apply it to the example mentioned earlier. Suppose the feasible candidate blocks are

$$\begin{array}{c} w_1,\ w_2,\ w_3,\ w_4,\ w_1w_2,\ w_3w_4, \\ w_1w_3,\ w_2w_4,\ w_1w_4,\ w_2w_3 \end{array} \quad (7)$$

while the rest of the blocks present in (1) are deemed infeasible. Hence, $n = 4$ (the number of workload items) $N = 10$ (the number of blocks to be considered). Given

the feasible blocks, the following integer program can be constructed:

$$\sum_{j=1}^{10} x_j \to min$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{10} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{bmatrix}$$
$$x_j \in \{0,1\} \quad j = \overline{1..10} \tag{8}$$

The solution of the problem corresponds to the optimal partition (that has the smallest number of blocks). Given the small size of the problem, it is not difficult to see the three optimal solutions:

$$x^1_{opt} = \{0,0,0,0,1,1,0,0,0,0\} \Rightarrow \{w_1w_2\},\{w_3w_4\}$$
$$x^2_{opt} = \{0,0,0,0,0,0,1,1,0,0\} \Rightarrow \{w_1w_3\},\{w_2w_4\}$$
$$x^3_{opt} = \{0,0,0,0,0,0,0,0,1,1\} \Rightarrow \{w_1w_4\},\{w_2w_3\}$$

Note that there may be both multiple optimal solutions or no solution at all, in the case that there is at least one item with workload higher than the capacity of a processing unit.

IV. The "Full Set" Approach

As presented in the previous section the solution process starts with the generation of feasible candidate blocks. In our work we use two different methods to generate these blocks. The first one is a brute-force method that generates all possible combinations of workload items (i.e., all possible blocks). After that, the blocks must be validated against the feasibility constraints, that is, the workload of a block must not exceed the capacity of a processing unit. The total number of blocks to be considered will always be less than the number of all possible combinations of workload items:

$$N \leq \sum_{i=1}^{n} C_n^i = \sum_{i=1}^{n} \frac{n!}{i!(n-i)!} \tag{9}$$

Having generated the candidate blocks, the integer program is composed and solved. We call this method *the basic* version of the "full set" approach, because we explicitly consider all possible solutions. Section VI shows that this version is applicable only for a very small number of workload items. The reason is obvious: N, defined by the expression (9) grows very fast and the set of candidate blocks (B) quickly becomes unmanageable (either exceeds the amount of RAM or takes too much time to be processed). Nevertheless, considering all feasible combinations guarantees the best possible result. The example from Section III is solved by the "full set" approach.

In order to achieve better performance of the candidate generation, more appropriate limits for i in the expression (9) can be found. Experimenting with the "full set" approach we found that considering the blocks with too few or too many workload items is useless. Such blocks never appear in optimal partitions. Intuitively, big-size blocks are most probably infeasible, while small-size blocks increase the number of required processing units, which is not favored by criterion (2) and is therefore rejected. By considering only medium-size blocks the performance of the "full set" approach can be significantly improved. Restricting the set of candidate partitions is a very common approach used to improve the solution of the set partitioning problem. Experiments showed that in comparison with brute-forcing, the following limits produce better computational characteristics (e.g., lower memory consumption and faster processing time) without deteriorating the quality of the solution.

$$N = \sum_{i=lower}^{upper} C_n^i = \sum_{i=lower}^{upper} \frac{n!}{i!(n-i)!}$$
$$lower = \lceil \frac{1}{2} n_{avg} \rceil, \; upper = \lfloor \frac{3}{2} n_{avg} \rfloor \tag{10}$$
$$n_{avg} = \frac{Capacity}{w_{avg}} = \frac{Capacity \cdot n}{\sum_{i=1}^{n} w_i}$$

Here n_{avg} is the average number of items in a block, w_{avg} is the average workload of an item and $Capacity$ is the capacity of processing units. We call this modification "size-restricted" modification of the "full set" approach.

As one can see the expression (10) takes into account only the number of items in a block. This, however, is not the only factor that can reduce the size of the set B. Another tendency was revealed by observing the results of the conducted experiments: the workload is distributed in such a way that every processing unit is utilized to the highest possible extent. In other words, every processing unit is packed with workload items as much as possible. Based on this observation, we suggest a second way of generating set B. By sequentially iterating through the set of workload items, we select items as long as the total selected workload is less than or equal to the capacity of the processing units. We call this version of the "full set" approach *load-restricted*. The listing below presents the details of the algorithm.

```
generate (items, eps, capacity) {
  iter = items.begin();
  while (items.size() > 0) {
    s1 = s2 = count = 0;
    //select items to a new block
    while (count < items.size() && items.size() > 0) {
      s2 = s1 + iter->value;
      if (s2 < capacity - eps) {
        //add the workload item to the block and continue
        s1 = s2; count = 0;
        block.add(iter.value);
        items.remove(iter); iter++;
      }
      else if (s2 <= capacity) {
        //add the item and stop selecting more items
        block.add(iter.value);
        items.remove(iter); iter++;
        if (iter == items.end())
```

```
          iter = items.begin();
        break;
      }
      else {
        //skip the item and continue
        count++;
        item++;
      }
      if (iter == items.end())
        iter = items.begin();
    }
    B.add(block);
    block.clear();
  }
  return B;
}
```

At the end of the procedure, the set B contains a number of blocks constituting a feasible partition of a given set. By running the $generate$ procedure multiple times and shuffling the set of items before each run a required number of candidate blocks (i.e., the set B) can be generated.

The run-time complexity of the both versions of the "full set" algorithm is determined by the algorithm used for solving the integer problem (2)-(4). We used the branch-and-bound algorithm. Its complexity is exponential [13]. Hence, the complexity of the "full set" algorithms is also exponential, $O(2^N)$, where N is the number of candidate blocks.

V. The "Column Generation" Approach

Two factors make the "full set" approach impractical. The first one is the size of N, which can be enormous, even when n is still reasonable (say, less than 10000). The second factor is the integrality constraint (4). Solving large-scale integer programming problems is not a trivial exercise, and requires more a complex solution procedure in comparison to linear programming problems. These two factors significantly limit the applicability of the approach.

This section shows how these difficulties can be overcome by a method suggested in [14]. The main idea is to enhance the "pricing out" stage of the simplex method. At this stage, Danzig and Wolfe [14] suggest *generating* a useful *column* by solving an auxiliary integer programming problem instead of looking over a vast existing collection of columns to pick out a useful one.

Put simply, column generation means beginning with a manageable part of a linear optimization problem, solving that subproblem, and then discovering the way of improving the solution by extending the subproblem with the parts of the original problem. This process is repeated until a satisfactory solution to the original problem is achieved [15]. In formal terms, column generation is a modification to the simplex method that adds columns corresponding to constrained variables during the pricing phase [13].

Column generation relies on the fact that in the simplex method, the solver does not need access all the variables of the problem simultaneously. In fact, a solver can begin working with only the basis (a particular subset of the constrained variables) and then use reduced cost to decide which other variables are needed [16].

To solve a set partition problem by column generation we start with a subproblem, called the master problem. That is, we choose several feasible blocks and solve the problem (2) – (5) for them. This will surely work in that it produces some answer (a feasible solution) to the problem, but it will not necessarily produce a satisfactory answer. To move closer to a satisfactory solution, we can then generate other columns. Other decision variables (other x_j) will be chosen to add to the model. Those decision variables are chosen on the basis of their favorable reduced cost with the help of a subproblem. This subproblem is defined to identify the coefficients of a new column of the master problem with minimal reduced cost.

Let π be the vector of the dual variables of the current solution to the master problem. The subproblem is then defined as follows:

$$1 - \sum_{i=1}^{n} \pi_i c_i \rightarrow min \tag{11}$$

$$\sum_{i=1}^{n} w_i c_i \leq capacity \tag{12}$$

$$c_i \in \{0,1\} \quad i = \overline{1..n} \tag{13}$$

The solution to the problem (11) – (13), vector c_{opt}, represents the coefficients of a new column of the constraint matrix A of the master problem. Adding a new decision variable (i.e., a new candidate block) to the master problem with the constraints coefficients c_{opt} will result in the best possible improvement of its solution. In this way, instead of explicitly considering a fixed set of columns (candidate blocks), we generate and add new ones to the master problem only if they improve its solution. This avoids the need of explicitly enumerating candidate blocks.

Having discussed all necessary aspects of column generation we present the *basic* five-step algorithm of solving the set partitioning problem with column generation.

- **S1** Compose the master problem (2) – (5) for a limited set of candidate blocks. The simplest way to generate this set is to place each of the n workload items to a separate block. In this case the matrix A is a diagonal matrix: elements of the main diagonal equal 1, while non-diagonal ones 0.
- **S2** Solve the master problem.
- **S3** Given the optimal dual solution of the master problem, compose the auxiliary column generation problem (11) – (13).
- **S4** Generate a new column (i.e., a new candidate block) by solving the auxiliary problem.
- **S5** Add the new column to the master problem and return to the step **S2**. Repeat the procedure until the improvement of the master problem solution becomes negligible.

 ISBN: 978-1-61208-153-3

Having experimented with the algorithm, we observed an even stronger tendency to pack blocks with as many items as possible. In comparison to the blocks generated by the "full set" algorithms, column generation produced blocks with workload closer to the capacity of a processing unit. This tendency is true for every n, large or small. This fact, in turn, created a hypothesis that if the correlation between the block fulfillment and n is weak, we can split a sufficiently large set of workload items into a number of smaller subsets of the size k, $k \leq n$, and run the algorithm on each subset independently without deterioration of the quality of the overall solution. The prime reason for this is decreasing the execution time. In linear and integer programming the solution time increases non-linearly with the size of a problem. This implies that solving two problems with 50 variables takes less time than solving one problem with 100 variables. In order to check this hypothesis, we modified the basic column generation algorithm accordingly. The resulting version of the algorithm is called *parallel*. As one can see from the experiment results, the hypothesis proved to be true and allowed a very efficient algorithm.

The run-time complexity of both versions of the "column generation" algorithm is determined by the complexity of the simplex method, which is exponential [13]. Thus, the complexity of the basic version is $O(2^N)$, where N is the number of considered blocks and corresponds to the number of generated columns. The complexity of the parallel version slightly differs and is $\frac{n \cdot O(2^k)}{k}$, where n is the number workload items to be distributed and k is described above. In our experiments, we took $k = 50$. One can now clearly see that the expected speed-up of the parallel version equals to $O(2^N) - \frac{n \cdot O(2^k)}{k}$.

VI. Computation Results

This work contributes, in total, five workload distribution algorithms: three versions of the "full set" algorithm, and two versions of the column generation algorithm. In order to validate them a number of experiments were conducted. This section describes the set up of the experiments and reports their results.

All suggested algorithms were implemented in C++ and used the IBM ILOG CPLEX V12.1 optimization engine in order to solve the linear programming problems. All algorithms were run on a Quad-core Intel Xeon E5450 3.00GHz machine with 8 GB of RAM. The experiments were conducted as follows. First, n workload items were obtained with a random number generator. In the experiments we used random number equally distributed in the range from 12 to 40. The capacity of a processing unit has been fixed at 100 in all experiments. Second, each of the five algorithms was run on the generated set of workload items and the execution time was measured. Table I contains the results of the conducted experiments. In addition to the execution time, the obtained solution (the number of required processing units) is presented. For the "full set" algorithms we also report N - the number of candidate blocks considered. For the column generation algorithms the number of generated columns is reported. Because basic and size-restricted versions of the "full set" algorithm fail to distribute more than 15 items, the statistics on them are not included in the table.

For example, during the ninth experiment 600 workload items were generated. The basic and size-restricted versions of the "full set" algorithm failed due to size of the set B. The load-restricted version distributed the items among 162 processing units, but took 4 minutes to complete, and processed 86390 candidate blocks. The basic version of the column generation algorithm distributed the same workload items among 155 processing units. The algorithm generated 1541 columns and took 6 minutes while the parallel version of the algorithm was able to achieve the same result in only 24 seconds.

VII. Conclusion

Cloud computing's pricing model creates an incentive for subscribers to minimize the consumption of rented resources. In the case of modularized software, multiple deployment options may exist, creating different possibible distributions of workload and resource consumption. The current research aims to developing a formal approach of distributing multiple workload items among a minimum number of processing units.

We designed and evaluated five algorithms that, given a set of workload items, distribute them among processing units of specified capacity. The algorithms can be classified into two different types: those that explicitly consider a fixed set of candidate options (the "full set" algorithms) and those that gradually improve the solution by considering dynamically generated options (column generation algorithms).

The combinatorial nature of the workload distribution problem makes any algorithm based on explicit enumeration of possible alternatives intractable. That is, even for reasonably sized input, the algorithms fail due to the overwhelming number of alternatives to be processed. The experimental results clearly demonstrated this phenomenon.

The results also showed that the basic version of the column generation algorithm produces the best solution. The solution found by the parallel version of the column generation algorithm is worse by approximately 1%. However, the speed of parallel version is much better. For this reason, we conclude that the best results are achieved with parallel version of the column generation algorithm.

Acknowledgments

The authors want to express special thanks to Nick Lanham for the numerous improvements he contributed to this paper.

 ISBN: 978-1-61208-153-3

Table I
EXPERIMENT RESULTS

		Full Set			Column Generation					
		Load-restr.			Basic			Parallel		
No	n	Sol.	N	Time	Sol.	Cols.	Time	Sol.	Cols.	Time
1	10	3	100	1 sec	3	15	1 sec	3	15	1 sec
2	15	4	137	1 sec	4	23	1 sec	4	23	1 sec
3	30	9	300	1 sec	8	25	1 sec	8	25	1 sec
4	50	14	473	2 sec	13	121	2 sec	13	121	2 sec
5	100	28	793	3 sec	26	233	6 sec	26	243	3 sec
6	150	40	2870	15 sec	39	322	21 sec	39	348	5 sec
7	250	71	8800	36 sec	69	672	45 sec	70	1523	11 sec
8	400	108	20000	1 min	104	912	3 min	104	938	17 sec
9	600	162	86390	4 min	155	1541	6 min	156	1523	24 sec
10	1000	275	133000	7 min	262	2637	22 min	263	2654	40 sec
11	1300	349	219240	19 min	330	3674	42 min	333	3243	60 sec
12	1500	405	303750	27 min	389	3853	93 min	390	3789	75 sec
13	2000	536	549000	31 min	514	5201	168 min	516	5017	84 sec
14	2500	670	846250	39 min	651	6435	274 min	654	6337	95 sec
15	3000	807	1200000	45 min	-	-	-	774	7519	112 sec
16	5000	1342	3365000	74 min	-	-	-	1296	12595	200 sec
17	10000	2698	12146396	193 min	-	-	-	2598	25170	378 sec

		Full Set					
		Basic			Size-restr.		
No	n	Sol.	N	Time	Sol.	N	Time
1	10	3	35673	5 sec	3	27990	3 sec
2	15	4	3012765	12 min	4	2366910	9 min
3	30	-	-	-	-	-	-

REFERENCES

[1] P. Murray, "Enterprise grade cloud computing," in *Proceedings of the Third Workshop on Dependable Distributed Data Management, European Conference on Computer Systems*, 2009, pp. 1–1.

[2] R. L. Grossman, "The case for cloud computing," *IT Professional*, pp. 23–27, March 2008.

[3] W. Tang and M. W. Mutka, "Load distribution via static scheduling and client redirection for replicated web servers," in *International Conference on Parallel Processing*, 2000.

[4] N. Nehra, R. B. Patel, and V. K. Bhat, "A framework for distributed dynamic load balancing in heterogeneous cluster," 2007.

[5] D. Grosu and A. T. Chronopoulos, "A game-theoretic model and algorithm for load balancing in distributed systems," in *16th International Parallel and Distributed Processing Symposium*, 2002, pp. 146–153.

[6] S. Iqbal and G. F. Carey, "Performance analysis of dynamic load balancing algorithms with variable number of processors," *Journal of Parallel and Distributed Computing*, vol. 65, pp. 934–948, 2005.

[7] J. Arabeyre, J. Fearnley, F. Steiger, and W. Teather, "The airline crew scheduling problem: A survey." *Transportation Science*, vol. 3, no. 2, p. 140, 1969.

[8] R. E. Marsten and F. Shepardson, "Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications," *Networks*, vol. 11, no. 2, pp. 165–177, 1981.

[9] J. Ford, L. R. and D. R. Fulkerson, "A suggested computation for maximal multi-commodity network flows," *MANAGEMENT SCIENCE*, vol. 5, no. 1, pp. 97–101, 1958.

[10] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961.

[11] M. Minoux, "Column generation techniques in combinatorial optimization: A new approach to the crew pairing problems," in *24th AGIFORS Symposium*, 1984, pp. 15–29.

[12] D. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 3*. Addison-Wesley, 2005.

[13] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*. Springer, 2005.

[14] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *OPERATIONS RESEARCH*, vol. 8, no. 1, pp. 101–111, 1960.

[15] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, pp. 1007–1023, November 2005.

[16] D. Feillet, "A tutorial on column generation and branch-and-price for vehicle routing problems," *4OR: A Quarterly Journal of Operations Research*, vol. 8, pp. 407–424, 2010.

 ISBN: 978-1-61208-153-3

Chaavi: A Privacy Preserving architecture for Webmail Systems

Karthick Ramachandran, Hanan Lutfiyya and Mark Perry

Department of Computer Science
University of Western Ontario
London, Ontario, Canada
Email: {kramach, hanan, markp}@csd.uwo.ca

***Abstract*—The last two decades have seen major innovations in the Internet and transformation of the way people do business, communicate and live. Concomitant with the Internet bringing the advantages of new services, is a growing awareness of threats to Privacy that the Internet can enable. When considered in this context, the Cloud Computing paradigm requires users forgive disturbing levels of trust by users in the servers that hold their information. There is a pressing need for innovative architectures to allow the user to rely on the server with little or no need for trust in the service provider. In this work, we give an introduction of privacy issues in Cloud Computing and discuss the state of art in the privacy enhancing technologies that can be used for Cloud Computing. We focus on webmail services and propose a privacy preserving architecture in which users can retain their mail in the servers of their service providers in a cloud without compromising functionality or privacy. We benchmark our system and present the results showing that it is feasible to architect a privacy preserving solution for webmail systems.**

Keywords-privacy-preserving; webmail; encrypted search.

I. INTRODUCTION

Cloud Computing is a model of computing in which the users can rent infrastructure, platform or software services from other vendors without requiring the physical access to the rented service [18]. There are three main types of cloud offerings: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS offers virtualized instances of bare machines leaving the installation and customization of softwares including the Operating System to cloud computing customers. In PaaS, an application framework is provided to the customers for developers to develop their software with. A SaaS provider offers a particular application as a web service, which customers can customize to their needs. The Cloud Service Provider (CSP) focuses on infrastructure and software expertise and aims to optimize their utility by providing centralized services for one or many clients. The benefit to the cloud service client (CSC) is that the cost associated with the underlying infrastructure and software services needed to support the CSC's application is reduced. There are two reasons for the cost reduction. One reason is that the underlying infrastructure and software services are shared among CSCs. The second reason is that since a CSP manages data, it can use creative business models like Contextual Advertising Model [16] for generating revenue by delivering advertisements to users based on the data. For example, webmail services such as Google can provide Gmail for free. As a result, Cloud Computing has been widely adopted. MarketsandMarkets [17] estimates that the cloud computing global market will increase from $12.1 billion (US) to $37.8 billion (US) in 2015 at a compound annual growth rate of 26.2 percent.

In spite of this widespread adoption, organizations are still wary of storing their sensitive data with a CSP. Privacy risk remains a major concern in the cloud computing environment [11].

The definition of privacy that we use was defined by Warren et al. [23] in 1890. Warren et al. described privacy as the "right to be let alone" with the focus on protecting individuals and is recognized in Convention for the Protection of Human Rights and Fundamental Freedoms. There are a variety of ways that the privacy of data can be compromised in a cloud service environment [4]. This includes the following:

1) Sharing of data with an unauthorized party: The Cloud provider could compromise the confidentiality of the data by sharing the data that it stores with unauthorized parties. This can go against the terms and conditions of the service and will qualify as a breach of security and contract. The end user may never be aware of such a breach.

2) Corruption of data stored: The Cloud Computing provider's root access to physical machines allows the Cloud Provider to have access that allows the Cloud Provider to modify/delete data. The Cloud Provider could tamper with the data making the data non-usable or modify the data in a way that system cannot detect the modification. This poses a serious threat to the integrity of the application.

3) Malicious Internal Users: The employee of a Cloud Computing Provider who has root access to these physical machines, could access the data and use it for their own advantage.

4) Data Loss or Leakage: When a virtual machine is used in an infrastructure, it poses a variety of security issues [10] which could lead to a compromise of the data. Moreover, when the facility that hosts the user's data is subjected to a natural calamity, it could risk the loss of the user's data.

 ISBN: 978-1-61208-153-3

5) Account or Service Hijacking: Another risk for the Cloud Computing provider is, if the service is hijacked, or the computer is hacked into by an intruder, the hacker will have access to data.

This work focuses on the following threats: (a) Sharing with an unauthorized party, b) Malicious internal users, and c) Account or service hijacking. Our work applies to the class of cloud services that stores data and provide searching as its primary functionality. This includes services such as webmail, collaborative document authoring (Google documents) and private blogs. The example used throughout this paper is webmail.

We proposed Chaavi, a webmail infrastructure that builds on the public/private key model to encrypt email with a custom implementation of encrypted indices for keyword searches using the server's infrastructure. Chaavi is the first system that addresses the above threats in a real working environment.

The rest of paper is organized as following. A motivating example of webmail services is described in Section II. Section III presents some of state of the art in preserving privacy for cloud computing services. Section IV reviews background and related work for searching on encrypted data. Section V presents the architecture of Chaavi system. The implementation details are discussed in Section VI. Section VII presents the experiments conducted to study the system and we conclude by stating our contribution and future work in Section VIII.

II. Motivating Example: Webmail Services

Webmail services offer user convenience. With a username, password, and Internet access users, are not tied to any particular equipment or location. Webmail services primarily offer the following functionality:

1) Mail Storage
2) Organization of mail
3) Keyword Searching

For (1) and (2), the service provider need not know the exact content of the mail. However, for performing a plain-text keyword search on email the user needs the service provider to know the content of the mail, so that the cloud provider's infrastructure can be used to index the mail content, which can in turn be used for the search process.

The usage of webmail services, has the following shortcomings:

1) The need to trust the service provider (e.g., Google, Yahoo, or Microsoft) as the mail is stored as plain-text in the service providers' servers (or using single key encryption). The mail is then prone to insider attacks (anyone with the access control will be able to read the mails).
2) There is an assumption that the provider is honest, and the security level is sufficient.
3) When the mail is transferred from one domain to another, it is transmitted through SMTP [19]. SMTP as a protocol does not support encryption. Technologies like Transport Layer Security [9] are used to transfer mail to other domains. However, the data is still protected only up to the layer at which it reaches the target mail server. Once it reaches the target mail server, the mail is again prone to insider attacks in the new domain.

To address such problems, various client encryption systems, such as Pretty Good Privacy (PGP) [26], have been developed. However, encryption using PGP make the mail non-searchable in the web server.

III. Related Work

Privacy Enhancing Technologies (PET) can be used by the developers of the application to enhance the individuals privacy in an application development environment. In this section, we survey state of the art in PET.

Homomorphic Functions: Homomorphic encryption schemes refer to asymmetric encryption techniques, where algebraic operations on plain text can be performed directly on a respective cipher text. This was first introduced by Goldwasser et al. [12], where the authors performed modular addition of two bits using multiplication of ciphertexts (Quadratic Residuosity Problem). The best result so far is a scheme by Boneh et al. [7], where additions are freely performed on encrypted domain. This still remains in the theoretical realm as more advanced abstractions need to be created for using homomorphic functions in practical applications.

Privacy By Secure Computation: The objective of secure computation is to evaluate a function f that takes inputs from two parties A and B without revealing the exact inputs to each other. The Yaos protocol [25] provides some of the basic techniques to perform a computation in a secure way without revealing the inputs. The Yaos protocol forces the expression of a computation problem in terms of logical circuit using gates. The input of each gate is randomly encrypted and then the final resulting output is decrypted to get the exact answer of the computation. The encryption and the decryption is done at the client's end. The expression of a simple problem using the Yaos protocol is found to be complex. Applications that typically reside in the cloud (e.g., mail) are too complex for this.

Privacy By Using Secure CoProcessors: Secure co-processors are currently the only realistic way to perform general-computing even when an adversary has direct physical access to the server. In our case the adversary could be the cloud service provider itself. It is a very limited computer with ROM, RAM and battery backup for persistent storage and an ethernet card. When installed in a computer, co-processors can be seen as a secure area inside a computer, which even the main processor cannot access. Privacy as a Service [13] recognizes these factors and proposes a system architecture in which a coprocessor is installed in every Cloud Computing system. The data loaded into the

 ISBN: 978-1-61208-153-3

cloud is classified based on its significance and security by the cloud user (No Privacy, Privacy with Trusted Provider, Privacy with Non-Trusted Provider). The data tagged with Privacy with Non-Trusted Provider level is processed by the secure co processor. Secure co-processors needs a separate hardware installation in each server. Also co-processors are expensive and are not yet economical to be used in a cloud computing environment.

Privacy By Encryption: Privacy can be enforced by encrypting all the data that is stored in the cloud. The main issue is that the cloud can be only used for storage of the data. As the data will be unrecognizable to the cloud service provider, it will not be possible for the cloud service provider to process the data nor to perform some number crunching tasks. Searchable encryption uses an algorithm which allows users to encrypt the data and then provides the server with trapdoor information [6], so that the server can search for a given string through the searchable encryption algorithm. This part is discussed in detail in Section IV-C.

Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data [8] proposes a new encryption scheme for keyword search over encrypted data in cloud computing environment with privacy and performance requirements.

In our work we achieve privacy by encryption by using searchable encryption scheme for a webmail software. Our focus is to study how this the encryption schemes can be engineered in a real working environment.

IV. Background

In this section, we review the basic elements common to webmail infrastructures. We also present an introduction to PGP and searchable encryption.

A. Mail Architecture

The webmail infrastructure is responsible for end to end delivery of email. Figure 1 presents architectural components and protocols typically used to support webmail applications.

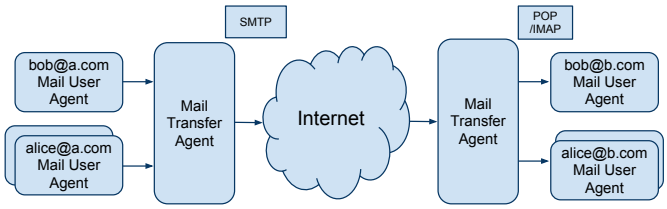


Figure 1. Email Architecture

1) Components: This subsection describes the architectural components.

Mail User Agent: The Mail User Agent (MUA) is used to manage a user's email. It acts on behalf of the user to send and receive mail from the Mail Transfer Agent (MTA). Popular MUAs include Microsoft Outlook, Mozilla Thunderbird, Apple Mail. In a webmail system, the MUA runs in the server and the pages are rendered as HTML pages for the browser.

Mail Transfer Agent: The Mail Transfer Agent (MTA) transfers messages from one server to another. It receives email either from another MTA or MUA. The transmission of email follows standardized protocols for message transfers.

2) Protocols: This subsection describes commonly used protocols.

Simple Mail Transfer Protocol (SMTP): SMTP refers to the standard for the transfer of messages from one server to another. It is used by MUA to relay mail through MTA and it is also used by MTA to send and receive mail between other MTAs. SMTP as a standard does not encrypt messages (unless Transport Layer Security encryption is used).

Post Office Protocol (POP) / Internet Mail Access Protocol (IMAP): POP/IMAP are email retrieval protocols that specify standards for downloading messages from the MTA for MUA. Examples of use is found with support for POP version 3 and IMAP as provided by Gmail.

3) Privacy Threats: In webmail systems, there is a server for webmail introduced into the standard mail system (Figure 1). It acts as the Mail User Agent for a number of users and manages email for all the users. The MUA, unlike the standard model (Figure 1), is centralized at the server. The webmail server uses POP/IMAP to download messages from MTA.

There are several privacy concerns with respect to email systems. If the connection to the webmail server is not secured using Hypertext Transfer Protocol Secure (HTTPS) all the data between a user's browser and the server will be in plain text. SMTP, unless used with Transport Layer Security (TLS) layer, is insecure. Even if the TLS layer is used, the mail will still be accessible by the owner of the MTA, through which the mail is routed. This is because TLS is designed to protect data in an insecure network (like Internet) and not from the communicating parties. Some of the security threats involved in email systems are identified by Kangas et al. [14], and Kaufman et al. [15]. These are detailed below.

Eavesdropping: When email is unencrypted, potential hackers who have access to network packets flowing through the network will be able to read the email sent. This can be achieved by enabling the promiscuous mode on ethernet cards.

Identity Theft: If the user's username and password is obtained, then hackers have full access to all the email content. Such password information can be obtained by eavesdropping on the network.

 ISBN: 978-1-61208-153-3

Invasion of Privacy: The recipient of the mail is able to get more information from the email header information than what the sender intends to reveal. For example, the header will reveal the sender's SMTP IP address and subject of the email sent.

Message Modification: Anyone who has administrator access to the webmail server can modify the messages stored in the server. It is not always possible for a recipient to determine that email has been tampered with.

False Messages: It is relatively easy to create false messages and send it as if it is from any person (as evidenced by spam).

Message Replay: Akin to message modification, the message created by user can be saved and sent again and again.

Unprotected Backups: Messages are stored in plain-text on SMTP servers, and backups will also contain complete copies of the messages. Even when the user deletes a message from the server, the backup will still hold the content.

Repudiation: As email messages can be forged (for example see your spam box), there is no way of validating that the email has been in-fact sent by a particular person. This has serious implications in business communications, electronic commerce.

B. Pretty Good Privacy

PGP was created by Zimmermann et al. [26], in 1991 to address the security issues with email. PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and public-key cryptography. Each public-key is bound to an email address. It serves as the verification mechanism for the origin of the email. As the email is encrypted using the private key of the user and the encrypted version is sent into the network, it addresses many security issues of the email infrastructure. For webmail systems, software such FireGPG [1] provide a browser extension that implements PGP. As PGP support enhances the security of the email system by encrypting the mails, the mail becomes unreadable by server. Hence the server cannot perform keyword searches on the mail.

C. Searchable Encrypted Data

Public Key Encryption with Keyword Search (PEKS) [6] is one of the seminal works in the area of making encrypted data searchable. The authors of PEKS propose to encrypt the message using the Public-Private key infrastructure. Along with this cipher text a Public-Key Encryption with Keyword Search (PEKS) of each keyword (the words that make up the message) is appended to the final message. To send a message M with keywords W_1, W_2, ... W_m the following information is transmitted to the server:

$$E_{A_{pub}}(M) \parallel PEKS(A_{pub}, W_1) \parallel ... \parallel PEKS(A_{pub}, W_m)$$

where A_{pub} is the public key of the user, $E_{A_{pub}}(M)$ is the encrypted message, $PEKS$ is the function that encrypts the keywords using A_{pub}. To test whether a word W is a part of the message, a user supplies $PEKS(A_{pub}, W)$ along with a trapdoor function T_w to the server, that can test whether $W = W'$ (W' being the keywords that are stored in the encrypted form in the server). If $W \neq W'$ the server learns nothing more about W'.

Public Key Encryption with Keyword Search Revisited [5] identifies some of the issues with the original PEKS and proposed a provably secure algorithm. The authors argue that if in PEKS the server starts learning the trapdoor then there can be a categorization of mail formed just based on the learned trapdoor information. The trapdoor information is the extra information sent to the server along with the encrypted keyword for the server to test for the existence of a keyword.

The authors also identify that in PEKS there is an assumption that the communication channel between the sender and the server is secure. To enable secure communication through insecure channels the authors propose a Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS), that uses a server's public-private key pair for communication.

V. ARCHITECTURE

This section describes the various components of Chaavi. Figure 2 gives the overall architecture of the system.

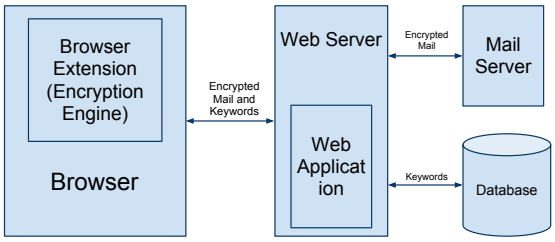


Figure 2. Chaavi - Architecture

A. Browser

The browser is responsible for rendering the pages created by the web application. Its default behavior can be modified or enhanced by using extensions in the browsers. Modern browsers such as Mozilla Firefox, Google Chrome provide functionality to write extensions and install the extensions locally.

B. Browser Extension

A browser extension is used in Chaavi to encrypt the secure message sent to the server. It is also used to decrypt the messages that are sent from the server. Additionally it has key generation and key management functionality. The extension is composed of the following modules.

 ISBN: 978-1-61208-153-3

Public-Private Key Generation: As stated earlier, Chaavi uses a public/private key model for securely communicating messages. In a public/private key model, a public-private key pair is generated when the system is initiated for the first time, for a particular user. The messages encrypted by the public key can be decrypted only by use of the private key. The public key as the name implies is shared in a public forum.

Keyword Encryption Key Generation: Public-Private key pair is used for secure message communication. A symmetric key is also generated to encrypt the individual keywords present in the mail. A symmetric algorithm (unlike the Public-Private key) is used here as the keywords need not be decrypted by anyone else other than the sender of the message.

Key Management: Key management is performed using a graphical user interface (GUI). The GUI enables the user to add or delete the public keys of the recipients with whom the user wants to communicate through mails.

Encryption: The functionality of the encryption module is to encrypt the messages that are sent to the server from the browser. It also extracts and encrypts the individual keywords in the message. The encryption module is triggered from the web application when the user submits a mail to send it to the web server. This module encrypts the message using the recipients's public key and the keywords with the keyword encryption key.

Decryption: When an encrypted message is sent from the server to the browser, the decryption module decrypts the messages using the private key of the user that is generated during system initialization.

C. Web Application

The webmail application provides graphical user interfaces for the users to read, send and search messages. It comprises of both server-side and client-side (browser) functionality.

When a user sends a message from the web application, the Encryption module encrypts the message and extracts and encrypts the keywords. The web application sends the encrypted message and keywords to the web server. On receiving the encrypted message and the keywords, at the server-side the application saves the encrypted message alongside the encrypted keywords in a database for future retrieval. The application then transfers the mail to the Mail Server (SMTP server) for the mail to be be delivered to recipient.

When the user wants to search for a particular keyword in their inbox, the encrypted keyword is sent to the server-side. The web application then searches for the mails corresponding to that particular encrypted word and then sends the encrypted mails back to the user.

D. Database

The mail storage and organizational functionality is already handled by the web application. One custom table, $search$ is added to the database which stores the $< message\_id, encrypted\_keyword >$ pair. This database is looked up when the user performs a keyword search.

E. Mail Server

The mail server sends and receives email communicated to it through the Internet. The mail server functionality is not modified by our system. The web application communicates with the mail server to send and receive messages.

VI. Implementation

The following software is used to implement the different components in the system:

- Browser - Google Chrome
- Browser Extension - Google Chrome using Javascript
- RSA encryption/decryption library from hanewin.net [3]
- AES encryption library [2]
- Web Application - Squirrelmail over PHP and MySQL
- Mail Server - Using the POP3 interface of the *csd.uwo.ca* mail server

The implementation details of individual modules of the system are detailed below.

A. Browser Extension

Public-Private Key Generation: The RSA algorithm [20] is used for the creation of keys. The key requires two large prime numbers as the input along with a random seed. All of these inputs are created by the extension randomly and provided as input for key generation. The keys are then stored locally along with the user name, for future retrieval in the local browser database.

AES Key Generation: The symmetric AES key algorithm is used to encrypt the individual keywords present in the mail. The AES key generation algorithm takes as input a random seed which is provided by requesting the user to move the mouse over the browser window. That generates some random co-ordinates which is then used to generate the key.

AES is a natural choice for the symmetric key algorithm as it has been analyzed extensively and used worldwide [24]. However, unlike PEKS [5], AES algorithm does not support trapdoor and hence it is susceptible to chosen plaintext attacks (The attacker has the capability to choose arbitrary plaintext and the corresponding cipher texts). Moreover the encryption of the keywords under AES negates the possibility of performing range searches (e.g., $10 < b < 20$) or similarity searches (name staring with 'ka').

Key Management: The GUI for key management is developed using the options functionality provided by the Chrome extension framework. It is used to insert the public keys of the recipients with whom the user wants to communicate. The private key of the user cannot be managed using this interface (the system automatically generates it when the user logs in for the first time). The keys are stored in the local storage database provided by HTML5.

 ISBN: 978-1-61208-153-3

Encryption: The user is provided with a HTML form from the web application which contains input fields to enter the recipient email address, subject and the contents of the mail. The form submission event ($onsubmit$ event) is associated with a custom submit event handler, which is hooked to the encryption module. The encryption module encrypts the contents of the mail using the user's public key and replaces the value in the field (contents of the mail) with the encrypted message. Along with this, the keywords in the message are extracted by the keyword extraction function and each keyword is encrypted using the AES key and stored in an object. This object is serialized in JSON (Javascript Object Notation) and sent to the server along with the encrypted message.

Decryption: When an encrypted message is sent from the server to the browser the server adds the attribute value $post-deencrypt$ to attribute $class$. The extension identifies these messages and decrypts the messages using the private key of the user. This decrypted message replaces the original encrypted message in the html page so that the user can see the message in the encrypted mail.

B. Web Application

An open source web application (Squirrelmail) is identified and it is modified for our application. Squirrelmail is responsible for storage and organization of the mails. Our custom module is developed in PHP and added to Squirrelmail to save the encrypted messages alongside the encrypted keywords and for the retrieval of the messages based on the given encrypted keyword.

VII. Experiments

The performance of algorithms used in Chaavi (Privacy Preserving Web Mail with Keyword Searches) is studied in terms of space and time consumed by the algorithm in the local client system. Even though the performance of the encryption algorithms has been studied before, we focus on the performance of our system. The results presented in this section are intended to provide some insight on the overhead provided by the algorithms in a browser based extension environment. Since encryption and decryption is performed in the client browser system, the encryption and decryption is independent of the number of users currently using the system. Hence, we focus on the performance of the encryption algorithms for a browser-based extension environment.

All the experiments are executed in a Pentium IV Core 2 Duo processor using Google Chrome 5.0.375.99 beta.

A. Time Complexity

The following algorithms are studied with respect to the execution time.

- Key Generation
- Encryption and Decryption (RSA Algorithm)
- Keyword Encryption (AES Algorithm)

1) Key Generation: Key generation is expensive since it involves finding two large random prime numbers and finding a product of the prime numbers based on the given random seed. The length of keys (as measured by bits) can be of sizes: 128, 256, 512, 1024. The higher the number of bits used, the more difficult it is to break the key (According to Schneier et al. [21], for breaking AES with key size greater than or equal to 256-bit through brute force will require fundamental breakthroughs in physics and understanding of universe). However, generating larger keys is time consuming. We present the average time taken for key generation for different bit sizes in Figure 3.

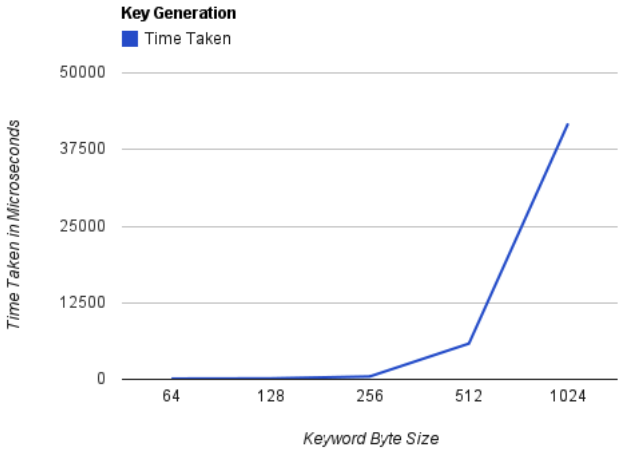


Figure 3. Key Generation

As can be seen the keyword bit size increases the creation time exponentially. The 1024 bit key generation takes around 41 seconds. However, as this is a one time activity (when the user sets up the system) the usability and inconvenience is minimal.

2) Encryption and Decryption: When the user wants to send an email the encryption module is executed each time, and the decryption module is activated when the user wants to read an email. This is a frequent activity and therefore more computation time spent on these modules will impact usability. The encryption and decryption algorithm is run over random data (which represents an email message) set using the Javascript library in Chrome browser. The performance of RSA algorithm is studied here in a browser environment. The following are the results using a 512 bit key.

It can be seen that at a relatively larger message size, around 212 KB, the time taken for encryption and decryption is less than 2 seconds. However as the message size increases in the order of megabytes, the time is around 16 seconds. A 67 MB message takes around 16 seconds to encrypt and 9 seconds to decrypt, which is still acceptable for sending such a large message. Moreover, most webmail systems have a limit of 10 MB on message sizes.

 ISBN: 978-1-61208-153-3

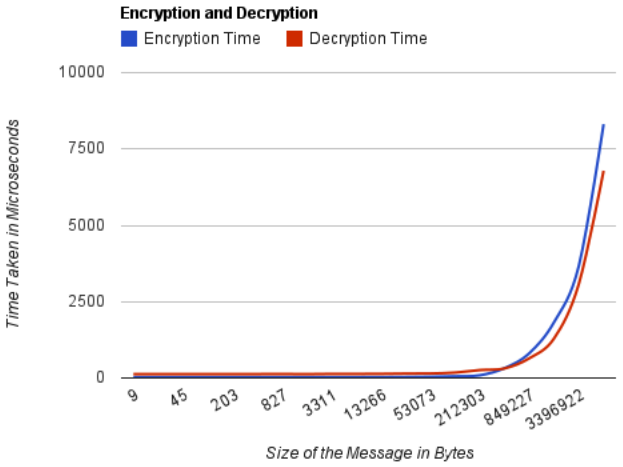


Figure 4. Encryption and Decryption

3) Keyword Encryption: In this phase the performance of AES algorithm is studied. Each word from the message is extracted and is encrypted using the AES algorithm. There is no decryption phase here, as the encrypted words are checked against each other.

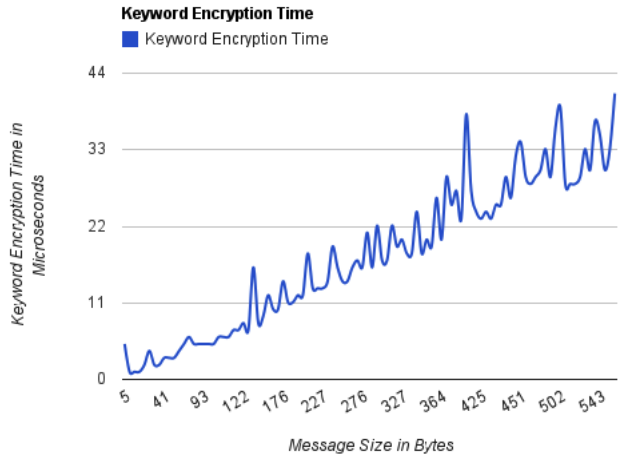


Figure 5. Keyword Encryption Time

It can been seen that there is a linear relationship between the message size and time taken for encrypting keywords. It has to be also noted that when there are duplicate words the encryption is not done twice. However, in these experiments each word was generated at random with a random size (with maximum as 25 bytes). The probability of the same word repeating is very low for this case.

B. Space Complexity

In our study of the space complexity, we were interested in the following:

1) Increase in size of the keyword index
2) Increase in the size of the final mail

1) Impact of increase in size on the keyword index: The AES algorithm is executed over the generated keywords and the impact of the size of the encrypted keywords on execution time is examined. There is close to a 10 times increase in the generated encrypted keywords compared to the keyword's actual size. This can pose a design challenge at the database level on how to store these keywords for efficient lookups at the server level.

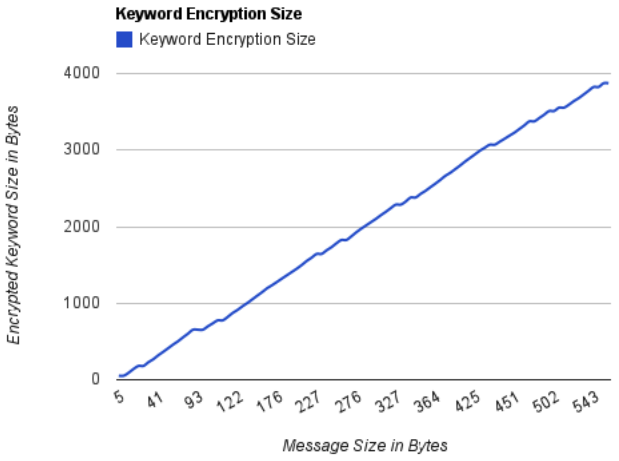


Figure 6. Keyword Encryption Size

2) Impact of increase on Final Message size: Here we study the total increase in the email size. The email that is sent to the server of the recipient will be in this format and the any increase in size, will increase the overall network traffic.

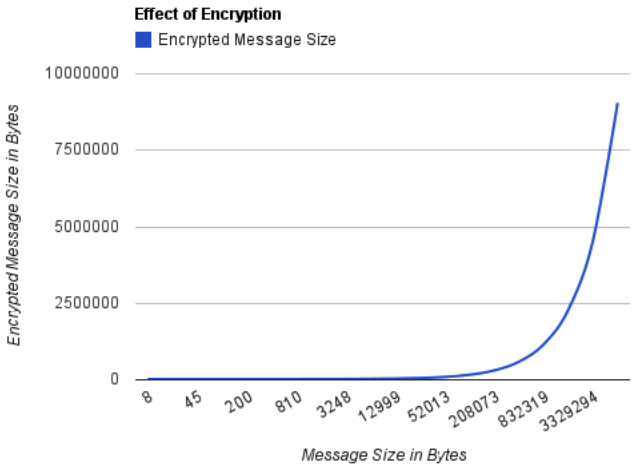


Figure 7. Message Size

It can be seen from the graph (Figure 7) that initially, when the message is transferred, there is not much of an

increase in the encrypted message size (8 bytes to 186 bytes, 18 bytes to 199 bytes, 404 bytes to 722 bytes). However as the size increases beyond 4MB there is a steep increase in the difference between the message size and encrypted message (4MB to 5MB, 8MB to 11MB, 66MB to 90MB). On average, there is a 3 times increase in size when encrypted using RSA. This is another major factor that has to be taken into consideration while using this system.

VIII. Conclusion

We proposed a privacy preserving architecture for our webmail system, that enables secure communication of messages using a public/private key model and privacy preserving keyword search functionality using AES key encryption algorithm.

Our approach requires every client to install an extension to their browser and the cloud computing provider to modify their webmail application to support encrypted keyword search. Even though technically this is a possible solution, economically a cloud provider might not prefer this approach. Most of the business models in web application are built around the contextual advertising model, where the cloud provider relies on the user's data to deliver the relevant advertisements to the user. In our case as the data is encrypted in the server, the cloud provider will not have access to the user's data. Works such as Toubiana et al. [22], try to address this problem by offloading the keyword extraction in contextual advertising to the client browser. Approaches like [22] needs to be modified for our architecture so that our system remains economically viable.

Unlike in PEKS [5], our system does not use a trapdoor function. This makes our system more susceptible to chosen plaintext attacks. If a recipient of a mail is also a potential attacker, the recipient can eavesdrop the encrypted keyword information sent from the sender to the server, and make a guess on what keyword represents the encrypted cipher by analyzing a number of mails sent to the recipient (attacker) from the same sender. However, our contribution is the proposal of the framework. The encryption algorithms used can be modified to utilize more secure alternatives in our architecture.

In our performance study, we see a considerable increase in the size of the message and the keywords after encryption. This will have a direct effect in the database storage and the keyword look up time.

We have also not implemented the functionality to add the incoming messages to the encrypted search database. Future work should address this. Future work also involves detailed study on the strength of the encryption, support to range and similarity searches, improvements to the algorithms used whilst maintaining performance.

Acknowledgements

The authors would like to thank the IBM Center of Advanced Studies and NSERC for their funding.

References

[1] http://getfiregpg.org/s/home. Online at 27th June 2011.
[2] http://www.hanewin.net/encrypt/aes/aes.htm. Online at 27th July 2011.
[3] http://www.hanewin.net/encrypt/rsa/rsa.htm. Online at 27th June 2011.
[4] Top threats to cloud computing v1.0. Cloud Security Alliance.
[5] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. *Computational Science and Its Applications–ICCSA 2008*, pages 1249–1259, 2008.
[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.
[7] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *Theory of Cryptography*, pages 325–341, 2005.
[8] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data. In *IEEE INFOCOM*, 2011.
[9] T. Dierks. The transport layer security (tls) protocol version 1.2. 2008.
[10] T. Garfinkel and M. Rosenblum. When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of the 10th conference on Hot Topics in Operating Systems-Volume 10*, page 20. USENIX Association, 2005.
[11] R. Gellman. Privacy in the clouds: Risks to privacy and confidentiality from cloud computing. In *World Privacy Forum*, pages 1–26, 2009.
[12] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.
[13] W. Itani, A. Kayssi, and A. Chehab. Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 711–716. IEEE, 2009.
[14] E. Kangas and L. President. The Case for Email Security. *Published as a Lux Scientiae Article, available at http://luxsci. com/extranet/articles/email-security. html (accessed 1 May 2007)*, 2004.
[15] L. Kaufman. Data security in the world of cloud computing. *IEEE Security and Privacy*, 7(4):61–64, 2009.
[16] D. Kenny and J. Marshall. Contextual marketing–the real business of the Internet. *Harvard Business Review*, 78(6):119, 2000.
[17] MarketsandMarkets.com. Cloud computing market - global forecast (2010 -2015).
[18] P. Mell and T. Grance. The nist definition of cloud computing. *National Institute of Standards and Technology, Information Technology Laboratory*, Version 15, 10-7-09:2, 2009.
[19] J. Postel. RFC821: Simple mail transfer protocol, 1982.
[20] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
[21] B. Schneier. Snake oil. crypto-gram newsletter (http://www.schneier.com/crypto-gram-9902.htmlsnakeoil) [online on 05th september 2011], February., 1999.
[22] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *17th Annual Network & Distributed System Security Symposium, San Diego, CA, USA*. Citeseer, 2010.
[23] S. Warren and L. Brandeis. The right to privacy. *Harvard Law Review*, pages 193–220, 1890.
[24] H. B. Westlund. Nist reports measurable success of advanced encryption standard - news briefs - national institute of standards and technology - brief article. *Journal of Research of the National Institute of Standards and Technology*, 2002.
[25] A. Yao. Protocols for secure computations. *Proceedings of the 23rd Annual IEEE Symposium on ...*, Jan 1982.
[26] P. Zimmermann. *The official PGP user's guide*. MIT Press, May 1995.

 ISBN: 978-1-61208-153-3

Distributed Storage Support in Private Clouds Based on Static Scheduling Algorithms

Dariusz Król
Academic Computer Center CYFRONET AGH
Cracow, Poland
dkrol@agh.edu.pl

Jacek Kitowski
Academic Computer Center CYFRONET AGH, and
Institute of Computer Science, AGH
Cracow, Poland
kito@agh.edu.pl

***Abstract*—This paper is focused on an extension to an open source Infrastructure as a Service Cloud called Eucalytpus for supporting distributed storage according to a defined storage strategy. As a proof of concept, three algorithms known from the scheduling theory were implemented, namely MonteCarlo, Round Robin and Weighted Queuing. To evaluate the extension, a set of tests were performed on a sample Cloud installation using a modified version of the Eucalyptus cloud. The paper ends up with a discussion on choosing the most eficient static algorithm for data storing based on the obtained results.**

Keywords - cloud computing; storage management; Eucalyptus; scheduling.

I. INTRODUCTION

Gartner has identified the Cloud computing as one of the top 10 strategic technologies in 2011 [1]. Today, most of the big Information Technology (IT) companies offer some of their products within public clouds already. These suppliers applied the Cloud paradigm to provide a wide set of applications in an easily accessible manner, starting with e-mail clients, through office suites to content resource management systems. Although, each of those applications provides different functionality, they have a few things in common, e.g., they can be accessed via a web browser, and they are provided using the pay-as-you-go manner.

Besides examples in the industry, many scientific facilities started adapting the Cloud computing. This is possible due to the existence of several open-source projects which implement the Cloud computing paradigm with open standards. While the adaption of clouds in the industry is often focused on applications, the scientific centers rather aims at providing infrastructure-level services which facilitate access to compute and storage resources.

A similar approach to resource provisioning is well known from many previous works concerning Grid environments [2]. While Clouds are business-oriented from the beginning, Grids are science oriented. From the user point of view, the main difference is the orientation on different usage modes [3]. While Clouds expose a small but well-defined interface set, Grids provides a wide-set of functions regarding similar functionality.

Existing clouds can be divided into three different groups with regard to the visibility and availability of a cloud from the users point of view. The most available are public clouds that can be used by everyone without any constraints. This category includes Amazon Elastic Compute Cloud (Amazon EC2) [4], Microsoft Azure [5], Google AppEngine [6] and many others. The opposite of public clouds are private clouds. In most cases, they are limited to the resources of a single organization and can be accessed only from within the organization's network and by an organization member. The third group concerns private clouds whose computation power and storage capacity can be extended by resources of public clouds. This group includes also hybrid clouds.

Another taxonomy of clouds concerns styles in which the customer uses Cloud. This taxonomy includes:

- Infrastructure as a Service (IaaS) Clouds which provide access to virtualized pool of resources using which customers assemble virtual machines,
- Platform as a Service (PaaS) Clouds which provide access to a well defined runtime environments and programming services which are used to develop applications without troubling with virtual machines,
- Software as a Service (SaaS) Clouds which deliver concrete applications which are deployed at the providers infrastructure.

Finally, clouds can be divided base on the type of resources which are provided. Today, this taxonomy includes two elements: compute clouds and storage clouds. The first group comprises clouds which provide access to computational power by running virtual machines or applications on a specified virtualized hardware, e.g., a virtual machine with a single, normalized, virtual CPU, 512 MB of RAM and 10 GB of hard drive capacity. On the other hand, the storage clouds enable users to store data sets in a number of ways, i.e., in files, (non-)relational databases or block devices. In theory, the storage clouds can provide an infinity storage capacity on demand.

In this paper, we focus on private, storage clouds. They can be used as an convenient way for storing users data, e.g., application results on storage resources of a single organization by organization members. It can be also used to virtualize different types of storage systems, e.g., disk arrays, local disks etc., to be visible as a single storage system from the end user point of view, thus it can increase the simplicity of sharing data between different users and applications. A storage cloud can be used to store different types of data, starting with text files and ending with binary files. As long as data can be written to a file, they can be stored in the Cloud.

To build a private, storage cloud in an effective way, a cloud implementation has to provide support for heterogeneous storage resources and different data distribution algorithms.

 ISBN: 978-1-61208-153-3

The former functionality provides a capability of connecting existing storage devices into a single system. The latter functionality is used to increase the performance of the cloud, e.g., read and write transfer rate.

In this paper, we intend to describe an extension for an existing, open-source Cloud which aims at providing a data distribution functionality based on static scheduling algorithms. Although, the developed extension is independent from a concrete data distribution algorithm, this paper focuses only on a few popular algorithms known from the queuing theory.

The rest of the paper is organized as follows. In Section 2, we describe a number of existing Cloud solutions as well as a few data management systems. Then, in Section 3, the Eucalyptus project is described in more details. In Section 4, a design of the extension of the Eucalyptus system which provides support for distributed storage is presented. Next, in Section 5 an implementation of our extension is presented. In Section 6, we come to an experimental evaluation of the presented extension. The paper is concluded in Section 7.

II. RELATED WORKS

OpenNebula [7] is an open-source toolkit for building compute-oriented private, public or hybrid clouds. The toolkit provides an abstraction layer on top of physical resources of a data center using the virtualization mechanism. It is oriented on deploying multitier services as virtual machines on distributed infrastructure. OpenNebula aims to overcome shortcomings of existing virtual infrastructure solutions, i.e., inability to scale to external clouds, a limited choice of interfaces with the existing storage and network management solutions, few preconfigured placement policies or the lack of support for scheduling, deploying and configuring groups of virtual machines. OpenNebula is fully open source and its source code can be freely checkout from a public repository. It supports different hypervisors, i.e., Xen [8], Kernel-based Virtual Machine (KVM) [9], VMware [10], for running virtual machines. In terms of storage mechanisms, it is limited to a repository of Virtual Machine (VM) images only. The repository can be shared between available nodes with the Network File System (NFS). It is also possible to take advantage of block devices, e.g., Logical Volume Manager (LVM) to create snapshots of images in order to decrease time needed to run a new instance of image. Due to this limitation, it is not a suitable tool for building storage clouds.

Another open-source solution for building different types of clouds is OpenStack. It is a joint effort of NASA and RackSpace. NASA contributed to the project by releasing its middleware, called Nebula [11], for managing virtual machines at physical infrastructure. RackSpace contributed with its storage solution known as Cloud Files [12]. OpenStack [13] is a collection of tools for managing data centers resources to build a virtual infrastructure. In terms of computations, OpenStack provides OpenStack Compute (Nova) solution which is responsible for managing instances of virtual machines. In terms of storage, OpenStack provides OpenStack Object Storage (Swift) which is an object storage solution with built-in redundancy and failover mechanisms. There is also a separate subsystem, called OpenStack Imaging Service, which can be used to lookup and retrieving virtual machine images. Since the first release of OpenStack was in October 2010, there are no articles about production deployments of the toolkit in either industry or scientific area yet. Thus, there is no information about the performance and stability of OpenStack. Also, OpenStack lacks of an interface that would be compatible with the Amazon clouds which is a *de facto* standard in the Cloud ecosystem.

Eucalyptus system [14] is an example of an open source project which became very popular outside the scientific community and is exploited by many commercial companies to create their own private clouds. It was started as a research project in the Computer Science Department at the University of California, Santa Barbara in 2007 and today is often treated as a model solution for providing infrastructure as a service. Eucalyptus aims at providing an open source counterpart of the Amazon EC2 and Simple Storage Service (Amazon S3) [15] clouds in terms of interfaces and available functionality.

There are two versions of the Eucalyptus Cloud: Community and Enterprise. The Community edition will be described in the next section in more details. The Enterprise Eucalyptus provides direct integration with Storage Area Networks (SANs) [16], e.g., Dell Equallogic or NetApp. However, to our best knowledge, this integration does not allow to combine different types of storage systems within a single Cloud installation. Also, a Cloud administrator can`t provide policy for data distribution among available storage resources.

Another commercial product is EMC2 Atmos which is a complete Cloud Storage-as-a-Service solution [17]. It provides massive scalability by allowing to manage and attach new storage resources from a single control center. Atmos delivers policy-based information management feature which allows to define bussiness level policies how the stored information should be distributed between available resources. It also reduces required effort for administration by implementing auto-configuring, auto-managing and auto-healing capabilities. Although, Atmos provides many interesting features and capabilities, it does not provide integrations with existing Clouds to our best knowledge. It is rather a separate solution oriented on the storage only which operates besides a computing Cloud.

DCache [18] is a data management system which implements all the requirements for a Storage Element in the Grid. It was developed at CERN to fulfil the requirements of the Large Hadron Collider for data storage. One of its main features is the separation of the logical namespace of its data repository from the actual physical location of the data. DCache exposes a coherent namespace built from files stored on different physical devices. Moreover, dCache autonomously distributes data among available devices according to the currently available space on devices, workload and the Least Recently Used algorithms to free space for the incoming data. Although dCache distributes data in an autonomic way, there are settings which can be configured to tune the dCache installation to specific requirements of a concrete user. This parameter set contains rules which can take as an input a directory location within the dCache file system and storage information of the connected Storage Systems as well as the IP address of the client and as an output such a rule returns a destination

 ISBN: 978-1-61208-153-3

where the data should be sent. DCache is a Grid-oriented tool by design, thus it is not compatible with existing Cloud solutions. DCache provides a programming interface similar to a filesystem interface which is at a lower level of abstraction comparing to the storage cloud interface. However, dCache could be used as a storage system which is used by a storage cloud rather than being a complete storage cloud solution.

III. EUCALYPTUS – OPEN SOURCE PRIVATE CLOUD

As describe above, Eucalyptus is one of the existing solutions for building private, public or hybrid clouds. It supports both compute and storage clouds. The most characteristic feature of Eucalyptus is the fact that it is fully compatible with the Amazon EC2 and S3 clouds at the interface layer. Therefore, it can be used interchangeably with the Amazon clouds without any modification of the users application.

Every Eucalyptus installation consists of a few loosely coupled components, each being able to run on a separate physical machine to increase scalability. The front end of such a cloud is "Cloud controller" which is an access point to the features related to virtual machines management. While "Cloud controller" is responsible for computation, the "Walrus" component is responsible for data storage. Each virtual machine runs on a physical host which is controlled by the "Node controller" element. A group of nodes can be gathered into a cluster which exposes a single access point, namely "Cluster controller" from the virtual machine management side and "Storage controller" from the virtual machine images repository side.

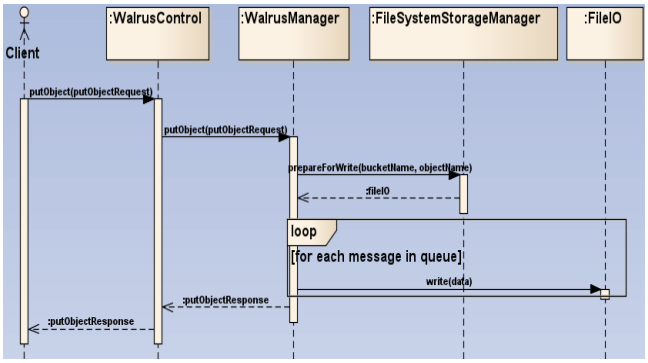


Figure 1: A sequence diagram of the "data storage" operation.

A. Data storage functionality

In terms of data storage, Eucalyptus provides two means for persisting the data generated by applications running in the Cloud: Object Storage and Elastic Block Storage (EBS).

The former one allows for storing virtual machine images along with any other files which are divided into a flat hierarchy of buckets and can be treated as the Amazon Simple Storage Service (S3) counterpart in the Eucalyptus system. Amazon S3 is a Cloud storage service which allows storing any type of data in form of files in a number of buckets (each with a unique name within a bucket) using a simple programming interface, i.e., *put*, *get*, *list* and *del*. The Eucalyptus Object Storage provides exactly the same set of functions which can be executed using a Representational State Transfer (REST) based interface. There are also several tools available which wrap the interface, e.g., a simple command line tool or programming language bindings.

The latter mechanism, i.e., Elastic Block Storage allows for providing virtual machines with block devices which are attached to virtual machines at runtime. However, unlike a virtual machine local disk, such an attached block device is not erased after the VM shutdown.

B. Data storage implementation

A part of the current implementation of storing an object within the Eucalyptus cloud is depicted in Figure 2. Due to high complexity, only one part of the "storing data" use case is presented, namely the one related to actual writing data to physical devices. The first part of the use case is related to handle HTTP requests which contain raw data that is going to be stored. Eucalyptus uses queues to handle incoming requests. Then, the *WalrusManager* object retrieves all the message objects from these queues, opens a file which is accessible with standard IO functions, and finally writes the data to the file.

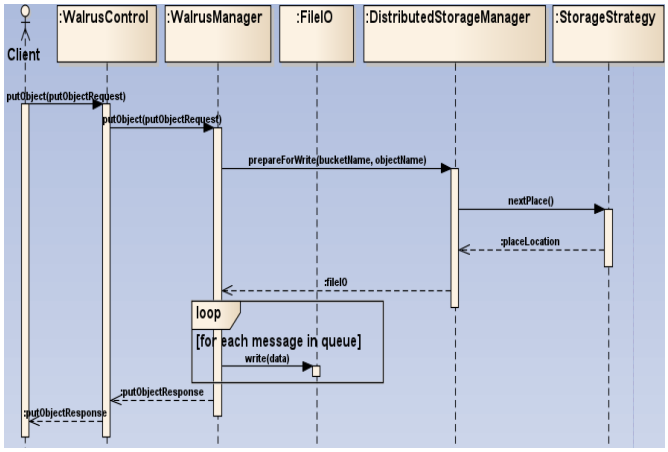


Figure 2: A sequence diagram of the modified storing data use case.

The most important part of the sequence diagram which concerns data distribution is the *preparingForWrite()* call. In the current version of Eucalyptus this method returns an object which uses the Java *FileChannel* class to write data. Moreover, a mapping between Eucalyptus objects and filesystem files implies that all the data has to be stored in a single directory. Moreover, this directory can be located only on a Walrus local disk or a volume that is attached to the Walrus machine, e.g., a disk array via Internet Small Computer System Interface (iSCSI) or a Network Attached Storage via NFS.

However, this means there is only one option to distribute the cloud data, i.e., using a distributed file system on an disk array attached to the Walrus machine which encompasses a number of storage resources. This limitation prevents from exploiting heterogeneous storage systems to build consistent storage cloud from the end user point of view. Moreover, even if heterogeneity is not an issue, distributed file systems do not provide a capability of defining storage strategies for data distribution. In most

cases, distributed file systems aim at either balancing the workload between storage devices or balancing the free space of storage devices. However, if clouds are in our scope, such a basic functionality is not sufficient.

IV. DATA DISTRIBUTION WITH STATIC SCHEDULING ALGORITHMS

In this section, we describe our solution to the problem of data distribution among several, possibly heterogeneous, storage resources. Starting with our motivation, an extension to the Eucalyptus cloud is next presented along with a sample storage strategies which are based on well-known scheduling algorithms.

A. Motivation

As described in the previous section, data distribution is poorly supported in the current version of the Eucalyptus cloud. Low-level mechanisms, i.e., distributed file systems, lack of flexibility in defining the storage strategies that exploit information about the Cloud in particular.

Just to mention a few possible applications of such strategies, let us imagine a situation where we have several disk arrays in our data center which can be used to provide storage capacity for our Cloud. However, we cannot use them all because they are shared between a number of other different projects and users thus their configuration, e.g., filesystem, cannot be modified. In such a situation, we could use only one of the available disk arrays which would probably not meet our needs because Eucalyptus does not provide means for connecting several disk arrays together into a single cloud storage.

Another possible situation is when we would like to separate users' data, based on groups a particular user belongs to. Such a users' group can be bound to a Service Level Agreement between the user and the Cloud provider. From the Cloud provider point of view, each users' group could be handled by a different physical device, i.e., the users who pay more are treated with more reliable and efficient resources.

Also many other situations can be described where support for distributed storage is crucial to succeed but the importance of this functionality should be clearly visible in advance.

B. Design and implementation

When designing an extension to Eucalyptus that provides support for distributed storage, we focused on making it as non-intrusive as possible. Thus, we decided to replace an existing implementation of the *StorageManager* Java interface, namely an instance of the *FileSystemStorageManager* class (depicted in Figure 1) with its another implementation which is aware of the distributed storage. By doing so, we can activate this functionality with only two modifications to the Eucalyptus source code, i.e., in the places where the *StorageManager* variables are instantiated. Even these modifications can be eliminated by using the Dependency Injection pattern [19] and one of its Java implementation, e.g., the Spring framework [20].

A modified version of the "data storage" use case is shown in Figure 2. Due to being part of the Eucalyptus cloud, this extension has access to the whole information about cloud users, user data, etc. Therefore, it can implement a storage strategy on a higher level of abstraction than a distributed file system.

The implemented prototype of this extension enables a Cloud administrator to decide which storage strategy should be used by only modifying one configuration file that besides information about the storage strategy, contains information about available storage resources.

C. Implemented data distribution strategies

Although, the described extension is versatile, i.e., various storage strategies can be implemented and used at runtime, we implemented three strategies as a proof of concept. We exploited algorithms known from the scheduling theory:

- *MonteCarlo* strategy which randomly (with a uniform distribution) chooses a place to store the given data.
- *RoundRobin* strategy which stores the given data alternately on each of the available resources.
- *WeightedQueue* strategy which divides the available bandwidth to a number of channels whose "width" is proportional to weights assigned to storage resources. In the basic version of this algorithm, the weights are assigned to each device arbitrarily by the administrator.

The proposed strategies represent a group of so called *static* scheduling algorithms. As opposed to *dynamic* scheduling algorithms, they do not change the scheduling scheme, i.e., the order of storage resources, as a response to changes in the environment, e.g., infrastructure workload.

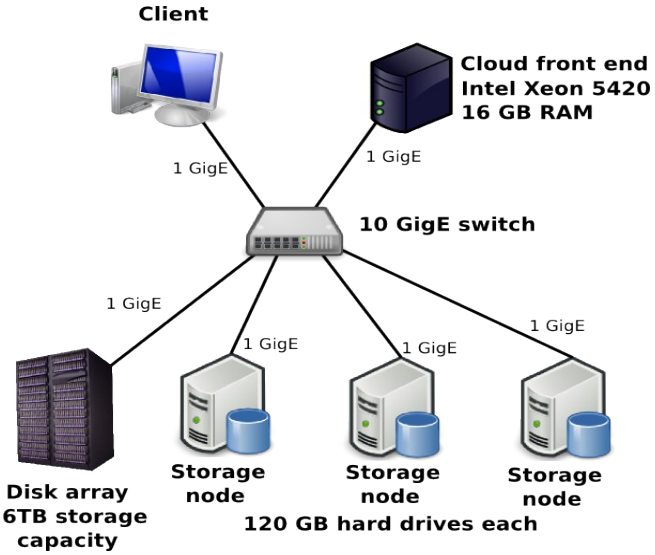


Figure 3: A map of a testing environment.

Although, the static scheduling algorithms can be less efficient than the dynamic ones, they are more predictable and straightforward. Thus, they are more suitable for testing the described functionality comparing the currently available Eucalyptus version. Also, they are more suitable than business-level algorithms because they allow to focus to performance analysis rather than on functional requirements,

e.g., distributing data of different users groups to different storage resources.

V. EXPERIMENTAL EVALUATION

In order to evaluate the implemented extension, a proper testing infrastructure has been composed and a number of tests were performed. The evaluation aimed at finding which storage strategy provides the highest throughput of the Cloud infrastructure. In addition, we would like to find out whether a cloud storage can be built based on commodity hardware, e.g., standard hard drives connected with a commodity ethernet network, instead of expensive disk arrays connected with a special network such as Storage Area Network (SAN) based on FibreChannel, with maintaining the Cloud performance at the same level.

A. Testing environment

Testing environment is a very important aspect of the experimental evaluation. Thus, we prepared a sample configuration for building a small Cloud installation based on a blade-class cluster nodes and a disk array. As a base server for an extended version of the Eucalyptus cloud we use a worker node with the following parameters:

- 2x Intel Xeon CPU L5420 @ 2.50GHz (4 cores each)
- 16 GB RAM
- 120 GB hard drive (5400 RPM)
- Ubuntu Linux 10.04.1 LTS.

Apart from the Cloud front end where the Cloud controller and Walrus components were installed, we also have three similar nodes for running virtual machines connected with the front end by Gigabit Ethernet.

However, a more interesting part of the environment concerns the storage. As a main storage for our cloud installation we used a part of a disk array accessible via iSCSI protocol, with 6 TB of storage capacity. Such a disk array, however, with a greater storage capacity available, could be used in a production cloud. As an additional storage, we decided to use hard drives from the additional worker nodes which are exposed via the NFS protocol.

To summarize, we depicted a map of the testing environment in Figure 4. In our opinion, the presented environment can be effectively used to evaluate different storage strategies because it contains heterogeneous storage resources such as hard drives and disk array distributed among a few machines all connected with open protocols and commodity network fabric.

B. Testing scenario

In the presented case, we proposed a scenario in which a number of users stores files in the Cloud simultaneously. Such a scenario is parametrized with the following elements:

- number of users running in parallel – 10
- file size – 128, 256, 512, 1024, 2048 MB
- storage strategy – MonteCarlo, RoundRobin, WeightedQueue (with a number of different weight vectors)

Each test scenario was perfomed 5 times and the mean value was computed. The performance evaluation metric used in the presented tests is the Cloud write throughput. The metric represents the total rate of writing data by the Cloud to its storage resources. This metric allows to compute the overhead generated by Eucalyptus to the storing data operation. Moreover, we can analyze the utilization rate of the storage resources with respect to different storage strategies.

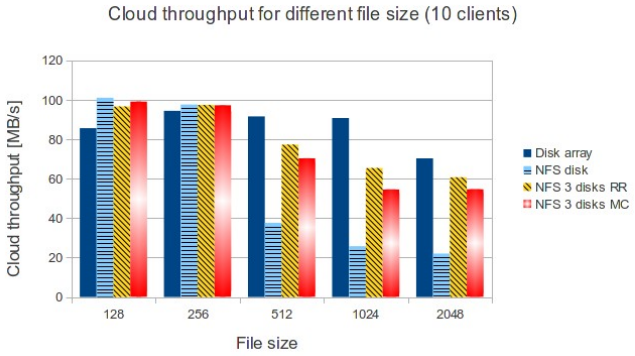


Figure 4: The Cloud throughput depending on a file size with 10 clients run in parallel.

C. Results and discussion

Firstly, the results coming from the tests performed with a single storage resource and with storage resources accessible via NFS are depicted in Figure 4. The results show a huge difference between the performance of the Cloud which uses a disk array and the Cloud which uses a common hard drive connected via NFS. The difference increases with the file size. This is expected due to the cache mechanism. When the file size is greater then the system cache then the performance of the Cloud gets stable. A second thing to notice is the performance of the Cloud which uses three connected hard drives via NFS. The mean performance of this configuration is smaller than in the configuration with a disk array but they are comparable. Also we can notice a slight performance gain when the *RoundRobin* strategy has been used. Also, we should notice a large diversity of measurement values in the storage configuration with a single NFS disk. The smallest diveristy of measurement values was obtained with a configuration of the disk array.

The second part of the results which contains the measured throughput with regard to the selected storage strategy is depicted in Figure 5. This test was performed with a Cloud installation which includes a disk array and three hard drives, exposed via NFS.

The results show that the *MonteCarlo* strategy is the worst one. For 1024 MB files the Cloud throughput for the *MonteCarlo* is less by 1/5 than the Cloud throughput for the *RoundRobin* strategy. The performance achieved in other strategies are similar and are close to 95 MB/s. Since the theoretical network performance is about 125 MB/s the achieved throughput is about 76% of the theoretical value and slightly more then 80 % in the best case.

Comparing the distributed storage to non-distributed storage, the results show about 10% of performance gain. Such a small gain is probably due to the limited network bandwidth rather than storage resources throughput

 ISBN: 978-1-61208-153-3

limitations. Thus, it is highly probable that, if there would be more than one physical network interface (as in our testing environment) coming from the Cloud front end, the Cloud throughput would scale better with the additional resources.

Table 1: Statistical parameters (in MB/s) for the throughput measurement for different storage strategies.

Storage Strategy	Mean	Variance	Confidence interval (α=0.05)
Round Robin	96.98	10.07	[93.96; 100.01]
Monte Carlo	90.50	110.43	[80.48; 100.52]
WQ-32111	96.71	6.91	[94.21; 99.22]
WQ-21111	95.33	0.21	[94.88; 95.77]
WQ-31111	96.37	1.26	[95.29; 97.44]

In Table 1, we gathered important statistical parameters which describe data from the second test case. Although the RR storage strategy leads to the largest mean throughput, the narrowest confidential interval can be obtained with the weighted queue strategy. The MC strategy is the most unpredictable.

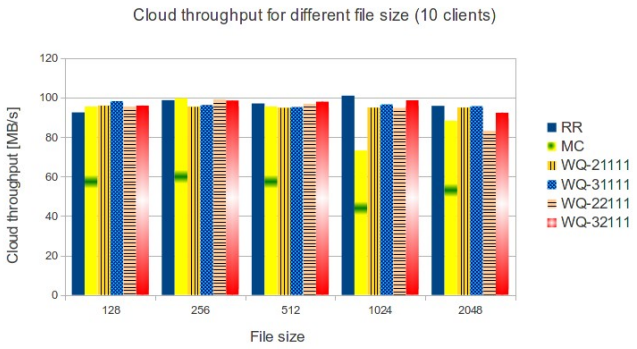


Figure 5: The Cloud throughput depending on a storage strategy.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we aimed at emphasizing the necessity of supporting distributed storage in building storage clouds. Upon having compared several open-source toolkits for building private clouds we decided to use Eucalyptus due to its compatibility at the interface level with the *de facto* standard in Cloud ecosystem, i.e., the Amazon clouds. As described in Section III, the current version of Eucalyptus does not provide sufficient functionality regarding data management. A non-intrusive extension to Eucalyptus has been proposed and implemented. The results from a number of performed tests show that a distributed storage can improve the Cloud throughput comparing the original implementation even if commodity hardware is used. Moreover, when using a distributed storage, the Cloud performance gets stable near the theoretical value of the network bandwidth.

The future work concerns improving the stability of the proposed extension. Also some new storage strategies, similar to those described in Section 4, are going to be provided.

ACKNOWLEDGMENT

This research is supported partly by the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project . The authors are grateful to Dr. Dr. Łukasz Dutka, Renata Słota and Włodzimierz Funika for valuable discussions.

REFERENCES

[1] Gartners report about the Top 10 Strategic Technologies for 2011, [online: http://www.gartner.com/it/page.jsp?id=1454221, as of April 16, 2011].

[2] The iRODS project website: [on-line: https://www.irods.org, as of April 16, 2011].

[3] S. Jha, A. Merzky, and G. Fox, "Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes", Journal Concurrency and Computation: Practice & Experience, vol. 21 (8), pp. 1087-1108, June 2009.

[4] Amazon Elastic Compute Cloud website [on-line: http://aws.amazon.com/ec2, as of April 16, 2011].

[5] Microsoft Windows Azure Platform website [on-line: http://www.microsoft.com/windowsazure/, as of April 16, 2011].

[6] Google AppEngine website [on-line: http://code.google.com/appengine/, as of April 16, 2011]

[7] D. Milojičić, I. Llorente, and R. Montero, "OpenNebula: A Cloud Management Tool," IEEE Internet Computing, vol. 15(2), pp. 11-14, Mar./Apr. 2011, doi:10.1109/MIC.2011.44.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, and A. Warfield, "Xen and the art of virtualization," in SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles. NY, USA: ACM, 2003, pp. 164-177.

[9] Kernel-based Virtual Machine project wiki. [on-line: http://www.linux-kvm.org, as of April 16, 2011].

[10] VMware website. [on-line: http://www.vmware.com, as of April 16, 2011].

[11] NASA Nebula website. [on-line: http://nebula.nasa.gov/, as of April 16, 2011].

[12] RackSpace CloudFiles solution website. [on-line: http://www.rackspace.com/cloud/cloud_hosting_products/files/, as of April 16, 2011].

[13] OpenStack project website. [on-line: http://www.openstack.org, as of April 16, 2011].

[14] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System", CCGRID '09 Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society Washington, DC, USA 2009.

[15] Amazon Simple Storage Service project website, [on-line: http://aws.amazon.com/s3/, as of April 16, 2011].

[16] Introduction to Storage Area Networks, IBM redbook, [on-line: http://www.redbooks.ibm.com/abstracts/sg245470.html?Open, as of April 16, 2011].

[17] EMC2 Atmos product web site, [on-line: http://www.emc.com/storage/atmos/atmos.htm, as of April 16, 2011].

[18] G. Behrmann, P. Fuhrmann, M. Gronager, and J. Kleist, "A distributed storage system with dCache", in G .Behrmann et al Journal of Physics: Conference Series, 2008.

[19] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", 1995, ISBN: 0-201-63361-2.

[20] Spring Framework website. [on-line: http://www.springsource.org/, as of April 16, 2011].

 ISBN: 978-1-61208-153-3

Open Environment for Collaborative Cloud Ecosystems

Oleksiy Khriyenko
Industrial Ontologies Group, MIT Department
University of Jyväskylä, P.O. Box 35(Agora)
Jyväskylä, Finland
oleksiy.khriyenko@jyu.fi

Michael Cochez
Industrial Ontologies Group, MIT Department
University of Jyväskylä, P.O. Box 35(Agora)
Jyväskylä, Finland
michael.s.l.cochez@jyu.fi

Abstract — **Cloud computing can be defined as accessing and utilizing third party software, services and resources and paying as per usage. It facilitates scalability and virtualized resources over the Internet as a service; providing cost effective and scalable solution to customers. There are two emerging methodologies for constructing infrastructure: "Cloudcenters" and "Infrastructure Web Services". Cloudcenters can be regarded as a virtualized data center. Infrastructure Web Services are more analogous to Service-Oriented-Architectures (SOA), require significant programming skills and are much more comfortable for software developers. It is a robust ecosystem of services which you can use in order to build your application, getting the traditional benefits of Cloud Computing such as self-service, pay-as-you-go, and massive scalability. Unfortunately, talking about openness and interoperability in cloud computing, cloud providers still operate very much in their own silos and private-cloud APIs drift further and further apart. Most data center vendors do not offer users complete vertically integrated cloud stacks. However, they are often providing solutions which imply a strong vendor lock-in. A lot of activities are currently aimed at the development of various Cloud computing environments and software engineering practices for the management of distributed applications, services and other resources. We are thinking about a future vision of a network of clouds. It should be an open market for components (applications, services, data sources, etc.) and composed ecosystem infrastructure services that facilitate appropriate collaboration for personalized needs. In this paper we would like to slightly modify the original cloud stack towards the development of an open environment for task-oriented personalized cloud ecosystems and apply a resource integration platform for this ecosystem elaboration.**

Keywords-collaborative clouds; cloud interoperability; component-based ecosystem infrastructure; semantic integration

I. INTRODUCTION

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and system software in the data centers which provide these services. 'Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry by making software as a service even more attractive and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it.' [1].

Clouds have emerged as a computing infrastructure that enables rapid delivery of computing resources as a utility in a dynamically scalable and virtualized manner. The advantages of cloud computing over traditional computing include: agility, lower entry cost, device independence, location independence, and scalability. There are two emerging methodologies for constructing infrastructure: "Cloudcenters" and "Infrastructure Web Services". Cloudcenters provide the same kinds of tools that data center and server operators are already accustomed to, but with all the advantages of cloud (i.e., self-service, pay-as-you-go and scalability). Instead of creating completely new paradigms, cloudcenters are a methodology by which you, the customer, can have a virtual data center hosted in the "sky". It allows the use of the same tools, paradigms and standards that are deployed in an industry standard data center today. Cloudcenters provide a direct equivalent to traditional data centers including all of the regular components you expect such as hardware firewalls, hardware load balancers, network storage, virtualized servers, dedicated networks, and the option for physical servers for workloads that should not be virtualized. Thus they are usually more desirable for IT staff, systems operators, and other data center specialists. Infrastructure Web Services on the other hand are more analogous to Service Oriented Architecture (SOA), require significant programming skills, and are much more comfortable for software developers. In this case, the infrastructure provides a number of different services (Object-based file storage, Servers on demand, Distributed database functionality, Content distribution, Messaging & queuing, Payment processing, etc.) that can be consumed individually or together to facilitate different kinds of applications. This is a robust ecosystem of services which you can use in order to build your application.

For all the talk about openness and interoperability in cloud computing, both public-cloud and private-cloud providers still operate very much in their own silos [2]. The cloud ecosystem is challenged by the fact that cloud service providers provide their own ways on how users or cloud applications interact with their cloud, resulting in vendor lock-in, non-portability and inability to use the cloud services provided by multiple vendors. This often includes the inability to use an organization's own existing data center resources seamlessly with the offered infrastructure. Cloud

 ISBN: 978-1-61208-153-3

computing is gaining popularity and IT giants such as Google, Amazon, Microsoft and IBM have started their cloud computing infrastructure. All of them are doing wonderful things — but they are doing so largely within their own environments. 'And while (most) data center vendors don't offer users complete vertically integrated cloud stacks, they are more than happy to lock users into their product lines as much as possible and form strong partnerships in areas they don't play.'[2]. Golden [3] states that current cloud implementations do not allow enterprise applications to be migrated conveniently; imply legal, regulatory, and business risks; are difficult to maintain ; lack service level agreements and do often not give a cost advantage.

Nowadays, activities are mainly aimed at the development of various Cloud computing environments and software engineering practices for management of distributed applications, services and other resources. However, development is still focused on enterprise level clouds, which may result in the creation of architectures with the drawback of heterogeneity, non-interoperability of components, and inability of the systems to be reconfigurable on demand. Effort is already done in order to make providers' offers interchangeable. One such example is the Open Virtualization Format (OVF). 'The OVF specification is a hypervisor-neutral, efficient, extensible, and open specification for the packaging and distribution of virtual appliances composed of one or more VMs. It aims to facilitate the automated, secure management not only of virtual machines but the appliance as a functional unit.' [4]. The same source states however that 'For the OVF format to succeed it must be developed and endorsed by ISVs, virtual appliance vendors, operating system vendors, as well as virtual platform vendors, and must be developed within a standards-based framework.' This requirement might show to be to strong in reality.

We think that it is time to start thinking about a future vision of a network of clouds. It should be an open market for components (applications, services, data sources, etc.) and composed ecosystem infrastructure services that facilitate appropriate collaboration for personalized needs. Such ecosystem-based environment allows the collection and management of applications and the composition of mash-ups based on them. The applications used to compose mash-ups can be found from the users own private pool of components and services or from the open marketplace provided by different cloud service providers. Furthermore, the ecosystem-based environment allows enterprises and individuals to choose what kind of ecosystem infrastructure services to utilize for the service collaboration and personalized user experience. Such architecture allows us to create personalized abstract clouds. An abstract cloud is a description of infrastructure, platforms and software which does not have to mention all concrete components. These concrete components can later on be selected, even on the fly, by the user of the abstract cloud. Abstract clouds can be made available through the open marketplace to be used as application oriented infrastructure or as a sub-cloud for own personalized infrastructure cloud composition. In this paper we would like to slightly modify the original cloud stack towards the development of an open environment for task-oriented personalized cloud ecosystems. We will apply a resource integration platform for this ecosystem elaboration.

II. SMART RESOURCE INTEROPERABILITY

A. Technologies Towards Intelligent Interoperability

With the presence of numerous vendors, the need for interoperability between clouds emerges. The goal is to make complex and developed business applications in the cloud interoperable. To achieve the vision of ubiquitous knowledge, the next generation of integration systems might need different technologies as the ones currently used. Technologies such as Semantic Web [5][6], Web Services [7][8], Agent Technologies [9], and Mobility[10]. Semantic technologies are viewed today as a key technology to resolve the problems of interoperability and integration within the heterogeneous world of ubiquitously interconnected objects and systems. Still, aspects of proactivity of these resources are quite in demand nowadays and should be considered more comprehensively.

In recent years, the complexity of computing environments has grown beyond the limits of human system administrators' management capabilities. With the advent of service-oriented computing (SOC), computing environments have become open and distributed, and components are no longer under a single organization's control. Moreover, the typical enterprise computing environment is a heterogeneous, irregular, multivendor pastiche which is difficult to configure, maintain, and trouble-shoot. Autonomic computing systems are expected to free system administrators to focus on higher-level goals [11]. Self-configuration (systems configuring themselves automatically when computing resources are added or removed), self-healing (discovering when, where and why systems are ailing and performing the appropriate self-repair and fault-correction operations), self-optimization (monitoring and controlling resources to ensure optimal functioning with respect to defined requirements, as well as optimizing performance and efficiency by reconfiguring themselves) can be performed by autonomic computing systems without human intervention. Autonomic computing systems can perform these functions at both the infrastructure and application levels. As such, autonomic computing systems strongly resemble multi-agent systems (MAS). MAS, in turn, interact with services, as designed and developed within SOC. When it comes to developing complex, distributed software based systems; the agent based approach is advocated in Jennings [12]. The vision of autonomic computing emphasizes that the run-time self-manageability of a complex system requires its components to be, to a certain degree autonomous themselves. From the implementation point of view, agents are the next step in the evolution of software engineering approaches and programming languages, a step following the trend towards increasing degrees of localization and encapsulation in the basic building blocks of programming models [13].

 ISBN: 978-1-61208-153-3

Developing and maintaining large-scale, distributed applications is a complex task. Middleware has traditionally been used to simplify application development by hiding low-level details and by offering generic services that can be reused and configured by application developers. However, middleware technology has not kept up with the growing demands that emerge in the digital society: the scale of distributed applications is rapidly increasing, the range of users that compose and configure applications has expanded significantly, and the increased scope of distributed applications has also resulted in more advanced application composition scenarios.

B. UBIWARE Platform: Integration Infrastructure for Heterogeneous Distributed Components

As a basis for our research towards open environment for personalized task/domain oriented cloud ecosystems we use the UBIWARE platform [14]. This platform follows the GUN vision described in [15]. The UBIWARE platform is a development framework for creating multi-agent systems. It is built on top of the Java Agent Development Framework JADE [16], which is a Java implementation of IEEE FIPA specifications. The name of the platform comes from the name of the research project, in which it was developed. The UBIWARE project introduced a new paradigm in software engineering and elaborated an approach towards creation of semantically enhanced agent-based integration middleware that makes heterogeneous resources proactive, goal-driven and able to interoperate with each other in collaborative environment [17]. In this project, a multi-agent system was seen, first of all, as a middleware providing interoperability of heterogeneous resources and making them proactive and in a way smart.

The core of the platform gives every resource a possibility to be smart by connecting a software agent to it. This agent enables the component to proactively sense, monitor and control its own state and communicate with other components which are also represented by agents in the system. Furthermore, the component can compose and utilize internal and external experiences and functionality for self-diagnostics and self-maintenance. UBIWARE enables the resources to automatically discover each other and to configure a system with complex functionality based on the atomic functionalities of the resources. It ensures a predictable and systematic operation of the components and the system as a whole by enforcing that the smart resources act as prescribed by their organizational roles and by maintaining the "global" ontological understanding among the resources [18]. The main goal of the platform is to provide interoperability between heterogeneous resources (applications and systems in our case) through semantic adaptation and the proactive agent assigned to each of the resources. All communication, resource discovery and use of resources (e.g., application and systems) are performed trough its corresponding agent. The platform has inter-platform communication mechanisms and allows integration, orchestration and choreography of resources registered and located on different platforms. UBIWARE is not an application like an operating system, word processing software or Internet browser. It is a set of tools that helps people develop software. With respect to cloud-based integration environment interoperability, we consider the UBIWARE platform as a tool that enables automatic discovery, orchestration, choreography, invocation and execution of different Business Intelligence services.

III. OPEN ENVIRONMENT FOR COLLABORATIVE CLOUD ECOSYSTEM

A. Cloud Stack for Collaborative Cloud Ecosystem

Most of the current cloud implementations are built on top of data centers. According to NIST [19] cloud computing incorporates Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), and provide these services as utilities. Data centers are a foundation of cloud computing which provides the hardware clouds run on. IaaS is built on top of the data centers and virtualizes the computing power, storage and network connectivity of the data centers, and offers them as provisioned services to consumers. In other words, consumers have the possibility to configure a virtual computer, where he can select a configuration of CPU, memory and storage that is suitable for the intended application. The whole cloud infrastructure (i.e., servers, routers, hardware based load-balancing, firewalls, storage, and other network equipment) are provided by the IaaS provider. The customer buys these resources as a service as needed. Examples of this layer include the Amazon EC2 service [20] and Microsoft's Windows Azure platform [21]. PaaS provides a development platform with a set of services to assist application design, development, testing, deployment and monitoring, hosted on the cloud. It is sometimes referred to as cloudware. Google App Engine, Microsoft Azure, Amazon Map Reduce/Simple Storage Service, etc are among examples of services residing in this layer. In SaaS, Software is presented to the end users as services on demand, usually in a web browser. It saves the users from the troubles of the software deployment and maintenance. The software is often shared by multiple tenants, automatically updated from the cloud, and no additional license needs to be purchased. Because of its service characteristics, SaaS can often be integrated easily with other mashup applications. One example of SaaS is Google Maps and its mashups across the Internet. However, the separation in IaaS, Paas and Saas is mainly a service model. Components and features of one layer can in practice also be implemented on another layer and the upper layer does not have to be built on top of its immediate lower layer.

In the cloud computing environment, everything can be implemented and treated as a service. Software development "in the cloud" has been one of the really interesting developments to come out of the cloud computing market so far. Regarding PaaS, more and more cloud providers enhance their platforms with specific services that simplify application development for their customers and, in such a way, bind the customers to their platforms. There are services like payment systems, information search systems, GEO-systems, specific data bases, etc. One example of this

 ISBN: 978-1-61208-153-3

kind of services is the datasets provided by Amazon [22]. Together with private specific services from cloud providers there are quite many freely open sources and commercial services provided by third parties. To generalize the concept of this kind of services, and take into account that users of such services are not human, but other applications and services, we name them as a SaaS for Software (SaaS4S) or SaaS for SaaS (SaaS4SaaS).

In the proposed solution, a service (be it infrastructure, platform or software) should be registered (connected through adapter) to the integration environment UBIWARE and be semantically annotated according to common ontology in order to enable the environment to discover and orchestrate them. Any service can have access restrictions. This gives an opportunity to use own capabilities together with, or instead of, others when security and privacy of processed data is crucial. Using the UBIWARE platform as a tool for application and service integration, we may create an open environment for the components across different clouds.

Figure 1 shows the proposed extended cloud stack which allows us to organize collaboration between services and applications located in different clouds. As in the original cloud stack, there are Application Development Tools that users of the PaaS layer use to develop and run their applications.

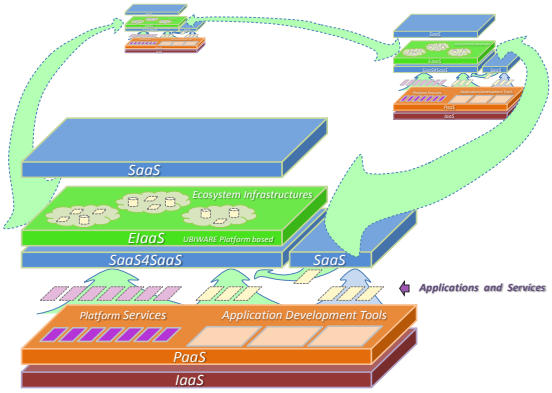


Figure 1. Cloud Stack for Collaborative Cloud Ecosystem.

There are services and functionalities that many of cloud providers supply with their platforms to facilitate users application development. Using the UBIWARE platform as one of the applications run on top of PaaS layer we may:

- transform applications that are presented for humans on the SaaS layer to services available for other software (SaaS4SaaS);
- support the users of the PaaS layer to develop and register applications directly for SaaS4SaaS layer;
- make specific platform services available for use on the SaaS4SaaS layer.

The UBIWARE platform allows semantic adaptation of different data sources and makes them accessible as services for other services and applications that operate through the platform. With correspondent tools (provided by the UBIWARE platform) users may create and define task and domain specific Personalized Ecosystem Infrastructures (PEIs) as compositions of services (addressed by ecosystem Infrastructure Modules) and data sources. They can then use them as services on demand – Ecosystem Infrastructure as a Service (EIaaS). Thus, on the EIaaS layer, the UBIWARE platform provides a possibility to create new services on top of cross-cloud semantic orchestration and choreography of distributed components.

B. *Personalized Context-Aware and Self-Configurable Cloud Ecosystem*

A component-based approach for the Cloud Ecosystem development provides us a flexible way to elaborate an ecosystem through the composition of different (heterogeneous) modules on the level of Ecosystem Infrastructure and on the level of Application composition (Figure. 2). We utilize the same approach of component-based system development on both levels and provide an interoperability of heterogeneous components (modules) developed by various providers in an open collaborative environment. As a foundation for a collaborative ecosystem environment, we consider a network of platforms that provide cross-platform communication, interoperability of heterogeneous components and a toolbox for their composition. The UBIWARE platform is developed as a smart semantic middleware for ubiquitous computing and is based on integration of several technologies: semantic web, distributed artificial intelligence, agent technologies, ubiquitous computing, SOA, Web X.0 and related concepts. We regard this platform as our basis and intend to extend its functionality towards the needs of this Cloud Ecosystem elaboration.

The Core Ecosystem Engine is an engine which provides a mechanism for a component-based Ecosystem Infrastructure composition. On the Ecosystem Infrastructure development level we have a pool of components – Ecosystem Infrastructure Modules (EIMs). These EIMs can be used for Personalized Ecosystem Infrastructure (PEI) creation where only relevant EIMs are composed. With respect to openness of our collaborative environment, such PEI can itself be published to the public zone and be used as sub-PEI in other personalized ecosystem infrastructures.

 ISBN: 978-1-61208-153-3

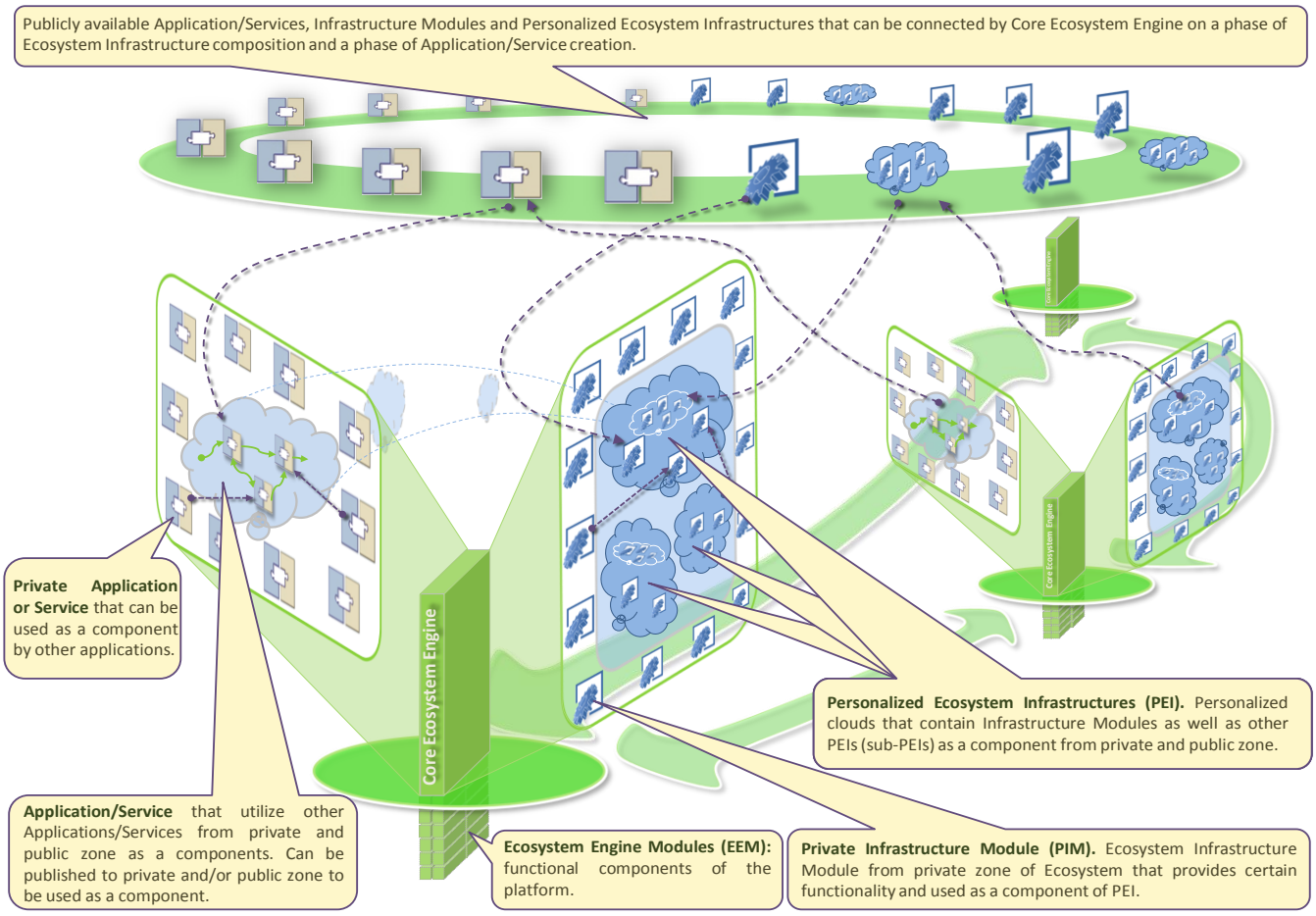


Figure 2. Open environment for collaborative component-based Ecosystems.

In the same way, the Core Ecosystem Engine of the platform provides a mechanism for component-based service composition on top of the selected Personalized Ecosystem Infrastructure. The user may have a private zone containing the set of available applications and services on the platform. At the same time, the platform allows the connection of publicly available services, published to the public zone of the environment. Semantic policy-based control of the platform brings security aspects to the system. This policy based control allows users to distinguish between public and private components and guarantees protection of sensitive data.

In order to make a valuable step towards intelligent services, we should not to limit ourselves to the creation of specific services. We have to think about more flexible solutions that allow us to create new services through orchestration of reusable collaborative intelligence and about a supportive integration environment that gives us the possibility to create new context-aware services through the integration of various data sources and intelligent capabilities with a flexible semantic process. To increase the flexibility and reliability of the applications and the services created on top of Ecosystem Infrastructures, we consider a semantic definition of Abstract Infrastructure. According to the semantic web vision, not only programs and data are distinguishable, but also components of more complex systems are considered as separate modules. These components may be replaced by components which are semantically similar and more suitable in the current context.

Applying a semantic web approach for the Ecosystem Infrastructure creation, the user may define a so called Abstract cloud, which will be on-the-fly transformed into concrete appropriate Infrastructure based on the available components from different Clouds depending on the correspondent context. Providing interoperability of heterogeneous components, Ecosystems should be flexible and at certain level intelligent. Utilizing the semantic web approach, the UBIWARE platform makes the Ecosystem proactive and able to configure itself on-the-fly depending on context and user needs. The platform provides a possibility for the user to define a process with preferences and constrains and executes it as an on-the-fly orchestration of available capabilities and available data, based on their semantic descriptions, through semantic matching and discovery mechanisms (Figure 3).

Although we have a complex network of heterogeneous services, applications and data sources distributed among different clouds, users of EIaaS layer see the Ecosystem as one common entity accessible through the common UBIWARE interface. Figure 4 shows us the general structure

 ISBN: 978-1-61208-153-3

of the Ecosystem. The UBIWARE platform provides corresponding tools for the Ecosystem Users (the providers and users of personalized Ecosystem Infrastructures) and transparency for collaboration of distributed components. Through interoperability between other UBIWARE platforms this automatically organizes an open market place of publicly available ecosystem infrastructures and services keeping the possibility of private zones for sensitive information.

C. *Case Scenario: Creation of Personalized Ecosystem Infrastructure and Launching Self-configurable Application as a Set of Composed Services*

This is a joint use case with two players that shows nested utilization of the ecosystem platform on two layers. It can also be regarded as two separate scenarios.

Player 1 is an ecosystem infrastructure provider who has a set of own components - infrastructure modules. The player would like to create a Personalized Ecosystem Infrastructure (PEI) as a set of (a) own modules, (b) some publicly available modules shared in the open marketplace of the components and (c) specific services provided by his cloud provider. To achieve that goal, Player 1 has to:

- register his own components locally to the platform's private zone (through the registration tool of the platform) and connect them via semantic adapters;
- find other necessary components from the open shared space provided by third parties;
- provide a semantic description of the abstract components that will be on-the-fly transformed (discovered and invoked) to appropriate ones for the current context;
- create appropriate adapters to make cloud-specific services available through them;
- Provide a semantic annotation of the created Ecosystem Infrastructure.

Thus, using features of the platform such as: adapter-based connection of components (data sources and services), component discovery (based on their semantic specification), browsing of available resources based on their semantic description and the tool for semantic annotation of resources, Player 1 may create a PEI, annotate it and publish it to the marketplace.

Player 2 is a user of the PEI (provided by Player 1 or any other EI provider) and a service provider at the same time. The player would like to find an appropriate ecosystem infrastructure to create and launch his own application on top of it, as a publicly available service. An application is meant to be a dynamic self- configurable composition of several services. The way the application should work is context dependant. Among the relevant context variables are service availability, reliability, cost, and user and service location.

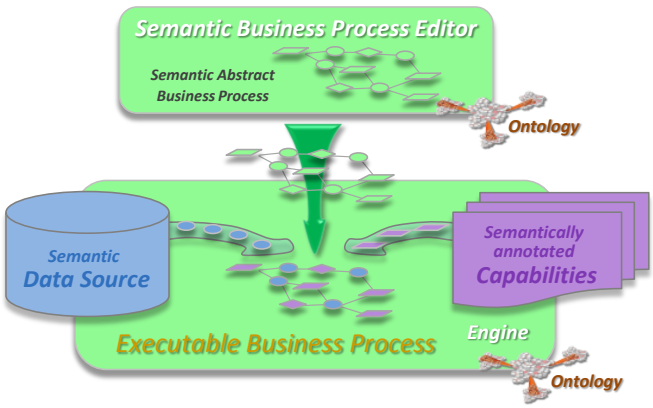


Figure 3. Semantic Abstract Business Process of UBIWARE.

Player 2 enters the platform and selects or finds (via the corresponding tools of the platform) an appropriate Ecosystem Infrastructure which fits the requirements of the player depending on task and domain specifics of the planned service. Utilizing the infrastructure components of the corresponding ecosystem and the abstract process definition tool of the platform, Player 2 defines a partially abstract process. Components are described through their semantic annotations and will at run-time be selected among appropriate available components, depending on the context. Thus, concrete instances of the composed service will be built on-the-fly and executed by the platform engine. After Player 2 has published his/her application, it can be used by service users on the web.

IV. CONCLUSIONS

Within this paper we aimed at showing possible steps on how openness and interoperability of cloud ecosystems can potentially be achieved. To achieve this goal, we utilized the UBIWARE platform as a tool for proactive interoperability of distributed heterogeneous components. We presented an open environment for collaborative ecosystems with personalized cloud architecture in a sense of task- and domain-specific application development. We extend the cloud stack which allows us to organize collaboration between services and applications located in different clouds. All the components that can be considered as task and domain-specific are put to the Ecosystem Infrastructure layer (EIaaS). To increase flexibility and reliability of the applications and services created on top of PEI, we considered a semantic definition of Abstract Infrastructure. This way, it becomes possible to define context-dependent Data and services to be used by applications and services. Utilizing semantic web and multi-agent system approaches, the UBIWARE platform makes the ecosystem proactive and able to configure itself on-the-fly. This configuration happens depending on context and user needs, using the advantage of semantically adapted data, intelligent capabilities, and semantic abstract business process definition techniques.

 ISBN: 978-1-61208-153-3

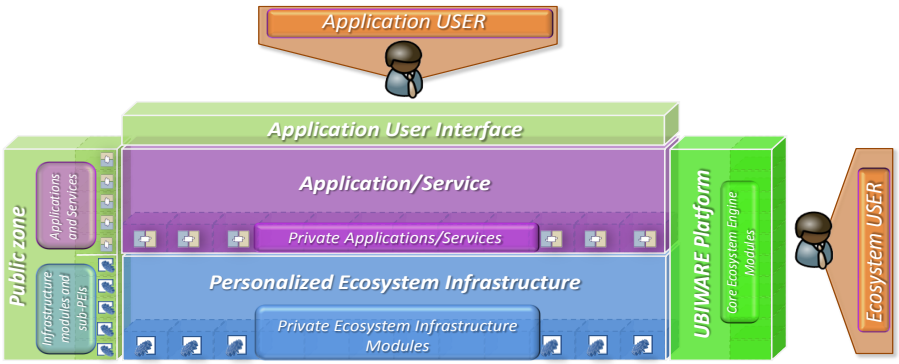


Figure 4. General structure of the Ecosystem.

ACKNOWLEDGMENT

This research is based on activities of the project under the Cloud Software program in TIVIT SHOK (funded by TEKES and consortium of industrial partners) and the cCloud project (Academy of Finland) in the Department of Mathematical Information Technology (University of Jyvaskyla, Finland). We are very grateful to the members of the Industrial Ontologies Group for a fruitful cooperation within this research topic.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing". Technical Report No. UCB/EECS-2009-28. EECS Department, University of California, Berkeley. February 10, 2009.

[2] D. Herris, "For Open Cloud Computing, Look Inside Your Data Center", 2010 [Online] URL: http://gigaom.com/2010/03/28/for-open-cloud-computing-look-inside-your-data-center/ [Accessed 30 June 2011]

[3] Bernard Golden. (2009, January) Computer World. [Online]. URL: http://www.computerworld.com/s/article/9126620/The_case_against_cloud_computing_part_one [Accessed 30 June 2011]

[4] Distributed Management Task Force, (2009, June) "Open Virtualization Format White Paper", Version 1.0.0, DSP2017.

[5] Semantic Web, 2001. [Online] URL: http://www.w3.org/2001/sw/ [Accessed 30 June 2011]

[6] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web", Scientific American 284(5), 2001, pp. 34-43.

[7] Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D. L., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T. R. and Sycara, K. (2002) DAML-S: Web Service Description for the Semantic Web. In: International Semantic Web Conference (ISWC), June 9th - 12th, Sardinia, Italy. pp. 348-363.

[8] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. 2002. "Importing the Semantic Web in UDDI." In Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web (CAiSE '02/ WES '02), Christoph Bussler, Richard Hull, Sheila A. McIlraith, Maria E. Orlowska, Barbara Pernici, and Jian Yang (Eds.). Springer-Verlag, London, UK, UK, 225-236.

[9] FIPA, "FIPA Interaction Protocol Library Specification Specification", FIPA00025, 2001. URL: http://www.fipa.org/specs/fipa00025/ [Accessed 30 June 2011]

[10] F. Curbera, M. Dufler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, "Unraveling the Web Services Web: An introduction to SOAP, WSDL and UDDI", Internet computing, 2002.

[11] F.M.T.Brazier, J.O. Kephart, H. Parunak and M.N. Huhns, "Agents and Service-Oriented Computing for Autonomic Computing: A Research Agenda," IEEE Internet Computing, vol. 13, no. 3, pp. 82-87, May/June 2009, doi:10.1109/MIC.2009.51

[12] N. Jennings, "An agent-based approach for building complex software systems". Communications of the ACM 44, 4, 2001, pp. 35–41.

[13] N. Jennings, "On agent-based software engineering", Artificial Intelligence 117(2), 2000, pp. 277–296

[14] UBIWARE Project [Online] URL: http://www.cs.jyu.fi/ai/OntoGroup/UBIWARE_details.htm [Accessed 30 June 2011]

[15] O. Kaykova, O. Khriyenko, D. Kovtun, A. Naumenko, V. Terziyan and A. Zharko, "General Adaption Framework: Enabling Interoperability for Industrial Web Resources", In: International Journal on Semantic Web and Information Systems, Idea Group, ISSN: 1552-6283, Vol. 1, No. 3, July-September 2005, pp.31-63.

[16] F. Bellifemine, G. Caire, A. Poggi and G. Rimassa, "Jade, A White Paper" [Online]
URL: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf [Accessed 30 June 2011]

[17] A. Katasonov and V.Terziyan, "SmartResource Platform and Semantic Agent Programming Language (S-APL)", In: P. Petta et al. (Eds.), Proceedings of the 5-th German Conference on Multi-Agent System Technologies (MATES'07), 24-26 September, 2007, Leipzig, Germany, Springer, LNAI 4687 pp. 25-36.

[18] O. Khriyenko, S. Nikitin and V. Terziyan, "Context-Policy-Configuration: Paradigm of Intelligent Autonomous System Creation", In: Joaquim Filipe and Jose Cordeiro (Eds.), Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS-2010), 8-12 June, 2010, Funchal, Madeira - Portugal, ISBN: 978-989-8425-05-8, pp. 198-205.

[19] P. Mell, T. Grance, "The NIST Definition of Cloud Computing (Draft)" Special Publication 800-145, January 2011.

[20] Amazon EC2, [Online] URL: http://aws.amazon.com/ec2/ [Accessed 30 June 2011]

[21] Microsoft Windows Azure [Online]
URL: http://www.microsoft.com/windowsazure/ [Accessed 30 June 2011]

[22] Amazon EC2 Publicly available datasets [Online] http://aws.amazon.com/publicdatasets/ [Accessed 30 June 2011]

 ISBN: 978-1-61208-153-3

Measuring Elasticity for Cloud Databases

Thibault Dory, Boris Mejías
Peter Van Roy
ICTEAM Institute
Univ. catholique de Louvain
dory.thibault@gmail.com, peter.vanroy@uclouvain.be,
boris.mejias@uclouvain.be

Nam-Luc Tran
Euranova R&D
Mont-Saint-Guibert, Belgium
namluc.tran@euranova.eu

***Abstract*—The rise of the Internet and the multiplication of data sources have multiplied the number of "Bigdata" storage problems. These data sets are not only very big but also tend to grow very fast, sometimes in a short period. Distributed databases that work well for such data sets need to be not only scalable but also elastic to ensure a fast response to growth in demand of computing power or storage. The goal of this article is to present measurement results that characterize the elasticity of three databases. We have chosen Cassandra, HBase, and mongoDB as three representative popular horizontally scalable NoSQL databases that are in production use. We have made measurements under realistic loads up to 48 nodes, using the Wikipedia database to create our dataset and using the Rackspace cloud infrastructure. We define precisely our methodology and we introduce a new dimensionless measure for elasticity to allow uniform comparisons of different databases at different scales. Our results show clearly that the technical choices taken by the databases have a strong impact on the way they react when new nodes are added to the clusters.**

***Keywords*-Cloud computing; key/value store; elasticity; NoSQL; Cassandra; mongoDB; HBase; Wikipedia.**

I. INTRODUCTION

Nowadays there are a lot of problems that require databases capable of storing huge quantities of unstructured data. The datasets are so big that they must be stored on several servers and, as new data are gathered and new users appear, it must be possible to extend the available storage and computing power. This can only be done by adding more resources into the cluster, e.g., adding servers. This addition is likely to have an impact on performance and therefore the goal of this paper is to present the definitions, methodology and results that are the outcome of our study of elasticity for a few chosen distributed databases. We also have defined a new dimensionless number to characterize the elasticity that ease the comparison between databases. The results are analyzed to explain the reason of some unexpected behaviors, but some stay unexplained for now.

This paper summarizes the results of a master's thesis [1]. We present first the detailed methodology and definitions, followed by the databases chosen, the measurement conditions, and the benchmark implementation. Finally, we present and analyze the measurement results.

II. STATE OF THE ART

The Yahoo! Cloud Servicing Benchmark [2] is the most well known benchmarking framework for NoSQL databases. It was created by Yahoo!. It currently supports many different databases and it can be extended to use various kinds of workloads. The benchmark used for the measurements presented here could have been implemented on top of YCSB as a new workload but it has not been for various reasons. The first reason is simplicity: it seemed easier to implement its functionalities directly instead of extending the big and far more complex YCSB where it would not have been so easy to control all the parameters. The second reason is that we wanted to explore the best methodology for measuring elasticity without being tied to the assumptions of an existing tool.

III. METHODOLOGY

A. Definitions

1) Performance: The performance is characterized by the time needed to complete a given number of requests with a given level of parallelization. The chosen levels of parallelization and number of requests used during the measurements are explained in the step by step methodology. In all the measurements of this article, we perform requests in batches called *request sets*. This allows us to decrease variability and improve accuracy in measurement time.

2) Elasticity: The elasticity is a characterization of how a cluster reacts when new nodes are added or removed under load. It is defined by two properties. First, the time needed for the cluster to stabilize and second the impact on performance. To measure the time for stabilization, it is mandatory to characterize the stability of a cluster, and therefore a measure of the variation in performance is needed. The system can be defined as stable when the variations between request set times are equivalent to the variations between request set times for a system known to be stable. That is, a system in which there are no data being moved across the nodes and when all the nodes are up and serving requests. These variations are characterized by the *delta time*, which is the absolute value of the difference

 ISBN: 978-1-61208-153-3

in time needed to complete a request set and the time needed to complete the previous request set. Concretely, for a given database, data set, request set, and infrastructure, the variability is characterized by the median value of the delta times and the system is said to be stable if the last X sets have a delta time smaller than the previously observed value. In this article we fix the value of X to 5, which gives satisfactory results for the measurements done.

We make the hypothesis that just after the bootstrap of the new nodes, the execution time will first increase and then decrease after an elapse of time. This is illustrated graphically in Figure 1 by the shape of the curve. In case the time needed for stabilization is very short, the average value and therefore the shape of the curve could be nearly unaffected by overhead related to elasticity, but at least the standard deviation will increase due to the additional work needed to move data to new nodes. It is important to take this standard deviation into account because highly variable latency is not acceptable. To characterize the elasticity in this article, we will take both the execution time and the standard deviation into account.

To characterize the elasticity with a single dimensionless number, we therefore propose the following formula:

$$Elasticity = \frac{A + B}{(Rt1 + Rt2)^2 * F} \quad (1)$$

Here A and B are the surface areas shown in Figure 1, where A is related to the execution time increase and B is related to the standard deviation, $Rt1$ is the average response time of one request for a given load before the bootstrapping of the new nodes, $Rt2$ is the average response time once the cluster has stabilized with the same load applied. Finally, F is the factor to suppress the dependency to the number of requests per node in the cluster. It is given by

$$F = \frac{Number\ of\ requests}{cluster\ size} \quad (2)$$

In all the measurements of this article, we assume that $N = M$, that is, we double the number of nodes.

The triangular area defined by the edges (Rt1,Rt2), (Bootstrap,Stable), and (Rt1,Stable) is not counted because even for perfect elasticity this triangle will exist as a performance ramp from level $Rt1$ to $Rt2$. The area $A + B$ is then purely due to elasticity and has a dimension of time squared. The value $Rt1 + Rt2$ are both inversely proportional to the average performance and have a dimension of time. The elasticity is therefore the ratio of the elastic overhead $A + B$ to the absolute performance $(Rt1 + Rt2)^2 * F$ and is a dimensionless number. The division by F removes the scaling factor of the size of the request set (e.g., the 10000 mentioned above).

B. Step by step methodology

Figure 3 illustrates the step by step methodology used during the tests. It is based on the following parameters : N the number of nodes, R the size of a request set and r the percentage of read requests. In practice, the methodology is defined by the following steps:

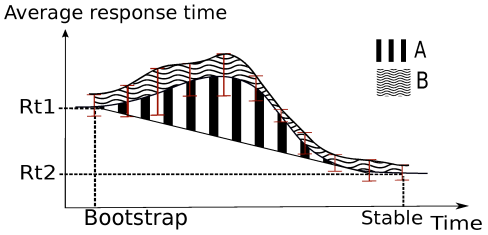


Figure 1. Surface areas used for the characterization of the elasticity

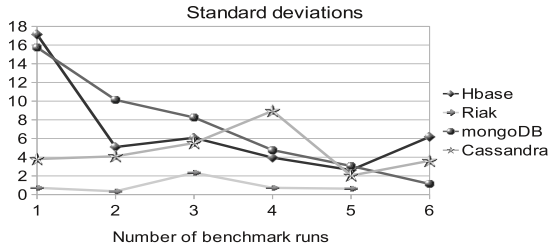


Figure 2. Observed standard deviations for 10000 requests with 80% reads

1) Start up with a cluster of $N = 6$ nodes and insert all the Wikipedia articles.
2) Start the elasticity test by performing request sets that each contain $R = 10000$ requests with $r = 80\%$ read requests and as many threads as there are nodes in the cluster when the elasticity test begins. The time for performing each request set is measured. (Therefore the initial request sets execute on 6 threads each serving about 1667 (≈10000/6) requests.) This measurement is repeated until the cluster is stable, i.e., we do enough measurements to be representative of the normal behavior of the cluster under the given load. We then compute the median of the delta times for the stable cluster. This gives the variability for a stable cluster.
3) Bootstrap new nodes to double the number of nodes in the cluster and continue until the cluster is stable again. During this operation, the time measurements continue. We assume the cluster is stable when the last 5 request sets have delta times less than the one measured for the stable cluster.
4) Double the data set size by inserting the Wikipedia articles as many times as needed but with unique IDs for each insert.
5) To continue the test for the next transition, jump to step (2) with a doubled number of requests and a doubled number of threads.

 ISBN: 978-1-61208-153-3

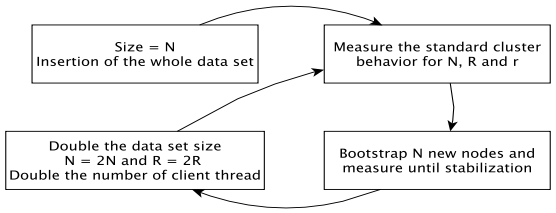


Figure 3. Step by step methodology

C. Justification of the methodology

One approach to characterize the variability is to use the standard deviation of request set times and a statistical test to compare the standard deviations. However, our experience shows that the standard deviation is too sensitive to normal cluster operations like compaction and disk pre-allocations. Figure 2 shows that the standard deviation can vary more than a factor of 4 on a stable cluster made of six 4GB Rackspace instances. This is why we use the delta time characterization instead. Because it is based only on the average values, it tends to smooth these transient variations. The median of all the observed delta times is used instead of the average to be less sensitive to the magnitude of the fluctuations.

Remark that we still use the standard deviation as part of the characterization of the elasticity. This characterization captures all the important information about the elasticity (time needed to stabilize, loss of performance, and variability) with the two surface areas (A and B) and normalizes it into a dimensionless number that can be used for comparisons.

Finally, the number of observations needed to have an idea of the normal behavior of a database cluster cannot be fixed in advance. Experience shows that, from one system to another, high variability in performance can arise at different moments. This variability is mainly due to the writes of big files on the disk, like compactions, disk flushes, and disk pre-allocations, all of which can happen at very different moments due to the randomness of the requests and the technical choices made by each database. The variability has a measurable result that will be discussed in the result section. In practice, the observations were stopped when the performance and standard deviation got back to the level observed before the compactions or disk pre-allocations happened.

D. Properties of the methodology

All the parameters are updated linearly in respect to the number of nodes that are bootstrapped in the elasticity test, but all those parameters are not updated at the same time during the methodology. However, the measurements obey several invariants, which are given in italics below.

The size of the request sets is always increased at the same time as the number of client threads, which implies that on the client side, *the number of requests done by each client thread is independent of cluster size*. On the database nodes, there are two different situations. When the elasticity test begins and during the entire first phase of the test, as many threads as there are nodes in the cluster are started, and therefore, *the amount of work done by each node in the cluster is independent of cluster size*.

The second phase starts when new nodes are bootstrapped and lasts as long as the cluster needs time to stabilize. During this time, the amount of work done by the nodes already present in the cluster should decrease progressively as newly bootstrapped nodes start to serve part of the data set. In a perfect system, all the nodes in the enlarged cluster should eventually do an amount of work that has decreased linearly regarding to the number of nodes added in the cluster. It is important to note that the eventual increase in performance that would appear at this point is not a measure of the scalability as defined earlier. This is due to the fact that, at this point, neither the data set nor the number of client threads has been increased linearly regarding to the number of nodes added. The goal of the elasticity test is only to measure the impact of adding new nodes to a cluster that serves a constant load.

Once the elasticity test ends, the size of the data set inserted into the database is increased linearly according to the number of nodes just added. As a consequence, during the next round of the elasticity test the amount of data served by each node has not changed. Therefore, *once the number of threads is increased at the beginning of the next elasticity test, the total amount of work (number of requests served and data set size) per database node does not change*.

IV. Databases chosen

The three databases selected for this study are Cassandra [3] 0.7.2, HBase [4] 0.90.0 and mongoDB [5] 1.8.0 because they are popular representatives of the current NoSQL world. All three databases are horizontally scalable, do not have fixed table schemas, and can provide high performance on very big data sets. All three databases are mature products that are in production use by many organizations [6] [7] [8]. Moreover, they have chosen different theoretical approaches to the distributed model, which leads to interesting comparisons.

All three databases are parameterized with a common replication factor of 3 and strong consistency for all requests in order to ensure a comparable environment on both the application and server side.

V. Measurement conditions

This section describes the budget allocated, the infrastructure and the data set used as well as the benchmark implementation.

A. Budget and infrastructure

We first explain our decisions regarding budget and infrastructure, since they affect the whole measurement process. The budget allocated for all the tests of this article is 800 euros. We choose to use the 4 GB cloud instances from Rackspace. This allowed us to perform measurements at full load for up to 48 nodes with all three databases.

Using cloud instances instead of dedicated servers has consequences on performance. Indeed several instances are sharing the same physical computer and therefore using cloud instances adds variability, depending on the resources usage of the other instances, to the measurements.

Finally, the data set per node has been chosen large enough to be sure that the subset of the data stored on each node could not fit entirely in RAM. It is important to remind the reader that the databases studied here are made to handle "Bigdata" problems where typically it would cost too much to fit all the dataset into memory. Therefore, with a focus on "Bigdata", it is natural to consider databases that cannot fit into memory.

B. Data set

The data set is made of the first 10 million articles of the English version of Wikipedia. They can be downloaded as a single archive provided [9] by Wikimedia itself. The dump was downloaded on March 7, 2011 and it takes 28 GB of disk space.

C. Benchmark implementation

The benchmark is written in Java and the code source is available as a GitHub repository under a GPL license [10]. The benchmark framework is used to automate the parts of the methodology that concerns the insertion of articles as well as applying the load and computing the results.

To approximate the behavior of Wikipedia users, the requests are fully random. Meaning that for each request, a uniform distribution (the Java class java.util.Random, initialized without seed) is used to generate a integer in the range of the IDs of the inserted documents. Then, after the article has been received by the client, a second integer is generated using a uniform distribution to decide if the client thread should update this article or not. Update simply consist in appending the string "1" at the end of the article.

VI. RESULTS

Figures 4 to 11 give graphs showing the elastic behavior of all databases at all transition sizes. These graphs represent the measured average time in seconds needed to complete a request set versus the total execution time in minutes. Standard deviations are indicated using symmetric (red) error bars, but it is clear that this does not imply improved performance during stabilization (downward swing)! The first part of each graph shows the normal behavior of the cluster under load. The first arrow indicates when the new nodes are bootstrapped and the second arrow indicates when all the nodes report that they have finished their data transfers. The graphs also show the standard deviations and the two thin (red) lines show the acceptable margins for the delta time that are computed from the first part of the graph.

Table I
STABILIZATION TIME (IN MINUTES, LOWER IS BETTER)

Database	Cluster size variation	Data tr. time	Add. time	Total time
Cassandra	6 to 12 nodes	113	28	141
HBase	6 to 12 nodes	3.3	9	12.3
mongoDB	6 to 12 nodes	172	11	183
Cassandra	12 to 24 nodes	175	26	201
HBase	12 to 24 nodes	3.2	14	17.2
mongoDB	12 to 24 nodes	330	22	352
Cassandra	24 to 48 nodes	86	2	88
HBase	24 to 48 nodes	8	37	45

Table II
ELASTICITY (LOWER IS BETTER)

Database	Cluster old and new size	Score
Cassandra	6 to 12 nodes	1735.
HBase	6 to 12 nodes	646.
mongoDB	6 to 12 nodes	4626.
Cassandra	12 to 24 nodes	1044.
HBase	12 to 24 nodes	70.
mongoDB	12 to 24 nodes	4009.
Cassandra	24 to 48 nodes	3757.
HBase	24 to 48 nodes	73.

Table I shows the stabilization times (in minutes), which consists of the times for all the nodes to finish their data transfers as well as the additional times needed for the whole cluster to achieve stabilization once all the data transfers are done. The time needed to finish all the data transfers is measured using tools provided by the databases to monitor data transfers across the cluster. The additional time to achieve stabilization is the time when the cluster reaches a stable level minus the time when the cluster reported that all the data transfers were done.

Table II shows the dimensionless elasticity scores according to the definition in Section III-A. In practice, the curves have been approximated by cubic splines interpolating the given point and those splines have been integrated using a recursive adaptive Simpson quadrature. The lower the elasticity score, the better the elasticity.

A. Analysis of the results

Analysis of the measurement results is made more difficult by the variability of the cluster performance under load before new nodes are bootstrapped. Those variabilities are very clear for Cassandra on Figure 4 and 5, for HBase on Figure 8 and for mongoDB on Figure 11. These big variabilities in performance have different origins but all of

 ISBN: 978-1-61208-153-3

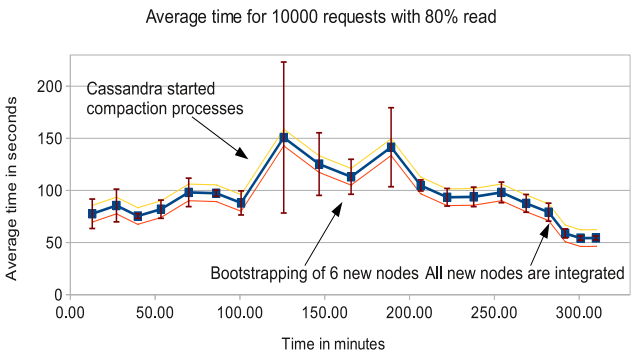


Figure 4. Elasticity under load Cassandra (6→12 n.)

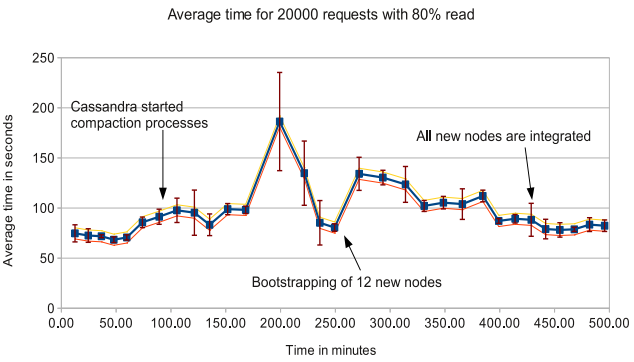


Figure 5. Elasticity under load Cassandra (12→24)

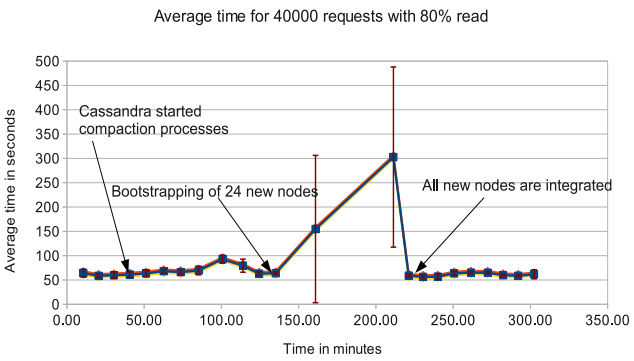


Figure 6. Elasticity under load Cassandra (24→48)

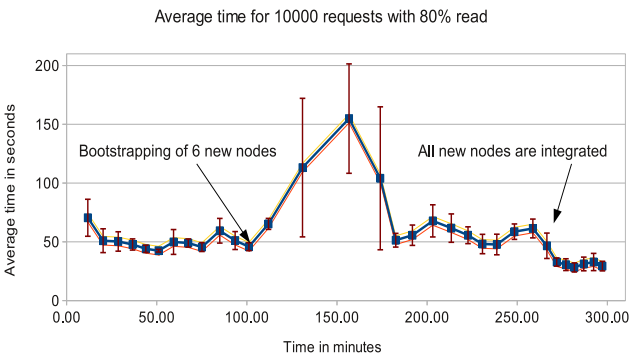


Figure 7. Elasticity under load mongoDB (6→12 n.)

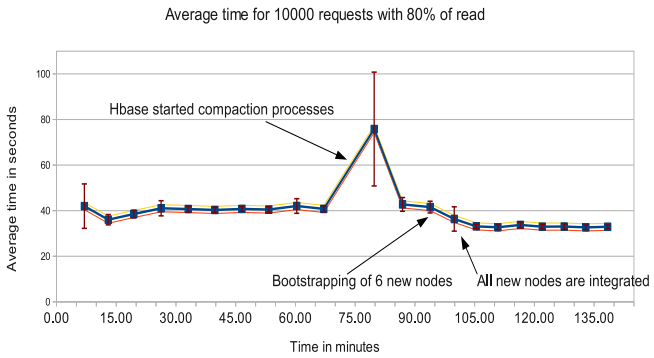


Figure 8. Elasticity under load HBase (6→12 nodes)

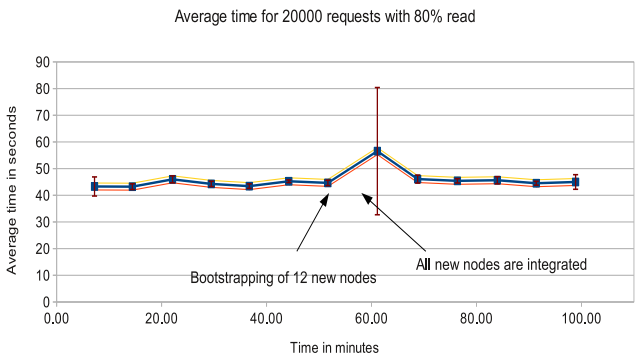


Figure 9. Elasticity under load HBase (12→24 n.)

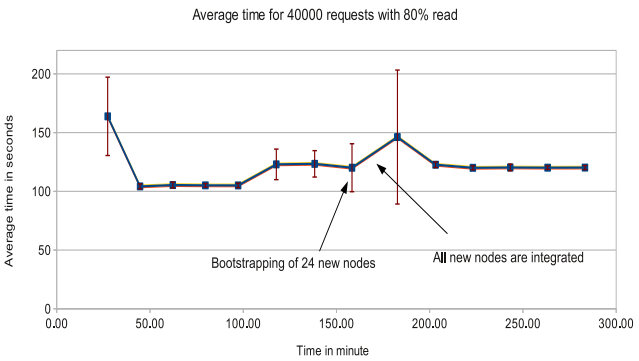


Figure 10. Elasticity under load HBase (24→48 n.)

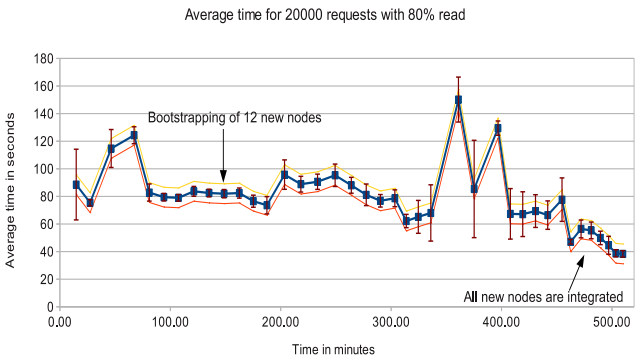


Figure 11. Elasticity under load mongoDB (12→24)

them have the same immediate cause: the writing of at least one big file on the disk. First, for Cassandra and HBase, big writes are triggered when compactions or disk flushes occur. The flushes occur when the *memtable* is full, and compactions follow after a few flushes [3] [11]. A load constantly updating data will, sooner or later, trigger compactions and disk flushes. Second, for mongoDB: big writes are only triggered when disk pre-allocations occur. mongoDB uses the mmap function provided by the operating system instead of implementing the caching layer itself, meaning that it is the OS itself that decides when to flush. mongoDB pre-allocates big files when it needs new storage space instead of increasing the size of existing files. In practice, mongoDB allocates space on disk each time it needs to store a lot of new data, like during chunks movements or big inserts.

Note that compaction is part of normal database operation that is needed both when handling client requests and when handling bootstrapped nodes during elastic growth. So we make no effort to remove the compaction cost from our measurement of elasticity. It is important to note that the only requests that will be slowed down by the writing of big files are the ones sent to nodes currently writing those big files. Therefore, when the number of nodes increases, the probability to send requests to a node currently doing a lot of I/O decreases. Indeed, looking at Figure 6 for Cassandra and Figure 10 for HBase, we observe the overall performance is more stable for bigger clusters.

On this infrastructure, the technical choice taken by mongoDB to make small but frequent disk flushes leads to less variability in performance than Cassandra. One could wonder what is the cause of the variability observed at the beginning of the chart on Figure 11 for mongoDB as no new nodes were bootstrapped at this time. This is caused by the fact that during the insertion, some nodes stored more chunks than the other and only started to distribute them across the cluster during the start of the test.

The variability of HBase performance is quite different from Cassandra even if their technical choices are close. By default the *memtable*'s size of Cassandra is 64MB and HBase is 256MB, leading to more frequent flushes and compactions for Cassandra but on the other hand, the compactions are also made on smaller files for Cassandra. The effect of compactions is only visible on Figure 8 and not on Figure 9 nor on Figure 10. This could be because the number of nodes is bigger and the effect of the compaction impacted a smaller number of requests.

Finally, there are no results for mongoDB going from 24 to 48 nodes. This is due to several problems encountered with mongoDB during the insertion of the articles. Starting with a cluster of size 12, mongod processes started to crash because of segmentation faults that caused data corruption, even with the journaling enabled. This problem was temporarily fixed by increasing the maximum number of files that can be opened by the mongod processes. But for 24 nodes, the segmentation faults were back with another problem. Eight threads were used to insert the articles, each of them making its requests to a different mongos router process, but all the writes were done on the same replica set. The elected master of this replica set was moving the chunks to other replica sets but not as fast as it was creating them, leading to a disk full on the master and at this point all the inserts stopped instead of starting to write chunks on other replica sets.

Elasticity

For the analysis of the elasticity results, we first explain some technical choices of the databases. The databases can be divided in two groups depending on the kind of work that the databases have to do when new nodes are added.

In the first group, which contains Cassandra and mongoDB, the databases have to move the data stored on the old nodes to the new nodes that just have been bootstrapped. In the case of a perfectly balanced cluster, that means moving half of the data stored on each of the old nodes to the new ones.

In the second group, which in this article contains only HBase, the database (HBase itself) and the storage layer (Hadoop Distributed File System [12]) have been separated to be handled by two distinct entities in the cluster. At the HBase level, each *region server* is responsible for a list of *regions* meaning that it has to record the updates and writes into *memtables* and it also acts as a cache for the data stored in the HDFS level. When new nodes running both a *region server* and a *datanode* are bootstrapped, the new *region servers* will start to serve a fair share of the available regions but the existing data will not be moved to the new *datanode*. Therefore there will be not big data transfer on new node bootstrapping.

The fact that HBase does not have to move all the data appears very clearly on the charts. HBase only needs a few minutes to stabilize while Cassandra and mongoDB take hours. The technical choices taken by HBase are a big advantage in terms of elasticity for this methodology. In Figures 9 and 10, HBase moves new regions to the region servers quickly, but the new region servers still need to load data, this is why the peaks happen *after* the new nodes are integrated.

For Cassandra, the impact of bootstrapping new nodes is less than the variability induced by normal operations for clusters smaller than 24 nodes, after that the impact is much more important than the usual variability of the cluster's normal operations. Note that the performance of Cassandra only improves when all the nodes are fully integrated because new Cassandra nodes only start serving requests when they have downloaded all the data they should store. The time needed for the cluster to stabilize increased by 54% between the tests of 6 to 12 nodes and 12 to 24 nodes, while it decreased by an impressive 50% between the

 ISBN: 978-1-61208-153-3

tests of 12 to 24 nodes and 24 to 48 nodes. The nonlinear increase is due to the fact that new nodes know which are the old nodes that should send them data thanks to the nodes' *Tokens*, leading to simultaneous data transfers between nodes across the cluster. On the other hand, the 50% decrease is still to be explained.

With mongoDB, the variability in performance added by the bootstrap of new nodes is much bigger than the usual variability of the cluster. Unlike Cassandra, newly bootstrapped mongoDB nodes start serving data as soon as complete chunks have been transferred. Therefore newly bootstrapped nodes that serve the few chunks already received will pre-allocate files to make room for the next chunks received leading to a lot of requests potentially served by nodes writing big files to disk and therefore degrading the performance. The time needed for the cluster to stabilize increased by 92% between the tests of 6 to 12 nodes and 12 to 24 nodes. This almost linear increase is due to the fact that there is only one process cluster wide, the balancer, that moves the chunks one by one.

The elasticity scores give an accurate idea of the elasticity performance of the databases, disadvantaging databases for the height of the peak and the time needed before stabilization. Note that, for HBase, the decreasing score is due to relatively smaller peaks as the cluster grows and the last one can also be explained by the fact that the performance is less, so the elasticity is relatively better with respect to this worse performance. Globally, the elasticity score also shows the advantage of HBase for clusters of all sizes.

VII. Conclusions and future work

The main conclusion of our measurements is that the technical choices taken by each database have a strong impact on the way each of them reacts to the addition of new nodes. For this definition of elasticity, HBase is a clear winner. This is due to its technical choices and architecture leading to much less data transfer on node addition.

This article gives measurement results only for systems that scale up, and not for systems that scale down. We decided for this limitation because we wanted to explore in detail what happens when a system scales up, and experience has borne out that these measurements are sufficiently surprising and technically difficult to carry out. We expect that future work measuring systems that scale down will give a fresh set of surprises.

We plan to continue expanding the cluster sizes to see if the current trends will last or if some other bottleneck will appear at some point. For example it would be interesting to see if it is possible to reach any bottleneck with systems, like HBase and mongoDB, using a centralized approach to store the localization information in the cluster.

We also intend to solve the problems encountered for mongoDB to measure its performance optimally. Then it would also be interesting to do the same tests but with different values for the parameters like the read-only percentage or using a different statistical distribution. We plan to extend our coverage of the measurement space and continue to refine our new elasticity measure. Finally, we intend to measure the performance of other databases like Riak and distributed caches like infinispan and ehcache.

VIII. Acknowledgements

We would like to thank Euranova for the idea of studying the elasticity of distributed databases and for their support that helped us improve this article [13]. Special thanks go to the director of Euranova R&D, Sabri Skhiri, for his insightful comments. We also thank Ivan Frain and Samuel Richard for their constructive comments.

References

[1] T. Dory, "Study and Comparison of Elastic Cloud Databases: Myth or Reality?" pldc.info.ucl.ac.be, Programming Languages and Distributed Computing (PLDC) Research Group, Université catholique de Louvain, Tech. Rep., Aug. 2011.

[2] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *SoCC*, 2010, pp. 143–154.

[3] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 35–40, April 2010. [Online]. Available: doi.acm.org/10.1145/1773912.1773922

[4] Apache HBase, "Frontpage," hbase.apache.org, Jun. 2011.

[5] mongoDB, "Frontpage," www.mongodb.org, Jun. 2011.

[6] Apache Cassandra, "Frontpage," cassandra.apache.org, Jun. 2011.

[7] HBase Wiki, "PoweredBy," wiki.apache.org/hadoop/Hbase/PoweredBy, Jun. 2011.

[8] mongoDB, "Production Deployments," www.mongodb.org/display/DOCS/Production+Deployments, Jun. 2011.

[9] Wikipedia, "Latest dump of Wikipedia English," download.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, Jun. 2011.

[10] GitHub, "Wikipedia-noSQL-Benchmark," https://github.com/toflames/Wikipedia-noSQL-Benchmark/, Jun. 2011.

[11] 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," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, 2008.

[12] 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. [Online]. Available: dx.doi.org/10.1109/MSST.2010.5496972

[13] Euranova, "Frontpage," euranova.eu, Jun. 2011.

 ISBN: 978-1-61208-153-3

Facilitating Bioinformatic Research with Mobile Cloud

Jinhui Yao[1,2], Jingyu Zhang[2], Shiping Chen[1], Chen Wang[1], David Levy[2]

[1]*Information Engineering Laboratory, CSIRO ICT Centre, Australia*
{Firstname.Lastname}@csiro.au
[2]*School of Electrical and Information Engineeering, University of Sydney*
{jinhui, jyzhang, dlevy}@ee.usyd.edu.au

***Abstract*—In this paper, we propose a concept of mobile-cloud by combining mobile and cloud together in a bioinformatic research application scenario. A mobile-cloud framework is developed, which facilitates the use of mobile devices to manipulate and interact with the scientific workflows running in the Cloud. In this framework, an independent trusted accountability service is used to provide data provenance and enforce compliance among the participants of a bioinformatic workflow. We have implemented a prototype which allows the bioinformatic workflow design and participation using mobile devices. We prove the concept of mobile-cloud with the prototype and conducted performance evaluation for the significant points of the bioinformatic workflow.**

Keywords-cloud computing; accountability; service orieanted architecture; mobile cloud.

I. INTRODUCTION

The emergence of computing resource provisioning known as the Cloud has revolutionized the modern day computing. It has provided a cheap and yet reliable outsourcing model for whoever with huge needs for computing resources. Given the fact that many scientific breakthroughs need to be powered by advanced computing capabilities that help researchers manipulate and explore massive datasets [1], Cloud offers the promise of "democratizing" research as a single researcher or small team can have access to the same large-scale compute resources as large, well-funded research organizations without the need to invest in purchasing or hosting their own physical infrastructure.

On the other hand, the concept of Service Oriented Architecture (SOA) allows flexible and dynamic collaborations among different service providers. A service can either directly be used for its mere function or be composed with other services to form new value-added workflows [2]. Through SOA, scientific workflows can be used to bring together these various scientific computing tools and compute resources offered as services in the Cloud to answer complex research questions. Workflows describe the relationship of the individual computational components and their input and output data in a declarative way. In astronomy, scientists are using workflows to generate science-grade mosaics of the sky [3], to examine the structure of galaxies [4]. In bioinformatics, researchers are using workflows to understand the underpinnings of complex diseases [5].

In scientific workflows, certain critical steps need the participation of respective research personnel or experts. For example, how the workflow should be designed and which scientific tools need to be involved must be decided by the expert in the area. And some complex patterns generated from the experiments need to be visually inspected by the scientists who will based on their domain knowledge and experience determine the next a few steps for further analysis. In this regard, it is highly desirable that scientists can have easy access to the services in the Cloud so that they can design and participate in the workflows efficiently.

To address the above need, with the impressive advanced in the technology, we believe using mobile devices can be an ideal solution. The processes in a workflow can be thoroughly integrated with portable devices. All activities are decided and monitored on time from the way that fit the human environment instead of forcing users to passively accept the computing results from cloud service. In this paper, we propose a novel design which facilitates the use of mobile devices to manipulate and interact with the scientific workflows running in the Cloud. In our system, the users can choose the services in the Cloud to form the workflows via their mobile devices, and each mobile device can serve as one service node to be involved in the workflows designed. The contribution of this paper is two fold, we first elaborate our framework enabling mobile devices to compose and participate in the workflows running in the Cloud. Then, we further propose our approach to incorporate accountability into the system in order to enforce compliance and provide data provenance.

II. THE APPLICATION SCENARIO

In the area of gene research, the recent development of the microarray technology [6] have led to rapid increase in the variety of available data and analytical tools. Some recent surveys published in Nucleic Acids Research describes 1037 databases [7] and over 1200 tools [8]. The analysis of microarray data commonly requires the biologist to query various online databases and perform a set of analysis using both local and online tools.

To illustrate with an example, here we explain the research study of the genetic cause of colorectal cancer, i.e., identify the genetic variation in human DNA that makes people susceptible to colorectal cancer. By studying the functions of the genes involved, biologists can have a better understanding of the cancer and find possible cure. The first step in this study is to perform experiments on mice, which share more than 90% DNA with human. Microarray experiments are performed on

 ISBN: 978-1-61208-153-3

both cancerous and healthy mouse colon tissues [6]. By comparing the results from mice with and without colorectal caner, biologists can identify candidate genes that may cause the cancer. Further analysis—such as searching for the functions known to these genes—are commonly performed to examine whether and how the candidate genes relate to the colorectal cancer. The followings are some standard analysis that are required for the study of microarray experiment results:

Quality Control. The raw microarray result data are processed, visualized and inspected by an expert, who can identify errors and discard the experiment.

Normalization. Microarray results from different samples need to be normalized before any meaningful comparison can be conducted.

Gene Differentiation. By contrasting the results from cancerous and healthy tissues, differentially expressed genes—genes that are active in cancer but not healthy tissue or vice versa—are identified.

Gene Study. Most differentially expressed genes are further studied to understand the biological foundation of the disease. Many resources are available to study these genes, such as the gene sequence, pathway database (e.g., KEGG), and gene function ontology. Experts need to be involved to make good decision as which study to conduct and which database to use.

We can see that the four standard analysis procedures we listed above not only can be extremely computing intensive but also require some decision making from the research scientists or experts at certain critical steps (e.g., quality control). It easily follows that, a viable approach to conduct such researches must utilize certain computing platform that has enormous computing capacity, yet research scientists can easily interact with the platform and the computing process conducted. This is essentially the reason for which we promote the "Mobile Cloud" - a composition of the Cloud, and the mobile devices – to be a suitable paradigm for complicated bioinfomatics researches.

III. A MOBILE-CLOUD SYSTEM FOR BIOINFORMATIC REARCH

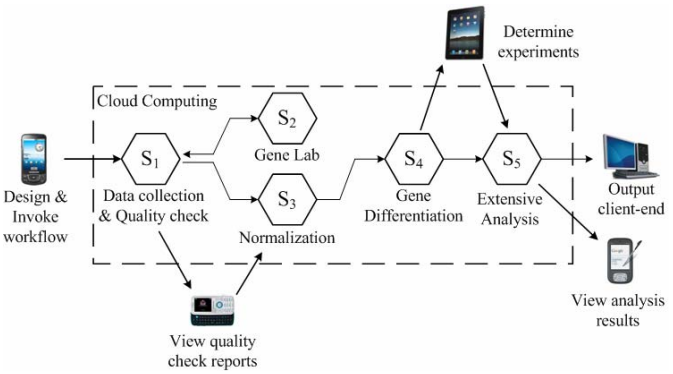


Figure 1. Overview of the proposed system

As we have established in previous sections, we propose to compose the Cloud and the mobile devices to conduct complex bioinformatice researches. The bioinformatic research scenario we chose is the study for the cause of colorectal cancer. Fig. 1 shows our proposed system with this research scenario.

In the Cloud, different computing intensive gene research tools are deployed by different research bodies and provided as services. Outside the Cloud, research scientists or gene analysts locate the desired services in the Cloud, and use them to compose a workflow for studying the cancer. The workflow starts with retrieving raw microarray data from the nominated "Gene Lab", after going through a sequence of processing, produce the final output to send to the "Output client-end". Multiple research personnel may be involved in the workflow, they participate in the workflow by using mobile devices to invoke or receive output from the services.

Our argument for using mobile devices to design and participate in the workflows is intuitive. As mentioned, in the workflow there are "critical steps" that require decision making by experts in the respective area, in order to continue the process. For example, after the quality check, an important decision needs to be made about whether the quality of the raw data suffices the requirements of the experiment. The experiment should be paused before the expert in charge has reviewed the quality check reports and confirmed the usability of the raw data. Therefore, mobile devices are indeed ideal for this task for its outstanding mobility compared to desktop computers or even laptop computers, i.e., one can freely use his mobile devices while waiting in a queue, on a bus, or even walking. Further, given the recent impressive advances in the mobile technology, the computing capability of mobile devices - however limited compared to desktops or laptops - is more than enough to run basic UI or display data sets and processing reports. Therefore, we believe mobile devices such as smart phones or tablet computers are indeed ideal to be used as light client-end to drive the heavy bioinformatic research workflows in the Cloud.

To enable mobile devices to construct and participate in the workflows running the Cloud, we have developed the Mobile Cloud middleware layer (MC-layer) to facilitate these. Fig. 2 provides an overview of the architecture, which consists of a user interface (residing on mobile devices), a Cloud environment containing various services and a middleware layer consists of three function units. Their respective functionalities are summarized as follows:

- **Cloud Environment** provides various services deployed by respective providers. The services have registered their access end point with the MC-layer.
- **Service Repository** stores the informtion about the services in the Cloud that has registered with it. Once a search request is received, it will find the best service or workflow that satisfy the functional and non-functional requirements specified.
- **Service Composition** is responsible of composing individual services into workflows.
- **Service Execution** conducts two jobs: (a) orchestrating workflows; (b) invoking Web services.
- **User Interface** allows users to register, design workflows and participate in a running workflow.

 ISBN: 978-1-61208-153-3

For mobile devices to construct workflows, they first need to send a search request to the Service Repository in order to get a list of the services/workflows they are looking for. A convenient UI has been implemented on the mobile devices to allow the users to easily design the workflows using the services listed by the Service Repository (the UI will be elaborated in the evaluations). Once the workflow have been designed, a representative XML based description script is generated to be submitted to the Service Composition unit. The Service Composition unit thus according to the script, composes the services to form the desired workflows. The services can be composed in two ways: i) centrally composed, where the MC-layer invokes the services in the sequence designed by the user; and ii) remotely orchestrated, where certain orchestration scripts such as BPEL will be generated and distributed to all the services involved for deployment.

In our system, the workflow designed by the users is an abstract workflow, that is, the users only need to specify the type of service needed, and the MC-layer will search its service reporsitory and select the best suited ones according to the user's specifications. Table 1 gives a sample of the workflow description script. As it is developed based on the BPEL, "sequences" and "flows" are used to specify serial and parallel composition, and "Actions" are used define the invocation operations. The sample describes the first half of the gene analysis workflow in Fig. 1. In some actions, the endpoint is set to be "OPTIMAL". This is to tell the Service Composition unit to choose the best suited services.

TABLE I. SAMPLE WORKFLOW DESCRIPTION SCRIPT

```
<sequence name="main">
   <Action operation="start" invoker="client" endpoint="QualityCheck"
   type="send&forget".../>
   <Action operation="fetchGene" invoker="QualityCheck"
   endpoint="GeneLab" type ="send&receive".../>
   <flow>
      <Action operation="sendForApproval" invoker="QualityCheck"
      endpoint="OPTIMAL" type ="send&forget".../>
      <Action operation="normalization" invoker="QualityCheck"
      endpoint="OPTIMAL" type ="send&forget".../>
   </flow>
   …
</sequence>
```

As we have established in our system design, mobile devices will be involved in the workflows as web services. To facilitate this, we created a customized web service engine to run on the mobile devices. Using this engine, mobile devices can both send and receive service requests, as well as interpreting the workflow description scripts delivered by the MC-layer. Once a user has designed and submitted a workflow, the workflow description script will be forwarded to the research personnel that are involved. The mobile devices they are using will interpret the workflow script and save the workflow logic. When a service request is received during the execution of the workflow, the UI will allow the user to view the content (e.g., quality check reports) and provide the list of the services that the user should send output request to according to the workflow logic (e.g., normalization services). For the technical details of the MC-layer, please refer to our previous publications about the Web Service Management System (WSMS) [12].

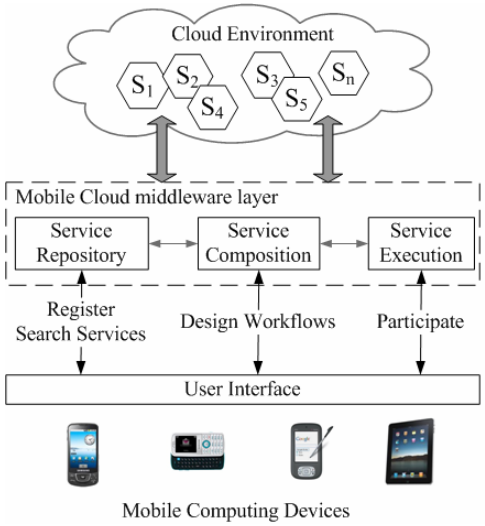


Figure 2. Overview of Mobile Cloud architecture

IV. ACCOUNTABILITY FOR COMPLIANCE AND PROVENANCE

The workflows in the Cloud are constructed using services provided by different parties who barely know each other. The correctness of the resultant workflow relies on the individual correctness of all participators. That is, if the service is compliant to the pre-defined workflow logic, or Service Level Agreement (SLA). The scientific integrity of the gene analysis results will be highly questionable if the services involved can act willy-nilly and get away with processing errors.

On the other hand, for scientific experiments not only the resultant data are considered, the steps of how these data are derived along the process can also be very valuable. It has been widely realized that data provenance plays an important role in the scientific researches [13]. It follows that, a mechanism is clearly needed to preserve the intermediate data forms generated by different services during the execution of the workflow, for compliance monitoring and provenance of the analysis results. We regard this mechanism critical for the viability of the paradigm we have proposed. In this section, we illustrate our design to incorporate accountability into the "Mobile Cloud" to address these issues.

Accountability can be interpreted as the ability to have an entity account for its behaviors to some authorities [9]. This is achieved by binding each activity conducted to the identity of its actor with proper evidence [10]. Such binding should be achieved under the circumstance that all actors within the system are semi-trusted. That is, each identified actor may lie according to their own interest. Therefore, accountability should entail a certain level of stringency in order to maintain a system's trustworthiness. Below, we identify several desirable properties of a fully accountable system:

- **Verifiable:** The correctness of the conducted process can be verified according to the actions and their bindings recorded.
- **Non-repudiable:** Actions are bound to the actors through evidence, and this binding is provable and undeniable.

 ISBN: 978-1-61208-153-3

- **Tamper-evident:** Any attempt to corrupt to recorded evidence inevitably involves the high risk of being detected.

We illustrate our proposed approach in Fig. 3. In our approach, accountability can be incorporated into activity-based workflow by requiring the entity conducting the process to log non-disputable evidence about the activities in a separate entity. In the figure, after incorporating accountability into an ordinary process, entity A is now required to perform logging operations before and after conducting the activity in its process. The evidence is logged in a separate entity - entity B - so that entity A cannot access the logged evidence. The evidence needed to be logged should contain enough information to describe the conducting activity. In our simple example, which is intuitive enough, the evidence should include the states of the factors concerning the start of the activity (e.g., the input variables) and the factors concerning its completion (e.g., the output value).

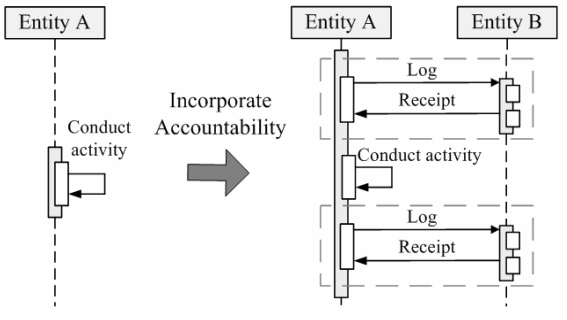


Figure 3. Example of incorporating accountability into process

The logging operations require the employment of PKI in all involved service entities. Each of them has its own associated public-private key pair issued by certificated authorities. The logging operations are as follows:

1. The logger (entity A) signs the evidence (E) by its private key (K_{A-}) to create a digital signature of the evidence (S_A).
2. The evidence and its signature are then logged in a separate entity (entity B).
3. When received, entity B creates a receipt by signing entity A's signature with entity B's private key (K_{B-}).
4. Lastly, the receipt (S_B) is sent back to the logger (entity A) in the reply.

Assuming the digital signature is un-forgeable, the signed evidence in entity B can be used to verify entity A's compliance; and yet any corruption or deletion applied to the evidence will be discovered using the receipt received by entity A. Under the circumstance that neither of the service entities is trusted; and assume they will not conspire to cheat, this structure manages to ensure the proper preservation of evidence associated with the process conducted.

To have the separate entity B to preserve the evidence, we propose to have special service nodes, dedicated to provide accountability to all underlying services involved in the workflow. Those special nodes are referred to as the accountability service (AS) nodes. Fig. 4 shows the structure. All the mobile devices, service nodes in the Cloud as well as local computing nodes that are involved in the workflow, register with AS nodes and submit evidence during the execution of the workflow. The implementation details of the incorporation of accountability have been elaborated in our previous work [11].

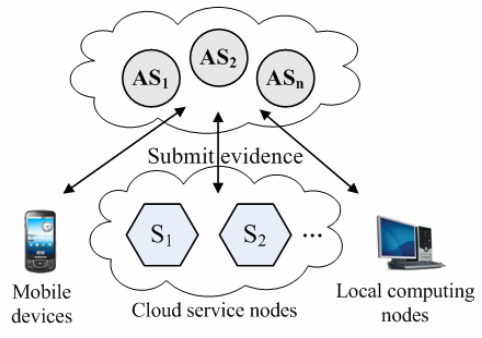


Figure 4. Accountability Service (AS) for compliance and provenance

Here the evidence can be any intermediate gene analysis data generated by the tools in the Cloud, or the decisions made by research personnel participated. With the evidence data logged, the core functionalities provided by the AS nodes are:

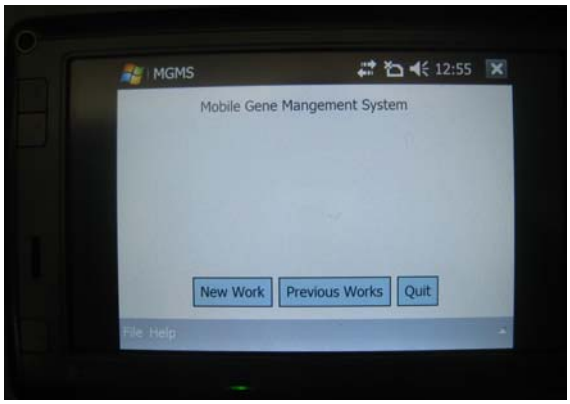
- Compliance verification. Through the analysis of the evidence data, the correctness of the behaviors of the underlying services is continuously validated.
- Data provenance. The evidence recorded capture the evolution path of the data as well as the entities responsible for each step.
- Workflow status monitoring. A global view over the workflow is maintained by the AS nodes. Such information can be used to assist the functioning of the MC-layer and the underlying services.

The AS nodes can either be provided by the Cloud, or by other third parties as long as they receive no benefit whether the underlying services are being compliant or incompliant. They play a neutral role. Note that it is undesirable to build the accountability mechanisms into the MC-layer as it is the entity which designs and orchestrates the workflow and is also subject to errors. Using AS nodes provided by unrelated third parties offers a higher level of honesty and stringency.

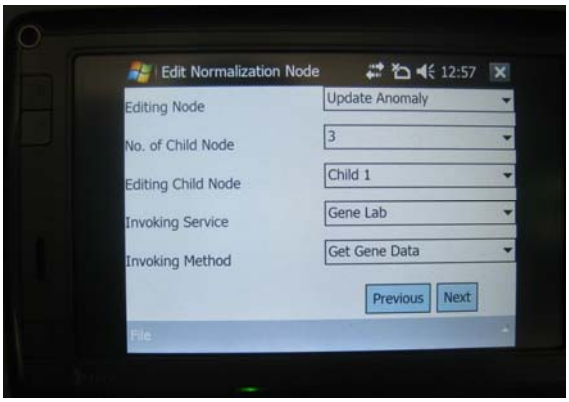
V. EVALUATIONS

We prototyped a demonstration system to showcase our mobile-cloud concept. Our system consists of four parts: i) a client UI deployed in the mobile device; ii) an MC-layer to search and compose services; iii) a number of demonstrating service nodes in Amazon EC2; and iv) an accountability service. In this section, we will first elaborate the implementation of the client UI; then we compare the communication latency when the accountability service is involved and uninvolved; finally we shows some processing latency when a real gene database (KEGG) is involved in a workflow running in the Cloud.

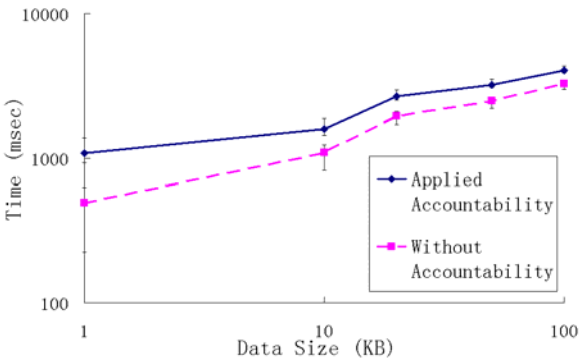
 ISBN: 978-1-61208-153-3



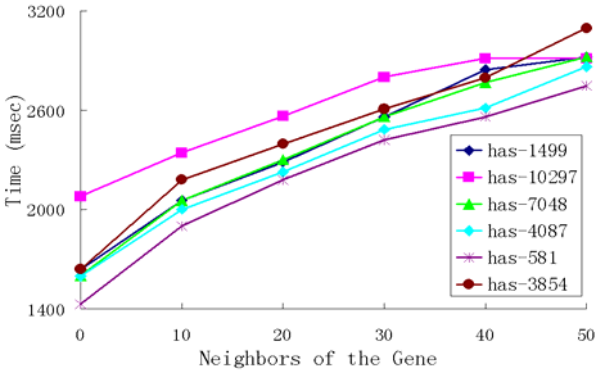
(a) Main User Interface of MGMS



(b) Design a workflow in MGMS



(c) Communication cost with AS and without AS in our system



(d) Performance of invoking KEGG service from the workflow einge

Figure 5. Mobile User Interface and Performance Evaluation

The UI on mobile device is developed using Java platform, micro edition (J2ME). The mobile web service feature is deployed and runs on a HTC 9500 mobile phone, which is running on IBM Websphere Everyplace Micro Environment that supports a connected device configuration (CDC1.1). Figure 5 (a) and (b) show two screen shots of the Mobile Gene Management System (MGMS) - a scientific workflows design and surveillance tools. A user can define or edit a scientific process from the "New Work" button or "Previous Work" button as shown in Figure 5 (a). Then, the user can select into process items and specify their detail information as shown in Figure 5 (b). System users define the steps from four aspects, what services carry out these tasks; the number of child nodes; which methods/services are invoked; and what are the inputs and outputs of each step. Finally, an abstract workflow in BPEL will be generated and uploaded to the WSMS in Cloud, which will instantiate the abstract workflow by filling up the endpoints in the BPEL with the best concrete services URLs.

Figure 5 (c) shows the interaction latency between mobile device, cloud nodes and the AS, with the average value being 492.7msec at 1kB and 3251.2msec after the communication size is increased to 100kB. According to our processes, mobile and cloud nodes need to communicate with the AS so the average value being 660.5msec for the whole system regardless the underlying operation work load. From the curve of this figure, we see the percentage that cost on AS is decreased from 54% to 19% with the communication size increasing from 1kB to 100kB.

To evaluate the performance of gene retrieving from gene bank services, we selected 6 example genes which are the genetic causes of colorectal cancer and retrieve their genetic neighbors from KEGG disease Database [21]. We test the response time from 0 neighbors to 50 neighbors. As shown in Figure 5 (d), it is clear that the latency is slowly increasing with changing the number of neighbors. The has-581 continually kept the best performance at all stages from the 1427msec for retrieving 0 gene neighbor to 2746.8msec for getting 50 neighbors. However, has-10297 spent 2078msec to search 0 neighbors and it cost 2912.6msec for finding 50 neighbors.

VI. RELATED WORK

Mobile computing provides a luggable computation model for users. Its portability makes it very ideal for many application scenarios. To extend its limited computing power, research communities have proposed novel designs to leverage the Cloud. Huerta-Canepa and Lee [22] proposed a virtual cloud system, Zhang et al. [23] detailed a distributed computing platform using mobile phones. They improve the capacities of mobile phones in the purpose of storage and computation. Works like [24-26] presented some compu-

 ISBN: 978-1-61208-153-3

tation offloading methods that move some parts of the applications to run on the Cloud. Executing parts of application remotely can save battery lifetimes and significantly extend computing resources. However, these solutions do not support platform-independent cooperative interaction over an open network. In addition, after moving some parts of applications from stand-alone handheld devices to the cloud, several issues need to be considered in advance such as privacy, trustworthy or provenance.

The importance of provenance for scientific workflows has been widely acknowledged by various research communities. Many approaches have been proposed to record the derivations of the data during the scientific process. Approaches like [14][15] allow the designer to capture the intermediate data forms generated by the experiments at different granularities. In our work, we introduced the concept of accountability which not only provides data provenance but can enforce compliance among the service providers. Compliance assurance has been studied decently in recent years, some remarkable works include [17] [18] [19] [20]. Our work differs from them at the point that we consider a more hostile environment where all service entities are expected to behave in any possible manner and deceive for their own benefit. Cryptographic techniques are deployed in our system to ensure the evidence are undeniable.

VII. CONCLUSION

In this paper, we have proposed a novel design to enable mobile devices to design and participate in the scientific workflows running in the Cloud. The scientific researchers can use mobile devices to sketch an abstract workflow design to be submitted to the mobile cloud middleware layer, which will choose and compose the optimal services according to the designer's requirements. On top of that, we further incorporated accountability mechanisms not only to provide data provenance during the process but also enforce compliance among all the service providers involved. Our testing data indicate that the cost of incorporating accountability is acceptable and becomes negligible when the transmission data become large.

REFERENCES

[1] W. Lu, J. Jackson, and R. Barga. AzureBlast: A Case Study of Developing Science. In proc. Workshop on Scientific Cloud Computing, pp. 413-420, 2010.

[2] O. Moser, F. Rosenberg, and S. Dustdar. Non-Intrusive monitoring and service adaptation for WS-BPEL. In proc. international conference on World Wide Web, pp. 815-824, 2008.

[3] Montage. http://montage.ipac.caltech.edu. Last accessed 8th Sep 2011.

[4] I, Taylor, M, Shields, I. Wang, and R. Philp. Distributed P2P computing within Triana: A galaxy visualization test case. In proc. IEEE International Parallel and Distributed Processings Symposium, 2003.

[5] T. Oinn, P. Li, D.B. Kell, C. Goble, A. Goderis, M. Greenwood, D. Hull, R. Stevens, D. Turi, and J. Zhao. Taverna/myGrid: Aligning a workflow system with the life sciences community. In *Workflows in e-Science*, Springer, 2006.

[6] M. Schena, D. Shalon, R.W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, October 1995.

[7] M. Y. Galperin. The molecular biology database collection: 2008 update. *Nucleic Acids Research*, pages gkm1037+, November 2007.

[8] M. D. Brazas, J. A. Fox, T. Brown, S. McMillan, and B. F. F. Ouellette. Keeping pace with the data: 2008 update on the bioinformatics links directory. *Nucleic acids research*, 36, July 2008.

[9] R. Mulgan. Accountability: An ever-expanding concept? In: Public Administration, pp. 555–573, 2000.

[10] A. R. Yumerefendi, and J. S. Chase. Trust but verify: accountability for network services. In proc. ACM SIGOPS European workshop, article No. 37, 2004.

[11] J. Yao, S. Chen, C. Wang, D. Levy, and J. Zic. Accountability as a service for the cloud, in proc. IEEE International Conference on Services Computing, pp. 81–90, 2010.

[12] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. Deploying and managing web services: issues, solutions, and directions. *VLDB J.*, 17(3):537–572, 2008.

[13] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science, in trans. ACM SIGMOD Record, volume 34, issue 3, pp. 31-36, September 2005.

[14] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation, in *SSDBM*, 2002.

[15] J. Zhao, C. A. Goble, R. Stevens, and S. Bechhofer. Semantically Linking and Browsing Provenance Logs for Escience, in *ICSNW*, 2004.

[16] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, and B. Didier. Multi-Scale Science, Supporting Emerging Practice with Semantically Derived Provenance, in ISWC workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, 2003.

[17] E. Mulo, S. Dustdar, and U. Zdun. Monitoring Web Service Event Trails for Business Compliance. In Proc. International Conference on Service-Oriented Computing and Applications , pp. 1-8, 2009.

[18] M. Huang, L. Peterson, and A. Bavier. PlanetFlow:maintaining accountability for network services. In Proc. ACM SIGOPS Operating Systems Review, pp. 89-94, 2006.

[19] Y. Zhang, K. Lin, and J. Y. J. Hsu. Accountability monitoring and reasoning in service-oriented architectures. In Trans. Service Oriented Computing and Applications, Volume 1, Number 1, pp. 35-50, 2007.

[20] A. C. Squicciarini, W. Lee, B. Thuraisingham, and E. Bertino. End-to-end accountability in grid computing systems for coalition information sharing. In Proc. Workshop on Cyber Security and Information Intelligence Research , 2008.

[21] M. Kanehisa, S. Goto, M. Furumichi, M. Tanable, and M. Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. In trans. Nucleic Acids Research, volume 38, Database issue, pp. 355-360, 2010.

[22] G. Huerta-Canepa and D. Lee. A virtual cloud computing provider for mobile devices. presented at the Proceedings of the 1st ACM Workshop on Mobile Cloud Computing Services: Social Networks and Beyond, San Francisco, California, pp. 61-65, 2010.

[23] J. Zhang, David Levy, Shiping Chen, and John Zic. mBOSSS+: A Mobile Web Services Framework. in Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific, pp. 91-96, 2010.

[24] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso. Calling the cloud: enabling mobile phones as interfaces to cloud applications. the 10th ACM/IFIP/USENIX International Conference on Middleware, pp. 83-102, May,2009

[25] K. Kumar and Y. Lu. Cloud Computing for Mobile Users. Computer, vol. 18, issue 99, pp. 51-56, 2010.

[26] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Cuckoo: a Computation Offloading Framework for Smartphones. In MobiCASE '10: Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, pp. 62-81, 2010.

 ISBN: 978-1-61208-153-3

Efficient Management of Hybrid Clouds

Sofie Van Hoecke, Tom Waterbley, Jan Devos
Electronics and Information Technology Lab (ELIT)
University College West-Flanders (Howest)
Ghent University Association - Courtrai, Belgium
{*sofie.van.hoecke, tom.waterbley, jan.devos*}*@howest.be*

Tijl Deneut, Johan De Gelas
Sizing Servers Lab
University College West-Flanders (Howest)
Ghent University Association - Courtrai, Belgium
{*tijl.deneut, johan.de.gelas*}*@howest.be*

***Abstract*—Cloud computing has become a significant technology trend driven by big players that is transforming our current IT industry. Public cloud computing comes with advantages such as cost savings, high availability, and easy scalability. However, Small and Medium-sized Enterprises are driven by different reasons for not outsourcing their IT infrastructure entirely. By combining the benefits of the private and public cloud, hybrid cloud computing allows Small and Medium-sized Enterprises to optimize their infrastructure and run their virtual machines where they will be most effective and efficient. We present in this paper a virtual infrastructure management tool that allows to set-up and manage hybrid clouds efficiently in a user-friendly way. Our tool provides automatic load balancing between the private and public clouds at the virtual machine level, and dynamically upscales on-premise virtual machines to public cloud servers based on cost and performance information.**

***Keywords*-hybrid cloud, virtual infrastructure management, SME, load balancing.**

I. INTRODUCTION

Cloud computing has become a significant technology trend, driven by big players like Amazon, Microsoft, Google and VMware, and transforming our current IT industry. Cloud computing delivers large-scale utility computing services to a wide range of consumers. Within cloud computing, users on various types of devices access programs, storage, processing and applications over the Internet, offered by cloud computing providers, resulting in a previously unprecedented elasticity of resources. Through economies of scale, cloud computing comes with advantages such as cost savings, high availability, easy scalability [1], and the transformation of capital IT expenditures into operational IT expenditures. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) relieves cloud users from maintaining their infrastructure, development areas, and respectively software.

The increasing adoption rate of cloud computing is currently driving developers, integrators and hosting companies to take cloud computing into account. The last few years, especially the number of providers delivering IaaS has increased quickly [2]. Consequently, companies need to revise their current assets as cloud computing is becoming a strategic weapon. The expansion of IaaS providers increases the options available to companies when acquiring resources in a cost effective manner. Berkeley even predicts that the economy of scale and statistical multiplexing may ultimately lead to a handful of cloud computing providers and "data center-less" companies [3].

However, we do not believe that most Small and Medium-sized Enterprises (SMEs) will become "data center-less" in the near future [4][5]: SMEs cover a wide spectrum of industries [6] and the number of SMEs far exceeds the number of large and very large organizations in almost every country all over the world. Besides the large number of SMEs worldwide, it is also recognized that SMEs constitutes a growing importance as providers of employment opportunities and key players for the well-being of local and regional communities. SMEs are driven by different reasons to maintain their own data center, such as legislation of storing data in-house, investments in the current infrastructure, or the extra latency and performance requirements. This drive is supported by the fact that SMEs have already invested heavily in their own private server equipment and software.

Consequently, we feel that a hybrid approach makes more sense for SMEs. Through the creation of hybrid clouds [7], one can use the internal infrastructure combined with public cloud resources (see Figure 1). This way, on one hand, critical applications can run on the hardware in the private data center or collocated at an SME hosting provider, and, on the other hand, the public cloud can be used as a solution to manage peak demands (cloudbursting) or for disaster recovery. These hybrid clouds capitalize on investments made on the private IT infrastructure, and upscale to the public cloud for specific application requirements.

The architectural concept of a hybrid cloud is overpowering: being able to dynamically move virtualized servers between your data center and a public cloud provider. However, there are still many research challenges to be tackled before hybrid cloud computing can become a reality. One key question is how to enable virtual infrastructure management, meaning the dynamic orchestration of virtual machines (VMs). The scaling efficiency and elasticity of hybrid cloud computing all depend on the efficiency of the virtual infrastructure management [8]. As many cloud providers are incompatible and use proprietary cloud soft-

 ISBN: 978-1-61208-153-3

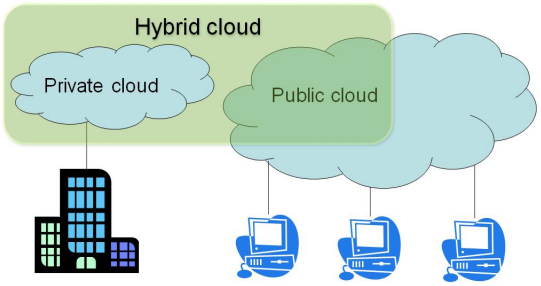


Figure 1. Hybrid cloud computing model

ware and APIs, it is hard to set-up hybrid clouds integrating different cloud solutions, resulting in vendor lock-ins. This is even more the case when it comes to SMEs as they depend strongly on external IT expertise [9][10]. There is little general support with respect to how to set up hybrid clouds and how to manage resources in hybrid environments where management has to act across different resource infrastructures [8]. Existing solutions for hybrid cloud computing and virtual infrastructure management require multiple tools to cooperate and a lot of manual configuration. To the author's knowledge, user-friendly tools that allow SMEs to manage the virtual infrastructure themselves are non-existing.

Therefore, the aim of this research is to design and implement an integrated virtual infrastructure management tool that allows SMEs to set-up and manage hybrid clouds efficiently in a user-friendly way. Our virtual infrastructure management tool provides automatic load balancing between the private and public clouds at the VM level, and dynamically upscales on-premise VMs to the public cloud servers based on cost and performance information. This way, SMEs can optimize their infrastructure and run their VMs where they will be most effective and efficient.

The remainder of this paper is as follows. The next section outlines the benefits and challenges of hybrid cloud computing for SMEs. Subsequently, related work in the field is presented in Section III, after which we define the objectives, design and implementation of our virtual infrastructure management tool in Section IV. Section V covers our experimental evaluation and results, after which we summarize the most important conclusions of our work in Section VI.

II. Benefits and Challenges of Hybrid Cloud Computing for SMEs

The hybrid cloud model extends the private cloud model by using both local and remote resources. These remote cloud resources are seamlessly integrated in the private infrastructure. Hybrid clouds are usually used to scale out when the local resources are exhausted, called cloudbursting. This way, companies can create highly elastic environments.

The benefits of hybrid clouds are amongst others [3][11]:

- **Optimal utilization**
  As peak loads can be up to ten times higher than the average load, traditional data centers need to be over-dimensioned, resulting in idle servers during average load and unnecessary costs. Hybrid clouds scale out to the public cloud to handle peak loads, so the private infrastructure can be dimensioned to handle the average case.
- **Risk transfer**
  The risk of downtime is reduced. Whenever there are problems with the private IT infrastructure, the load can be transferred to the public cloud who ensures high uptimes.
- **Availability**
  Hybrid clouds can upscale to the public cloud or even let the public cloud completely take over operations, this way providing high availability without requiring redundancy and geographic dissemination in the private infrastructure.

However, there are also many challenges and issues for hybrid cloud computing [3][11][12], especially when targetting SMEs:

- **Interoperability and vendor lock-in**
  Vendor lock-ins were already identified as a major risk factor in IT outsourcing [13]. Additionally, failures can also hit public cloud providers, even the big players. According to [3], the only solution is using multiple cloud providers, but, again here, this is only possible when vendor lock-ins are avoided.
- **Hybrid cost**
  Hybrid cloud infrastructures have on one hand a setup and operating cost for the private IT infrastructure (such as hardware, power, cooling, maintenance) and, on the other hand, a pay-per-use cost for the public part at the cloud provider. This hybrid cost model makes it hard to reveal and predict the total costs and benefits of an IT investment project.
- **Security**
  Hybrid cloud computing requires solid service level agreements with and trust in the public cloud providers. As the servers are no longer shielded by the company's firewall, other security measures have to be applied.

III. State of the Art

Today, large technology vendors as well as open-source software projects both address the hybrid cloud market and are developing virtual infrastructure management tools to set-up and manage hybrid clouds. VMware's vCloud offers live migration of virtual appliances and machines between data centers and allows service providers to offer IaaS while maintaining compatibility with internal VMware deployments. HP provides three offerings for hybrid cloud computing: HP Operations Orchestration for provisioning, HP Cloud Assure for cost control, and HP Communications

 ISBN: 978-1-61208-153-3

as a Service for service providers to offer small businesses on-demand solutions. Also Amazon reaches towards hybrid cloud models with its Virtual Private Cloud service.

In addition to the large technology vendors, also open-source software projects are providing on-premise and public cloud integration. Eucalyptus Enterprise provides a software infrastructure for on-premise cloud computing and enables to work within VMware environments and provision resources to Amazon Web Services. Ubuntu Enterprise Cloud (UEC) is shifting from Eucalyptus towards OpenStack and is also compatible with Amazons EC2. OpenNebula [7] is another open-source project that supports the dynamic execution of multi-tier services on a distributed infrastructure consisting of both data center resources and remote cloud resources. Nimbus also provides a virtualization framework to help manage cloud deployments for IaaS. Finally, openQRM extended its focus and now also supports public clouds, currently however limited to Amazon EC2.

Although many options are on the table, and many advertise they support hybrid cloud computing, the current initiatives of the large vendors either result in (i) a hypervisor and/or vendor lock-in, (ii) require a separate interface to manage the private and public cloud, or (iii) require additional tools to implement the load balancing. vCloud only supports ESX as hypervisor, UEC Xen and KVM, and openQRM ESX, Xen and KVM as hypervisor. The virtual infrastructure management capabilities of the open-source solutions provide more choice but require a lot of scripting and are difficult to configure and use. If a flexible hybrid cloud is the goal, the choice of the underlying virtualization platform is crucial, putting the open-source solutions afront. As Nimbus only supports a limited number of hypervisors, and Eucalyptus is more appropriate for private clouds [7][11], OpenNebula and openQRM are the best options today [14].

However, although OpenNebula and openQRM are the best options today, important features are missing like monitoring VM instances or retrieving the VM's IP addresses in order to implement advanced load balancing mechanisms. Also, a graphical interface is either missing or not user-friendly. As general speaking, SMEs are lacking behind in adoption of IT compared to larger companies [15], a good, user-friendly, vendor-independent virtual infrastructure management tool is needed to help SMEs efficiently set-up and manage hybrid clouds. To our knowledge, we are the first to provide such a tool that allows automatic load balancing between the private and public clouds at the VM level, and dynamically upscales on-premise VMs to the public cloud servers based on cost and performance information.

IV. Design of the virtual infrastructure management tool

Below the objectives, general concept and implementation of the virtual infrastructure management tool are described.

A. Objectives

Three requirements for hybrid clouds will be fulfilled: firstly, the hybrid cloud should be able to autonomously handle different load patterns, including peak demands, balance the load and upscale when needed to the public cloud. Secondly, the hybrid cloud should be transparent for the VMs in the infrastructure. This makes sure that the hybrid cloud logic needs only to be implemented in our tool, and standard virtualization software can be used on the VMs. And thirdly, the hybrid cloud architecture should be easy and user-friendly to set-up and configure. Human-platform interaction for configuring the hybrid cloud should be straightforward by using user-friendly user interfaces so that training and dependency on external IT expertise can be minimized. Current hybrid cloud initiatives do not fulfill these requirements and therefore a new tool is designed and presented in this paper.

B. General concept

Consider the use case of the SME Nieuws.be to illustrate the general concept and show how current SME data centers can be optimized in order to gain competitive assets to the public cloud. Nieuws.be is an internet company that aggregates and distributes national and international news on the web, collected by the redaction or contributed by one of their readers. By providing filtering and news-on-demand, they fulfill new market requirements and gain market share within the widely spread news business. As a result, they need an infrastructure to support their heavily visited news site.

In the traditional way, Nieuws.be buys and maintains an infrastructure of six load balanced web servers and one database. Using virtualization and by setting up a hybrid cloud, they can optimize their infrastructure by consolidating lightly used servers. Using virtualization, three web servers can be ran on a single machine. The database is heavily used, so is virtualized on a dedicated machine. As a result, the same performance can be achieved by only three (heavily used) physical servers. In case of peak demands, the heavily used servers cannot handle the load, and therefore cloudbursting is used to automatically allocate additional resources in the public cloud. Additionally, as Nieuws.be is a local news site, the request pattern also depends strongly on the time of the day. As can be seen in Figure 2, during night hours, requests to Nieuws.be fall back to a minimum. Therefore, during down times, the hybrid cloud can further optimize by migrating lightly used VMs to a single host, allowing to shutdown one of the servers.

One can immediatly see that this hybrid cloud architecture results in a two-fold cost reduction: on one hand, only three instead of six servers need to be bought; on the other hand, the infrastructure has a huge energy saving. Besides the cost benefit of consuming less energy, it additionally has the social benefit of reducing their energy footprint.

 ISBN: 978-1-61208-153-3

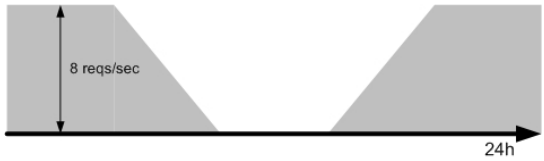


Figure 2. Request pattern of Nieuws.be

Other use cases that illustrate the benefit of hybrid cloud computing to handle peak loads are the Hallmark infrastructure around Valentine's day, or the Colruyt infrastructure in December due to holiday purchases and gifts (see Figure 3).

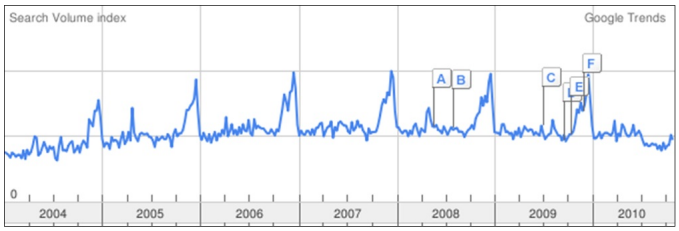


Figure 3. Request pattern of Colruyt

C. Implementation

The main functionality of the virtual infrastructure management tool provides efficient management of hybrid clouds through automatic scaling and load balancing. Figure 4 illustrates the architecture for the virtual infrastructure management tool.

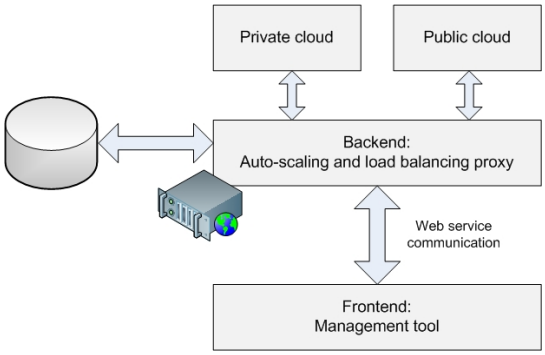


Figure 4. Architecture of the virtual infrastructure management tool

As can be seen in this figure, two parts make up the virtual infrastructure management tool: on one hand, a proxy that implements different load balancing algorithms and provides configurable thresholds for upscaling to the public cloud; on the other hand, a management interface that visualizes and manages the hybrid cloud, clusters VMs and remotely configures the proxy.

The major task of the proxy is forwarding the incoming requeststs to the appropriate VMs. The proxy supports different load balancing algorithms in order to do so. At the moment round robin and weighted round robin are supported, but more algorithms will follow in the future. Round robin is especially suited for load balancing when the different VMs have (almost) the same performance. If the VMs have different specifications, weighted round robin can be used to compensate for these differences. There, servers are presented requests in proportion to their weighting resulting in fairly distributing the requests amongst VMs, instead of equally distributing the requests. To support the weighted load balancing, the performance of all VMs needs to be monitored and the thresholds for up- and downscaling need to be configured. As each public cloud instance type differentiates itself from the others in terms of price, number of virtual cores, available memory and I/O bandwidth, these pricing and performance models are used to derive the weights for the weighted round robin load balancing and can also be used to implement more advanced load balancing and upscaling algorithms in the proxy.

The management interface, presented in Figure 5, provides a tab for visualizing the current VMs in the infrastructure, both private and public ones, as well as the functionality to start and stop these VMs. New VMs can also be added, and clusters can be generated. A cluster is a group of VMs providing the same service. Each cluster can use its own load balancing and scaling settings. The management interface also provides a tab to configure the proxy's load balancing algorithms and scaling thresholds.

Both the proxy and management tool are implemented using C# in combination with the .NET Framework. In order to fulfill the defined requirements above, it is important that the VMs in the private cloud can be addressed the same way as the VMs in the public cloud in order to simplify the hybrid cloud management. Therefore, the proxy is implemented on a dedicated VM in the cloud without an interface, and uses a plug-in system to communicate with different private and public cloud providers. In order to support the automatic upscaling, the VMs also need to be monitored. In order to do so, the APIs of the different cloud providers are used. All requests sent to the proxy are load balanced and forwarded to the according VM (private or public); the response however is directly sent to the client, skipping the proxy. The communication between the management tool and the proxy is implemented using Web service technology and WCF. In order to adjust the proxy settings, the user can use the graphical management interface which sends Web service requests to the proxy in order to configure the thresholds and load balancing algorithms. At the backend, a database is used to store the properties of the hosts, clusters, VMs, scaling thresholds and load balancing constraints. This database is updated by periodically monitoring the VMs in the background. The database is implemented using SQL Server and the Entity Framework. This way, the relational structure of the database is abstracted and one can directly

 ISBN: 978-1-61208-153-3

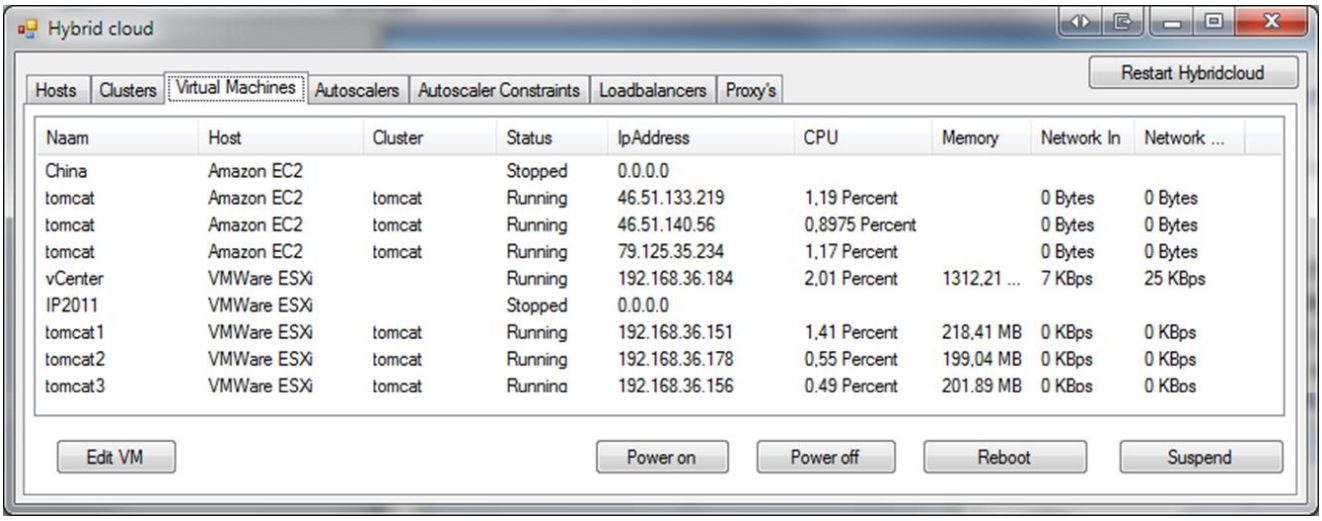


Figure 5. Screenshot of the management tool presenting the infrastructure overview

work with the created object classes.

Currently, VMware is supported for the private part of the hybrid cloud, and Amazon for the public part of the hybrid cloud. More cloud providers will be added in the future. For the public cloud part, the Amazon EC2 service is used to manage the public VMs, and the Amazon CloudWatch service is used to monitor the status of the VMs. The Amazon AWS API can be used starting from Microsoft .NET Framework v2.0. The VMware vSphere API is similar to the Amazon AWS API for .NET, but then applied to a VMware cloud. In order to set up the hybrid cloud, the VMware vSphere PowerCLI API is used. This PowerCLI offers in the first place an interface to Windows PowerShell (which is a new and advanced command-line shell for Windows), but the dlls can also be imported in .NET projects resulting in the same functionality being available using programming code.

V. Evaluation and Results

The components of the virtual infrastructure management tool have been implemented and are currently being evaluated.

Figure 5 presents a screenshot of the management interface where the current infrastructure is visualized, presenting all private and public VMs and their properties. Each of these VMs can be started, stopped, rebooted or suspended. The management tool also provides tabs to add additional VMs to the infrastructure or to create service clusters. The proxy configuration can also be done in the graphical tool: thresholds for up- and downscaling can be configured (see Figure 6) and a load balancing algorithm can be selected and tuned.

The operation of our virtual infrastructure management tool was verified through an experimental performance study. During the evaluation, we started with two web

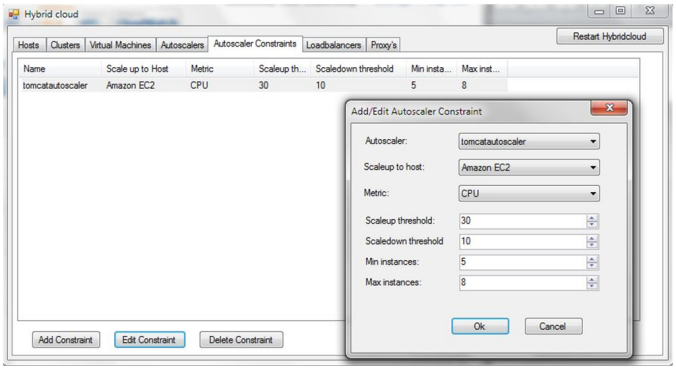


Figure 6. Screenshot of the management tool presenting the scaling tresholds configuration

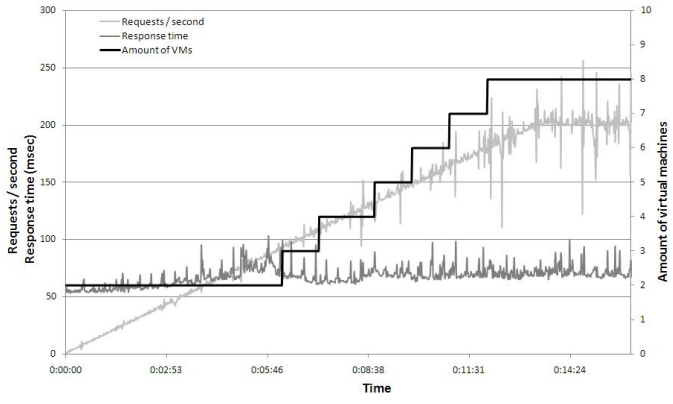


Figure 7. Average response time and amount of VMs in function of the requests per second

servers in the private cloud and upscaled to maximum six web servers in the public cloud, resulting in an infrastructure consisting of eight VMs. The number of requests per second

was increased over time. The test results are presented in Figure 7. As can be seen in this figure, the response time initially increases as the requests per second increases. When the upscaling thresholds are reached, VMs are added, stopping the response time from further increasing as the load is now balanced over an expanded infrastructure.

VI. CONCLUSIONS

This paper highlights the opportunities of hybrid cloud computing for SMEs and presents a virtual infrastructure management tool that can be used by SMEs to set-up and manage their hybrid cloud. Different reasons can drive SMEs to maintain their own data center instead of becoming "data center-less".

Providing a tool to easily set-up and manage these hybrid clouds takes into account the technical possibilities, the SMEs perspective, and the economic tradeoff between the different business models such as classic data centers, private cloud computing and public cloud computing. We are aware of the fact that these hybrid clouds are not the best solution for every SME. If the restrictions of the public cloud not apply and the SMEs only have a limited IT infrastructure and expertise, then outsourcing to public clouds can be very interesting due to economy of scale.

Results clearly illustrate that current SME data centers can be optimized to compete with the public cloud. As the number of SMEs far exceeds the number of large and very large organizations in almost every country all over the world, this approach results in interesting business benefits. By using hybrid clouds, SMEs critical or latency sensitive applications are kept on the infrastructure (collocated or not) in which they have already invested, and applications are moved toward cloud computing enabled data centers in order to handle occasional peak requests. This methodology allows SMEs to freeze capital investments and move applications toward cloud computing enabled data centers. Hybrid cloud computing may therefore become a very important competitive feature of SME data centers to leverage the economies of scale that the "public clouds" offer.

We will continue the design of more advanced load balancing algorithms by taking into account the different pricing and performance models. Future work also includes the development of additional plug-ins for our proxy so on one hand more private and public cloud providers are supported, and on the other hand also security can be managed. Especially the latter is challenging as classical security models may be insufficient as data is replicated and distributed in potentially worldwide infrastructures [8].

REFERENCES

[1] N. Leavitt, Is Cloud Computing Really Ready for Prime Time?, Computer, 42(1), pp. 15-20, 2009.

[2] D. Hilley, Cloud computing: A taxonomy of platform and infrastructure-level offerings, Georgia Institute of Technology, Technical Report GIT-CERCS-09-13, April 2009.

[3] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, et al., Above the Clouds: A Berkeley View of Cloud Computing, Technical Report, February 10, 2009.

[4] J.A. Welsh and J.F. White, A Small Business Is Not a Little Big Business, Harvard Business Review, 59(4), pp. 18-32, 1981.

[5] N.A. Sultan, Reaching for the cloud: How SMEs can manage, International Journal of Information Management, 31, pp. 272-278, 2011.

[6] J. Bolton, Small Firms: Report of the Committee of Inquiry on Small Firms, Cmnd 4811, HMSO, London, 1971.

[7] B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster, Virtual Infrastructure Management in Private and Hybrid Clouds, IEEE Internet Computing, 13(5), pp. 14-22, 2009.

[8] K. Jeffery and B. Neidecker-Lutz, The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010, Expert Group Report, European Commission, 2010.

[9] J. Thong, C.S. Yap, and K.S. Raman, Top management support, external expertise and information systems implementation in small businesses, Information Systems Research, 7(2), pp. 248-267, 1996.

[10] C.S. Yap, C. Soh, and K.S. Raman, Information systems succes factors in small business, Omega - The International Journal of Management Science, 5(6), pp. 597-609, 1992.

[11] P.C. Heckel, Hybrid Clouds: Comparing Cloud Toolkits, Seminar Paper Business Informatics, University of Mannheim, 2010.

[12] S. Fraser, R. Biddle, S. Jordan, K. Keahey, B. Marcus, et al., Cloud computing beyond objects: seeding the cloud, Proceeding of the 24th ACM SIGPLAN Conference on Object oriented programming systems languages and applications, pp. 847-850, 2009.

[13] B.A. Aubert, M. Patry, and S. Rivard, A Framework for Information Technology Outsourcing Risk Management, Database for Advances In Information Systems, 36(4), pp. 9-28, 2005.

[14] P. Sempolinski and D. Thain, A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus, Proceedings of the IEEE International Conference on Cloud Computing Technology and Science, pp. 417-426, 2010.

[15] J.G. Devos, H. Van Landeghem, and D. Deschoolmeester, IT/IS and Small and Medium-Sized Enterprises: Literature Overview, paper submitted for publication in the Journal of Small Business Management.

 ISBN: 978-1-61208-153-3

Cloud Computing and its Application to Blended Learning in Engineering

Sanda Porumb, Bogdan Orza, Aurel Vlaicu
Communication Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{sporumb, orza, aurel.vlaicu}@com.utcluj.ro

Cosmin Porumb, Ioan Hoza
R&D Department
HyperMedia Ltd
Cluj-Napoca, Romania
{cosmin, ionut}@hpm.ro

***Abstract*—The education process in engineering means theory and practice, individual study, group-based projects or experimental work that involves equipment, simulation/emulation software packages and laboratory applications. In order to develop e-learning platforms for higher and postgraduate education in engineering, new methodologies should be taken into consideration: project- and problem based learning, virtual laboratory (remote access to laboratory infrastructure and task evaluation) or remote assistance for diploma projects and mobility grants. This paper presents new blended learning methodologies and the manner they can be customized for higher and postgraduate education in engineering by using the cloud computing paradigms.**

Keywords - cloud computing; blended learning; virtual laboratory; hybrid classware; project-based learning; problem-based learning

I. INTRODUCTION

In recent years, with the advances of the Internet and e-learning technologies, a blended mode of learning, which effectively combines the traditional face-to-face learning and e-learning, has evolved. Yet, this blended learning mode is not widely adopted in higher and postgraduate education in engineering. One major reason is that teachers are not familiar with the practice of designing courses under the blended learning environment. Another important aspect is that many teachers do not consider the e-learning methodologies as stable and functional enough for engineering, especially for laboratory and project task completion. The third reason is that academic staff considers the act of teaching/learning engineering as more than individual study and online assessment.

Engineering consists of lecture attendance, project development, hands-on laboratory-based activities and computer simulation work. In this way, the educational act can be considered as learner-centered. Manseur [5] presented the synchronous distance learning concept (SDL) and its application to Electric and Computer Engineering and Mathematics. Students follow lectures live via videoconferencing but they attend laboratory sessions taught by on-site faculty. The advanced technology has been used for linking the local and the remote classrooms: the lecturer teaching in one location is videotaped and can be seen live on a TV screen in the other classroom. The hands-on experimentation is difficult to conduct without access to often expensive equipment and components and without competent on-site laboratory tutors. In order to complete the lab, the SDL environment consists of two sets of fully equipped and staffed laboratories, one on each end of the SDL-connected campuses.

Qiu [7] proposed a blended learning environment that implements the face-to-face teaching and e-learning capabilities in Advanced Software Engineering. A set of integrated projects was selected as stimulus to learning. Both inter- and intra-group collaborative learning are encouraged. A survey conducted at the end of the course revealed that students accept the problem-based learning quite well, and their academic achievements were also better than expected. The methodology consists of grouping student in teams, dividing the semester in project phases and developing the project using iterations.

This paper is organized as follows: the related works and proposals are presented in Section II. Section III is dedicated to the blended learning models for higher and postgraduate education. Several important aspects are taken into consideration: how to improve the retention factor in the individual study, how to provide remote access to laboratory infrastructure and applications and how to support fundamental and applied research activities within individual, group-based projects or international partnerships. Section IV starts with the technological aspects and continues with the deployment diagram of the blended learning platform for technical education and continues with the elastic cloud environment presentation. The experimental results, especially the platform deployment for "Economic and Exact Sciences" and "Applied Electronics, Telecommunications and Information Technology" domains, including the blended learning support, practice and assessment processes, are highlighted in Section V of this paper. In conclusion, the authors underline the importance of SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) concepts in higher and postgraduate education by presenting a complex scenario for extending legacy e-learning systems in order to support blended learning capabilities.

II. RELATED WORKS

Mendez [3] illustrates that in traditional web-based learning mode, system construction and maintenance are located inside the educational institutions or enterprises, which led to a lot of problems, such as significant investment needed but without capital gains for them, which

 ISBN: 978-1-61208-153-3

leads to a lack of development potential. In contrast, cloud-based e-learning model introduces scale efficiency mechanism, i.e. construction of e-learning system is entrusted to cloud computing suppliers, which can make providers and users to achieve a win-win situation. The cloud-based environment supports the creation of new generation of e-learning systems, able to run on a wide range of hardware devices, while storing data inside the cloud.

Laisheng [9] highlighted a new business paradigm in educational area by introducing the cloud computing in order to increase the scalability, flexibility and availability of e-learning systems. The authors have evaluated the traditional e-learning networking model, with its advances and issues, and the possibility to move the e-learning system out of schools or enterprises, inside a cloud computing infrastructure. The separation of entity roles and cost effectiveness can be considered important advantages:

- The schools and enterprises will be responsible for the education process, as well as for content management and delivery, and the vendor takes care of system construction, maintenance, development and management;
- The e-learning system can be scaled, both horizontally and vertically, and the educational organization is charged according to the number of used servers that depend on the number of students.

Ouf [10] has presented an innovative e-learning ecosystem based on cloud computing and Web 2.0 technologies. The article analyzes the most important cloud-based services provided by public cloud computing environments such as Google App Engine, Amazon Elastic Compute Cloud (EC2) or Windows Azure, and highlights the advantages of deploying E-Learning 2.0 applications for such an infrastructure. The authors also identified the benefits of cloud-based E-Learning 2.0 applications (scalability, feasibility, or availability) and underlined the enhancements regarding the cost and risk management.

Chandral [11] focused on current e-learning architecture model and on issues in current e-learning applications. The article presents the Hybrid Instructional Model as the blend of the traditional classroom and online education and its customization for e-learning applications running on the cloud computing infrastructure. The authors underline the e-learning issues, especially the openness, scalability, and development/customization costs. The existing e-learning systems are not dynamically scalable and hard to extend – integration with other e-learning systems is very expensive. The article proposed the hybrid cloud delivery model that can help in fixing the mentioned problems.

The e-learning platforms for higher and postgraduate education in engineering should provide remote access to both educational materials and laboratory infrastructure. They also need to implement synchronous/asynchronous collaborative learning features, as well as blended assessment functionality. Such a platform is expensive and its development can take much time. The cheapest solution is to opt for public cloud computing services, even if the component integration and customization will need important investments.

The learning cloud prototype presented here is a fully functional, application-oriented, and in the same time, low-cost solution that provides SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) capabilities. Software as a Service is used to deliver the educational applications to the browser of the user/ customer from the learning cloud. It helps the faculties and departments with limited IT resources to deploy and maintain needed software in a timely manner while, at the same time, reducing energy consumption and expenses. Platform as a Service facilitates the development and deployment of applications, such as laboratory simulation software packages, without the cost and complexity of buying and managing the underlying infrastructure (hardware and associated software). Infrastructure as a Service gets on-demand computer infrastructure (virtual desktop or data center, e.g.).

III. BLENDED LEARNING MODELS

From the teaching point of view, six essentials are identified: teaching subjects, teaching content, teaching environment, teaching models, teaching organizers and teaching administration. In order to improve their knowledge and skills, the students (subjects) actively participate to both real and virtual educational acts. So, the learning service providers should pay attention to both teaching modes: face-to-face and Internet-based. The advances point out the manner of getting them together, in order to expand the real educational environment and make the virtual platforms an important part of the educational system.

A. Blended Learning Model for Higher Education

The traditional e-learning platforms consist of the learning management system, learning content management system, assessment and communication modules (especially forum and messaging). The third generation of e-learning platforms provides with advanced services such as online courses, tutorials and webinars. The education process in engineering means theory and practice, individual study, team projects or experimental work and involves laboratory equipment, simulation/emulation software packages and applications.

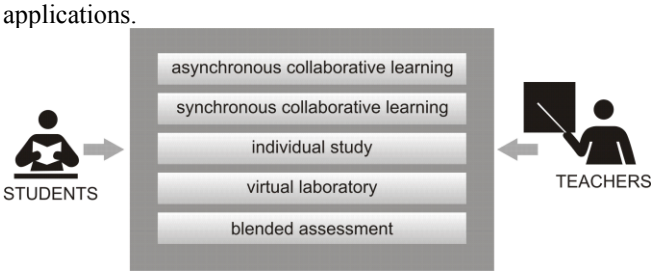


Figure 1. Blended Lerning Model for Higher Education in Engineering.

The e-learning platforms for higher education in engineering implement new methodologies such as: project- and problem based learning, virtual laboratory (remote access to laboratory equipment and applications and task

evaluation) or remote assistance for diploma projects and mobility grants. The blended learning model illustrated in Figure 1 proposes the following educational phases: asynchronous/synchronous collaborative learning, individual study support, virtual laboratory and blended assessment. Two main blocks should be taken into consideration in the e-learning system architecture: hybrid classware and asynchronous collaborative learning modules. The hybrid classware is a complex blended learning approach that provides with classroom-based education, synchronous collaborative learning (online course/tutoring/mentoring), virtual laboratory (remote access to laboratory equipment and applications) and blended assessment (practice and theory) capabilities. The problem-/project-based learning and individual study (interactive courses/tutorials) features are implemented in the asynchronous collaborative learning section.

B. Blended Learning Model for MSc Programmes

In MSc programmes the students are focused on research and career development activities. The educational schema consists of live lectures, hands-on experimentation, individual and group-based projects, virtual team cooperation and mobility grants. It is defined around the following skills: information synthesis in theory and hands-on experimentation or online simulation, requirement analysis, project design, implementation, or result presentation.

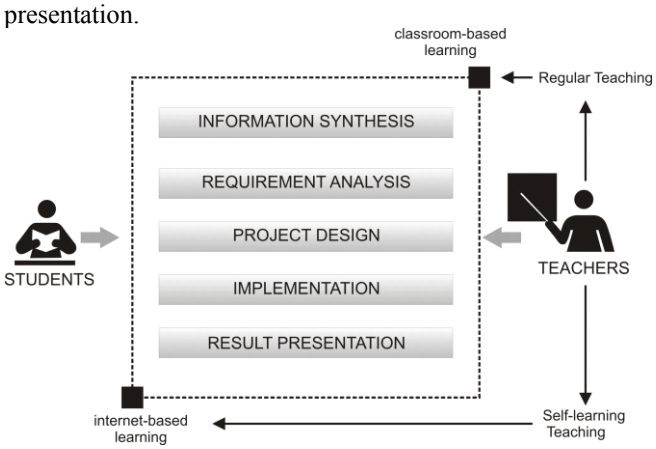


Figure 2. Blended Lerning Model for MSc Programmes in Engineering.

The blended learning model illustrated in Figure 2 proposes the two main phases in the educational act: regular teaching and Internet-based learning. Information technology is important in education, even when regular teaching involves advanced technologies such as multimedia presentations, video projectors, or smartboards. Self-learning means individual study starting from educational materials created and posted onto the e-learning platform by the teachers, then browsing the Internet to find and select correct information about subjects related to the educational materials.

Online tutoring approach consists of interactive tutorials and face-to-face Internet-based learning. Interactive tutorials can be also used as the introduction part of hands-on experimentation activities. The face-to-face Internet-based learning includes the online classroom/webinar sessions and remote assistance during the international research projects or mobility grants.

C. Blended Learning Model for PhD Programmes

Blended learning is not new - what is new is the recognition of its potential to help in fundamentally redesigning the learning experience in ways that could enhance the traditional values of higher education and postgraduate scholarship (MSc and PhD programmes). Preparing PhD students according to a blended strategy can be challenging, since it requires gaining different teaching skills and technologies. Redesigning the educational process takes into account new teaching and learning opportunities, managing the educational content both online and in-class, and preparing PhD students to work in a hybrid format.

In Romania, the PhD scholarship based on European Social Funds (ESF) constrains the students to complete their PhD in three years, so, the activities should be well defined and supported by clear results. This aspect completely changed the PhD methodology (illustrated in Figure 3). For the moment, the PhD students must be integrated within research projects and work close to real and efficient research teams. Most of the research projects are developed according to the Scrum methodology [2]. This way, the authors took into consideration the blended learning and Scrum methodology for improving the education and research activities in PhD scholarship based on ESF Funds.

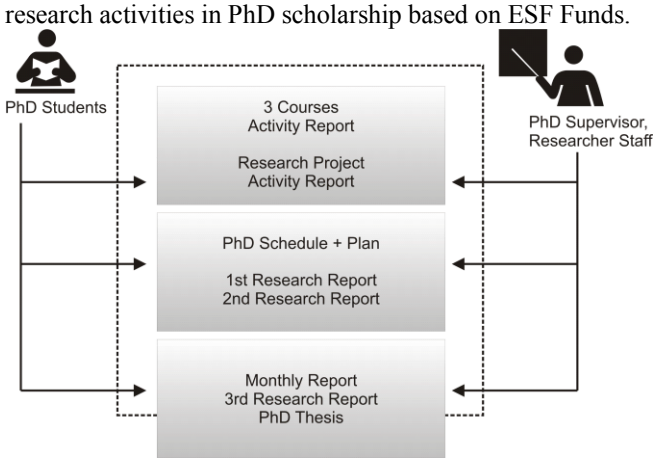


Figure 3. Blended Lerning Model for PhD Programmes in Engineering.

The PhD programme means theory, practice and research activities with results published in scientific journals and conference proceedings. This way, the problem- and project-based learning should be considered as necessary. The Technical University of Cluj-Napoca provides with PhD programmes in the following domains of interests: Automation, Computer Science, Electric Engineering, Applied Electronics and Telecommunications, Civil Engineering, Mechanics, etc. With the big number of domains and the increasing number of MSc and PhD students, the blended learning environment that supports MSc and PhD activities should be both horizontally and

 ISBN: 978-1-61208-153-3

vertically scalable. In conclusion, an elastic learning cloud infrastructure should be implemented.

IV. E-LEARNING CLOUD INFRASTUCTURE

From the beginning, the role of blended learning was to improve the educational process by increasing the degree of students' satisfaction, retention factor and students' enrollment and developing students' skills. In higher education, especially in engineering, the blended learning is a need because of the diversity of teaching/learning activities. The quality of the learning act can be considered another important aspect, so increasing number of students enrolled should not affect the educational process. The learning cloud means reliability and scalability, as well as cost effectiveness.

Laisheng [9] proposed a generic e-learning cloud and identified several challenges such as: charge, bandwidth, security, user's awareness and acceptance, educational forms and methods and resource development, and proposed solutions for each challenge. By setting up a market-oriented charging mechanism and by combining two types of fees, school fees and individual fees (with school charging for general resources and individual charging for special resources) can be considered a solution. The bandwidth problem is almost fixed in Romania because between RoEduNet and each important Internet service provider there are peering services. In order to keep the integrity and confidentiality of data an encryption mechanism should be implemented for both storage and transmission.

The e-learning cannot completely replace teachers; it is only an updating for technology, concepts and tools, giving new content, concepts and methods for education, so the roles of teachers cannot be replaced. The teachers will still play leading roles and participate in developing and making use of e-learning cloud. The blended learning strategy should improve the educational act. Moreover, the interactive content and virtual collaboration guarantee a high retention factor (up to 80%) [4].

A. E-Learning Cloud Architecture

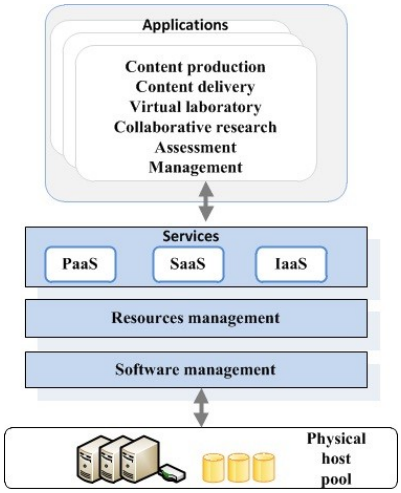


Figure 4. Learning Cloud Architecture.

The proposed learning cloud architecture illustrated in Figure 4 can be divided into the following layers: hardware resource layer as a dynamic and scalable physical host pool, software resource layer that offers a unified interface for e-learning developers, resource management layer that achieves loose coupling of software and hardware resources, service layer, containing three levels of services (software as a service, platform as a service and infrastructure as a service), application layer that provides with content production, content delivery, virtual laboratory, collaborative learning, assessment and management features.

B. E-Learning Cloud Setup

In the classic blended learning model, teachers assign teaching tasks, conduct regular lectures, or train students' skills. The students attend the online autonomous learning act and cooperative learning sessions, or accomplish teachers' assignments. The teachers make assessments over students' learning effect and solve their problems. So, teachers set objectives and tasks of different levels, they put forward requirements and suggestions according to the teaching contents and make assessments to students' learning effects through task-based activities. Teachers also answer students' questions and offer essential teaching to major and difficult points. In addition, teachers can also use multimedia to enhance teaching content. Of course, teachers create flexible and diversified theoretical and practical scenarios and teaching contents, using authentic materials to let students come upon more technical information related to real problems/projects. Students work out their own learning plans, determining learning methods autonomously. They conduct on-line autonomous learning when they study each unit, finish its test via Internet and do some statistics to the test results. Teachers also encourage students to cooperate with each other to finish simple learning tasks or complex group-based projects. Through cooperative learning, students cannot only acquire knowledge, their team spirit and coordination will also be fostered, skills in dealing with people will be improved and abilities to express themselves will be enhanced. In applied electronics, telecommunications and information technology, the learning environment also provides with hands-on experimentation work, simulation software packages and semester/diploma projects.

We proposed a learning cloud environment built around Citrix XenServer. XenServer is an enterprise-ready, cloud-proven virtualization platform that contains all the capabilities required to create and manage a virtual infrastructure and provides an efficient management of Windows and Linux virtual servers and delivers cost-effective server consolidation [1]. The initial setup, illustrated in Figure 5, must support the teaching/learning activities and practice. It should be a dynamic environment, able to create university/programme instances. Each instance consists of six virtual machines: two allocated for web hosting, two for the data warehouse and two for the virtual library. The initial setup also includes the collaborative work environment that hosts the

 ISBN: 978-1-61208-153-3

asynchronous/synchronous collaborative learning tools: course authoring tool, interactive tutorial, messaging, forum, web conferencing, online focus group, or virtual classroom.

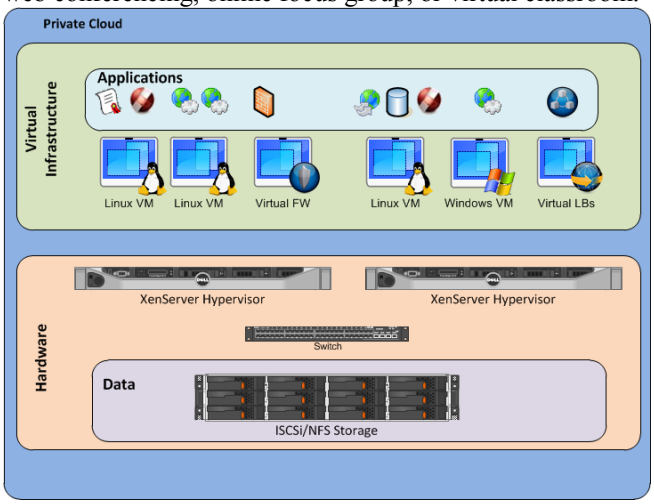


Figure 5. E-learning cloud setup.

The learning management system allows the students to schedule online laboratory activities. The e-learning cloud infrastructure implements an advanced resource pooling mechanism (see Figure 6) that dynamically allocates twenty virtual machines for each university instance when the first student scheduled a virtual laboratory session. When fifteen of the initial virtual machines are allocated, the resource pooling mechanism allocates other twenty. The virtual machine will not be a powerful one. Its role is to provide students with remote access to lab equipment and simulation software packages needed for completing the tasks. The activity starts with an interactive tutorial, where the tutor describes the tasks and gives some suggestive examples related to the current work. The students remotely access the lab equipment and/or applications and complete the tasks.

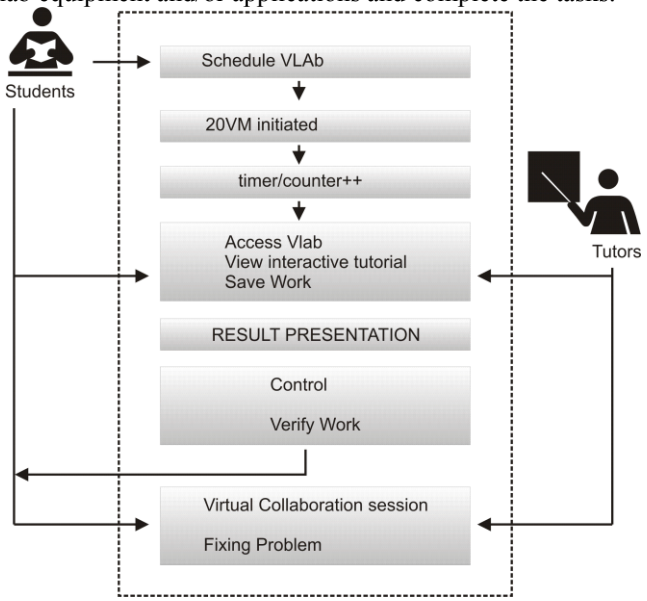


Figure 6. Virtual Lab Approach.

At the end of the lab session, each student saves its own work, in order for the tutor to verify it. If the tasks are not properly done, the tutor notifies the student to repeat the work or to attend a collaborative session in order to fix the problems together.

The online access to the laboratory infrastructure complies with a well-defined schedule. It is almost impossible to allocate one virtual machine for each student enrolled in the educational program. This way, the students will access the virtual laboratory in groups of ten students. At the same time, we can have groups of ten virtual machines to be allocated for each field/line of study.

V. EXPERIMENTAL RESULTS

Each educational organization should have its own staff that manages both educational act and content. When registering, the account manager should specify the number of students, form of study, education domain, then an intelligent block processes the information and provides with the most appropriate configuration needed for such a programme.

The "Aurel Vlaicu" University of Arad opted for an e-learning cloud-based service, in order to support blended learning in the Faculties of Economic and Exact Sciences. The manager of the Distance Learning Department completed the registration forms and defined a clear structure of BSc programme for 2 faculties, 5 domains, 3 years of study, 72 teachers and more than 3000 students. The educational process in the Faculty of Economic Sciences consists of flexible individual study, individual and group-based projects, online and face-to-face teaching, online and classroom-based assessment, webinars and web meetings between students and/or students and tutors. In the Faculty of Exact Sciences, it also includes virtual laboratory activities, especially remote access to lab applications (software development environment), and semester/diploma project support.

The configuration block automatically creates the virtual machines (VM) and allocates the hardware and software resources: two VM allocated for web hosting, two for the warehouse and two for the virtual library. In the Faculty of Exact Sciences, the virtual laboratory involves one virtual machine allocated for each student, the virtual desktop that allows the student to complete his own work, and a reduced storage space necessary for saving the work at the end of the laboratory session. The virtual machine has minimal hardware and software requirements: it should support the software packages needed for completing the current tasks.

The e-learning cloud prototype is also implemented in the Technical University of Cluj-Napoca, Faculty of Electronics, Telecommunications and Information Technology, for MSc and PhD programmes. One of the pilot courses, "E-Business Technologies", involves 25 students, some of them involved in Erasmus mobility grants. By using the hybrid classware component, the Erasmus students have been able to actively participate to courses and lab activities. The teachers were also assisting the students during semester or diploma projects.

 ISBN: 978-1-61208-153-3

The virtual educational environment will provide with classroom-based lectures, online courses, interactive tutorials, virtual laboratories (especially access to simulation software packages), problem- and project-based learning, and remote assistance for semester and diploma projects. The e-learning cloud automatically deploys an instance for each educational progamme that consists of six virtual machines. The management component processes the learning schedule related to each programme, controls and re-allocates the hardware and/or software resources, invokes the interactive/collaborative tools and provides online access to educational resources and laboratory infrastructure.

The hybrid classware, illustrated in Figure 7, supports both synchronous collaborative leaning and face-to-face teaching. It enables the teachers to present the educational material in the classroom and simultaneously project it in the virtual space. The students can opt for assisting the presentation in the classroom or using the virtual classroom component.

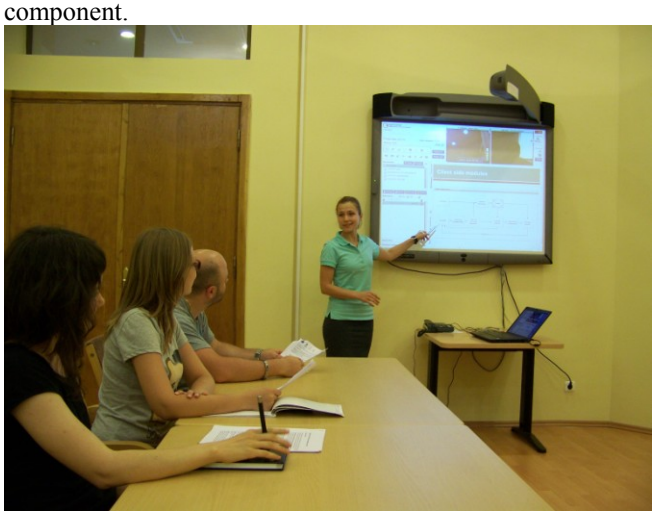


Figure 7. Hybrid classware implementation.

A complementary tool that allows the lecturer to dynamically handle the educational content is integrated into the learning cloud. Two types of educational content are stored into the virtual library: public and private content. If the lecturer considers one of his/her materials as really important for the public interest, that material will be uploaded on the server, converted to an internal format (SCORM compliant) and stored into the virtual library as a public material. If the material is private, or if the lecturer has no rights to make it public, it will be converted to the slideshow format and then stored into the library as private. The tutor is able to browse the media library, load it on the shared space and share it among the virtual classroom session. Asynchronous collaborative learning is also allowed. The lecturer is able to create interactive learning content by using the course authoring tool and store it into the virtual library. The student accesses the virtual library, browses the content and manages his own schedule.

When setting up the hardware and software resources for the MSc Programme “Multimedia Technologies”, the staff should evaluate the laboratory equipment and applications to be integrated. For example, the lab activities for “Distributed Databases” and “Multimedia Databases” courses involve SQL Server and Oracle support, the ones for “Speech Synthesis and Recognition”, “Multimedia Data Compression and Encoding” and “Speech Compression” courses need Matlab. The lab activities for “Advanced Software Methods in Telecommunications” course need Rational, Visual Studio and JBuilder.

Each laboratory activity should be performed according to the tutor’s specifications. The specifications consist of theory, objectives, interactive tutorials, demonstrations and external resources. If the laboratory objectives are related to software development, customization or analysis, the virtual machine allocated to each student just creates an instance of the development environment or software package used for completing the tasks.

There are courses, such as “Mobile communications – 3G and 4G”, that also involve simulation packages and hardware equipment. The simulation packages such as QualNet network simulator, can be exposed in the same manner as the software packages or the development environments, even if they are connected to real hardware devices or not.

LabView can be also used for handling hardware devices. If exposing the hardware equipment via LabView, within the virtual machine, only one student or team can control it, at a moment. In order to avoid conflicts and protect the equipment and student work, the remote access to hardware devices must be optimized.

In the Technical University of Cluj-Napoca there are 3019 MSc students and 1432 PhD students registered in 9 faculties and following different educational programmes. Not just the diversity of themes and interdisciplinary character of MSc and PhD recommend the implementation of a learning cloud environment. Another important aspect refers to research management during the PhD mobility grants, where important priorities are knowledge transfer and approaching of new technologies.

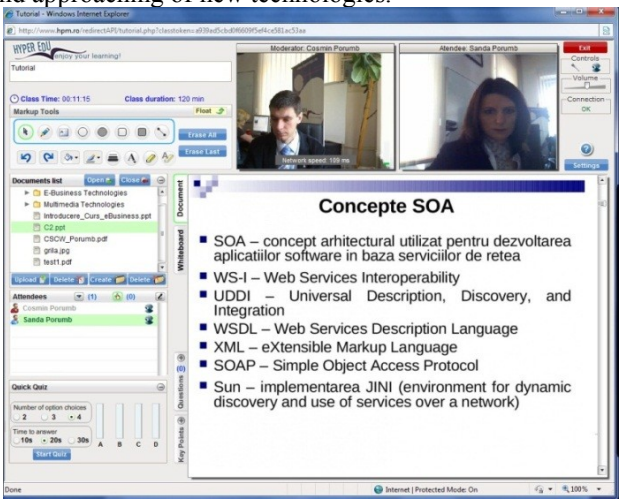


Figure 8. Scrum implementation in research projects.

Most of the diploma, dissertation and research projects comply with the Scrum methodology. So, the authors propose the blended learning approach and Agile Scrum methodology to be implemented in the project-based learning module. The semester and diploma projects will be developed according to Agile Scrum methodology. It allows iterative development and full control of the project phases. The students are grouped in virtual teams (2-3 members). Forum, messaging and online focus group, document management and sharing capabilities (see Figure 8) are added to the project module in order to allow team members to collaborate during the project.

In traditional engineering education, knowledge assessment consists of complex procedures such as periodical evaluation, project evaluation and the final knowledge assessment and it involves the teachers and students. The assessment model in the third generation of e-learning is learner-centered and it consists of questions with one or more correct answers, as well as open answers. So, the students should complete the online assessment tests and the teacher will receive notifications about students' tests and centralizes the results before closing the educational act. The presented prototype proposes a blending assessment method that preserves the traditional assessment methods and the flexibility the online assessment tools grant.

The learning environment allows the management staff to setup the own educational platform or invoke needed interactive/collaborative tools. The cloud computing paradigms (SaaS, PaaS and IaaS) enable transparent access to services, software packages or hardware infrastructure. This way, the head of a department/programme manager that already implemented an educational platform and prefers to use it instead of re-implementing a new approach can opt for transferring the platform onto the new e-learning cloud setup (based on the Infrastructure as a Service paradigm) or extend the existing functionality in order to support more features. It assumes the integration of the legacy system by using the Software as a Service paradigm.

The cloud computing environment, it is open for organizations and enterprises. The registration procedure is very simple: the responsible of an educational/training programme (MSc, PhD, even BSc) must complete the registration forms by specifying the requirements, then the intelligent configuration block automatically allocates the needed resources and creates the hardware and software components that support such a programme.

The learning management features include the statistics and reporting capabilities. The reporting and statistics components provide with the information related to education and research activities the actors performed within the platform:

- The number of educational resources and interactive materials created and uploaded into the platform;
- The number of assessment sessions the tutors created and scheduled per month/week/day;
- The number of synchronous collaborative learning sessions scheduled per month/week/day;
- How many students accessed the interactive materials per month/week/day and completed the periodical assessment sessions;
- How many students collaborated within the research/team projects and the contribution of each team member;
- How many topics have been created within the course forum and how many students participated to a topic;
- How many students used the multimedia messaging in order to communicate to the colleagues;
- The number of collaborative sessions the students scheduled within the research/team projects;
- The number of interactive tutorials the students met tutors in order to clarify important aspects regarding the educational content and activities;
- The number of students that completed the laboratory tasks according to the pre-defined schedule;
- The number of students that needed help during the laboratory tasks and how fast and clear was the tutor's support;
- How many team/research projects have been completed according to the pre-defined scheduled;
- How many students studying abroad have been assisted remotely;
- The number of MSc and PhD students are involved in virtual research teams;
- The number of interactive training sessions has been scheduled and delivered via hybrid classware;
- The number of virtual machines has been allocated for laboratory activities;
- The bandwidth usage per month/week/day (Figure 9);
- The CPU usage per virtual machine;
- The memory usage per virtual machine;
- The overloading per virtual setup.

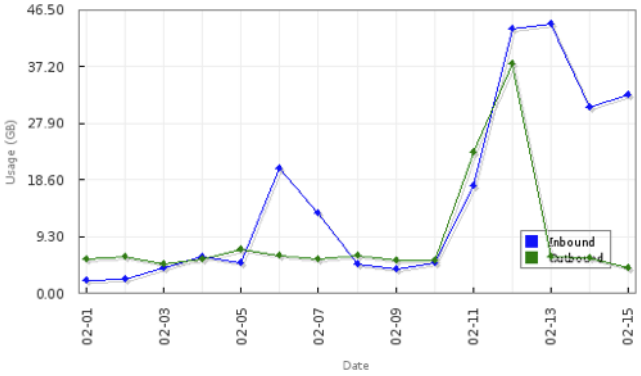


Figure 9. "Aurel Vlaicu" Univeristy. Bandwidth usage (February, 2011).

VI. CONCLUSIONS

The article presents the blended learning concept based on cloud computing paradigms and the manner it can be customized for higher and postgraduate education in

engineering. It starts from a functional analysis between traditional e-learning platforms and blended learning environments dedicated to higher and postgraduate education then it continues with the technological aspects and the deployment diagram of an e-learning cloud environment for engineering education.

If analyzing the e-learning cloud setup with the platforms presented in the introduction and related works, the advantages are conclusive: individual study support, Internet-based collaborative learning features, online access to lab infrastructure, collaborative research capabilities, project-based learning and problem-based learning functionality delivered using a complex but low cost infrastructure. Due to cloud computing (SaaS, IaaS and PaaS paradigms) implementation, the e-learning service providers can easily setup new learning environments or extend their existing systems in order to support blended learning capabilities.

The most important advantage of the cloud computing is the cost effectiveness. Instead of investing funds in the own e-learning infrastructure and educational software packages, the educational institutions should pay more attention to the content, staff, marketing and student enrollment, which can grant the service improvement. If opting for cloud-based services there are no IT costs, neither IT specialists to employ. The educational institutions register in the e-learning cloud and pay just what they consumed. The online access to collaborative learning tools and flexible individual study are implementing using SaaS paradigm. The development and deployment of laboratory applications use the PaaS concept. In order to implement laboratory equipment/infrastructure sharing or virtual desktop functionality, the faculties and departments can opt for IaaS services.

Such systems allow students to enroll in educational programmes even if the job is very restrictive because most of the learning activities can be remotely done. Several enhancements in the educational act have been identified. The implementation of the interactive learning approach in individual study grants a high retention factor (up to 80%) and the collaborative learning develops the soft skills and teamwork capabilities. The hybrid classware approach implements the synchronous collaborative learning methodologies and allow the students to actively participate to the educational act. Its main role is to keep the responsibility of learning on the teacher's end but also make students more responsible, communicate to each other and work and study as a team. Fundamental and applied research support, task management features and remote access to lab equipment and applications are also supported. The e-learning cloud setup should be considered as the most reliable solution for virtual laboratory and student assistance during the semester, diploma, dissertation or research projects.

ACKNOWLEDGMENT

The paper was supported by the project "Development and support of multidisciplinary postdoctoral programmes in major technical areas of national strategy of Research - Development - Innovation" 4D-POSTDOC, contract no. POSDRU/89/1.5/S/52603, project co-funded by the European Social Fund through Sectorial Operational Programme Human Resources Development 2007-2013.

REFERENCES

[1] D. E. Williams, "Virtualization with Xen(tm): Including XenEnterprise, XenServer, and XenExpress", Syngress Publishing House, ISBN: 1597491675 2010, 2007.

[2] M. Hicks and J.S. Foster, "Adapting Scrum to Managing a Research Group", Technical Report CS-TR-4966, University of Maryland, Department of Computer Science, 2010.

[3] J. A. Méndez and E. J. González, "Implementing Motivational Features in Reactive Blended Learning: Application to an Introductory Control Engineering Course", IEEE Transactions on Education, Volume: PP, Issue: 99, 2011.

[4] A. Vlaicu, S. Porumb, C. Porumb, and B. Orza, "Advanced Concepts for Interactive Learning", WSEAS Engineering Education, ISSN 1790-1979, Issue 6, Vol.3, 2006.

[5] R. Manseur and Z. Manseur, "A Synchronous Distance Learning Program Implementation in Engineering and Mathematics", Proc. 39th ASEE/IEEE Frontiers in Education Conference, San Antonio, Texas, USA, Pages 1-6, 2009.

[6] Y. Huixin, "Development of Blended Learning Modes and Its Practice in Computer Aided Language Learning Danikas", Proc. 2nd International Conference on Information Science and Engineering (ICISE), pages 2021-2024, 2010.

[7] M. Qiu and L. Chen, "A Problem-based Learning Approach to Teaching an Advanced Software Engineering Course", Proc. 2nd International Workshop on Education Technology and Computer Science, pages 252-255, 2011.

[8] Z. Hu and S. Zhang, "Blended/Hybrid Course Design in Active Learning Cloud at South Dakota State University", Proc. 2nd International Conference on Education Technology and Computer (ICETC), pages V1-63-V1-67, 2010.

[9] X. Laisheng and W. Zhengxia, „Cloud Computing: a New Business Paradigm for E-learning", Proc. 3rd International Conference on Measuring Technology and Mechatronics Automation, pages 716-719, 2011.

[10] S. Ouf, M. Nasr, and Y. Helmy, „An Enhanced E-Learning Ecosystem Based on an Integration between Cloud Computing and Web2.0", Proc. IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), pages 48-55, 2011.

[11] D. Chandran and S. Kempegowda, „Hybrid E-learning Platform based on Cloud Architecture Model: A Proposal", Proc. International Conference on Signal and Image Processing (ICSIP), pages 534-537, 2010.

 ISBN: 978-1-61208-153-3

On-demand Data Integration On the Cloud

Mahmoud Barhamgi[1], Parisa Ghodous[2], Djamal Benslimane[3]
Claude Bernard University (Lyon1)
69622 Villeurbanne, France
[1]Mahmoud.barhamgi@liris.cnrs.fr
[2]Parisa.Ghodous@liris.cnrs.fr
[3]Djamal.benslimane@liris.cnrs.fr

Abstract— **On-demand data integration is among the key challenges in Cloud Computing. In this paper, we present an ontology-based framework for describing and integrating data on the fly to answer transient business needs. We provide a semantic modeling for cloud's data services. The proposed modeling makes it possible to automatically resolve the different types of data heterogeneity that would arise when data from heterogeneous and autonomous providers need to be combined together to answer the business's data needs. We validate our approach with a prototype. The main contribution of this paper is an efficient on-demand integration system for the clouds.**

Keywords— On-demand data integration; Ontologies; Services.

I. INTRODUCTION

Cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet. Cloud computing is attractive to business owners as it eliminates the requirement for users to plan ahead for provisioning, and allows enterprises to start from the small and increase resources only when there is a rise in service demand. However, despite the significant benefits offered by cloud computing, the current technologies are not mature enough to realize its full potential. Many key challenges in this domain need to be addressed and solved. Data management and integration is among the key challenges that will keep receiving a particular attention from the research community over the coming years [6] [8] [14]. The Data-as-a-Service concept has been introduced in recent year as first step to virtualize access to data sources in clouds and SOA architectures [2][3][5][12]. A DaaS (Data-as-a-Service) service provides a simplified, integrated view of real-time, high-quality information about a specific business entity, such as a *Customer* or *Product*. The information that it provides may come from a diverse set of information resources, including operational systems, operational data stores, data warehouses, content repositories, collaboration stores, and even streaming sources in advanced cases.

Even though the introduction of DaaS services has allowed to shield the applications developers from having to directly interact with the various data sources that give access to business objects (i.e., *customers*, *orders*, *invoices*, etc.) and enabled them to focus on the business logic only, most of the time the business needs require the combination of multiple DaaS services from different service providers [13]. For instance, let us consider the following query: "*what are the driving directions for a facility of a given type (e.g., Restaurant, Theater, etc.) in a given city?*" -this is a typical application of Google maps *maps.google.com*. Let us assume that we have the following two DaaS services: S_1 returns the addresses of facilities of a given type in a given city; S_2 returns the driving directions between two given addresses. The execution of the above mentioned query involves the composition of S_1 and S_2 services. However, DaaS services composition is a hard task that may involve many data integration challenges. First, the semantics of DaaS services needs to be formally defined to automate their selection. The standardized service description languages (e.g., WSDL [17]) do not provide means for defining the services' semantics. Second, services may define different data structures for their manipulated data entities. For instance, the same piece of data such as "Address" may be represented differently by different DaaS services; i.e., the same data item has different XML structures. Structural data heterogeneities need to be addressed to allow for the automatic composition of DaaS services.

In this paper, we present an approach to compose cloud's DaaS services on the fly for the purpose of answering on-demand data integration needs. In the proposed approach, the semantics of DaaS services are defined using domain ontologies. This allows for automating their selection and composition and makes it possible to resolve the schematic data heterogeneities (a.k.a. structural data heterogeneities) of data items exchanged among heterogeneous DaaS services. We present also a system that exploits the proposed semantic modeling to compose DaaS services.

 ISBN: 978-1-61208-153-3

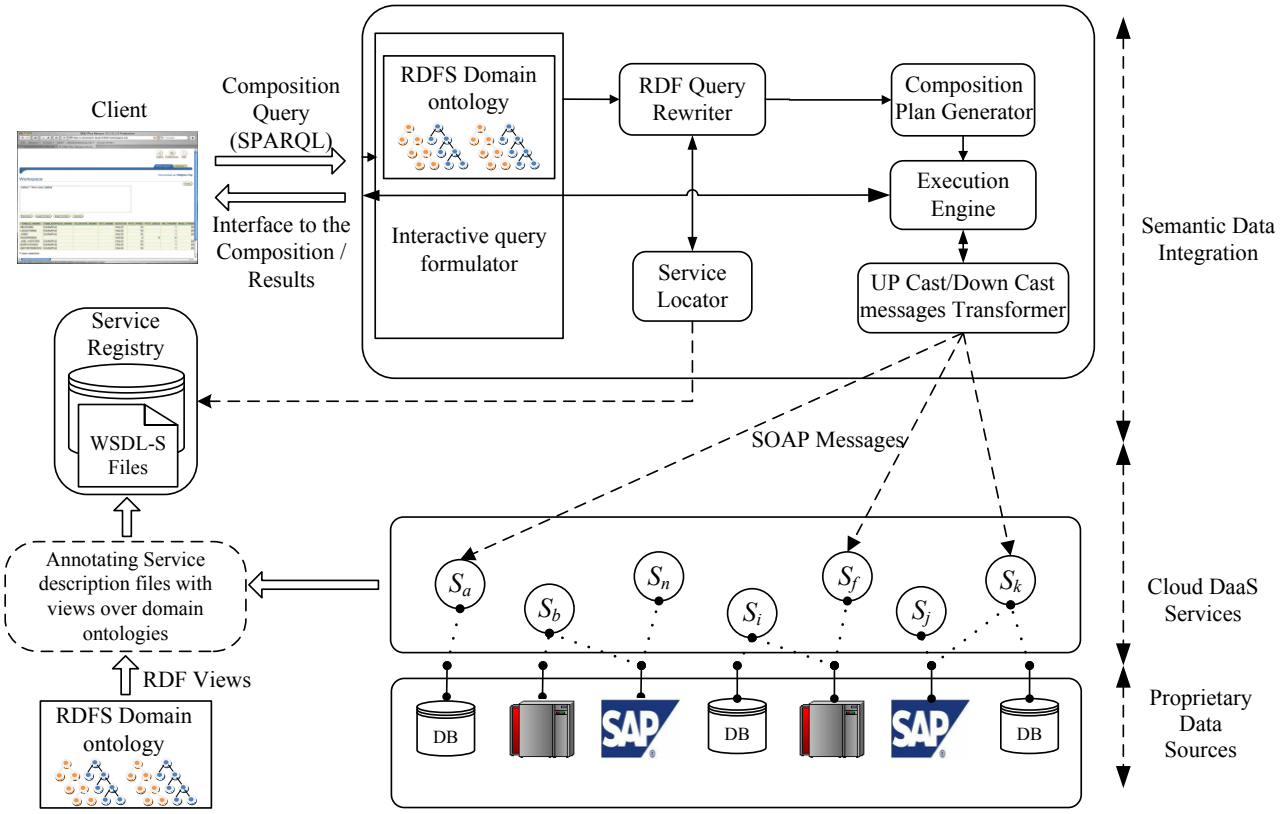


Figure 1: An overview of the proposed declarative approach to cloud services composition

The rest of this paper is organized as follows. In Section 2, we describe our framework for on demand data integration. In Section 3, we present our modeling to cloud DaaS services and users' queries. In Section 4, we showcase through an example how data integration queries are resolved by query rewriting and DaaS service composition. In Section 5, we overview related work. We provide concluding remarks in Section 6.

II. A Declarative Approach to Compose Cloud DaaS Services

In this section, we present a declarative framework for composing cloud DaaS services that addresses the challenges discussed earlier in the introduction. We show the different phases involved in DaaS services composition, starting from the service modeling to the generation of the final composition that will be returned to users.

Figure 1 presents our DaaS service composition framework. The first step towards the automation of DaaS services composition is to semantically represent their capabilities. In our approach, we model DaaS services as *RDF views* over domain ontologies. An RDF view uses concepts and relations whose meanings are formally defined in domain ontologies to define the semantics of a DaaS service. The RDF views are then used to annotate the service description files (e.g., WSDL files, SA-Rest, etc.).

Users (i.e., cloud application developers) in our approach formulate their composition queries over domain ontology using the do facto ontology query language SPARQL [18]. Non-savvy users can be assisted in formulating their queries by the `Interactive Query Formulator` component. Based on our proposed modeling to DaaS services (i.e., RDF views), the well-known query rewriting techniques can be used to compose them; i.e., our composition system rewrites the received queries in terms of available DaaS services using a query rewriting algorithm. For that purpose, we have devised an efficient RDF-oriented query rewriting algorithm [1]. The algorithm is implemented by the `RDF Query Rewriter` component and exploits the semantic annotations that we added in the service description files to select and compose the DaaS services that are relevant to the query. The composition system will then arrange the selected services in the composition execution plan (this is carried out by the `Composition Plan Generator` component). The composition plan will be displayed to the users, who can then invoke the compositions with their inputs. Note that when service providers define the semantics of their DaaS services using the RDF views over domain ontologies, they also provide the mappings between the defined views and the XML schemas of input and output messages of their

services. The mappings are also attached to the service description files as annotations and are used by the *Up-Cast/Down-Cast Messages Transformer* component when invoking component services. This is necessary since the same data item may have different structures between the ontology and the XML schemas of Input and Output messages (for instance, a (datatype) property in the ontology like "NAME" may be represented by two elements "FirstName" and "LastName" in an Input or Output XML schema). We detail all of the previous steps in the subsequent subsections.

III. A Semantic Description for DaaS Services and Composition Queries

In our approach, we model DaaS services as *RDF views* over domain ontologies. An RDF view describes the semantics of a DaaS service in a *declarative way* using concepts and relations whose meanings are formally defined in domain ontologies. Consider, for example, the services: $S_1(\$t,\$c,?n,?s,?b)$ and $S_2(\$c_1,\$s_1,\$b_1,\$c_2,\$s_2,\$b_2,?r)$ that we will use throughout the paper. Inputs are prefixed with "$" and outputs with "?". S_1 returns the facilities of a given type "*t*" (e.g., hospitals, hotels, etc) in a given city "*c*". The service S_2 returns the driving directions "*r*" between two addresses represented by the cities ("c_1" and "c_2"), the streets ("s_1" and "s_2") and the buildings ("b_1" and "b_2"). These two services can be composed together to look for facilities of a given type and obtain the driving directions to them. Figure 2 (Part-A) shows a graphical representation of the RDF views defined for S_1 and S_2. The RDF views in Figure 2 describe the semantics of services from the ontology point of view, where the blue ovals are concepts in ontology (e.g., *Facility*, *Address* and *Route*) whereas the arcs are properties. The defined RDF views are then used to annotate the service description files (e.g., WSDL files, SA-Rest, etc). These views define the semantics of services in a formal way and will be used during the selection and composition of DaaS services.

In the proposed approach, users (i.e., application developers) need only to focus on the needed data by formulating their composition queries over domain ontologies. They are not required to manually select services and build the composition plan by mapping the inputs and outputs of component services to each other and drop code to resolve data incompatibilities. Figure 2 (Part-B) shows a graphical representation of the query in the running example.

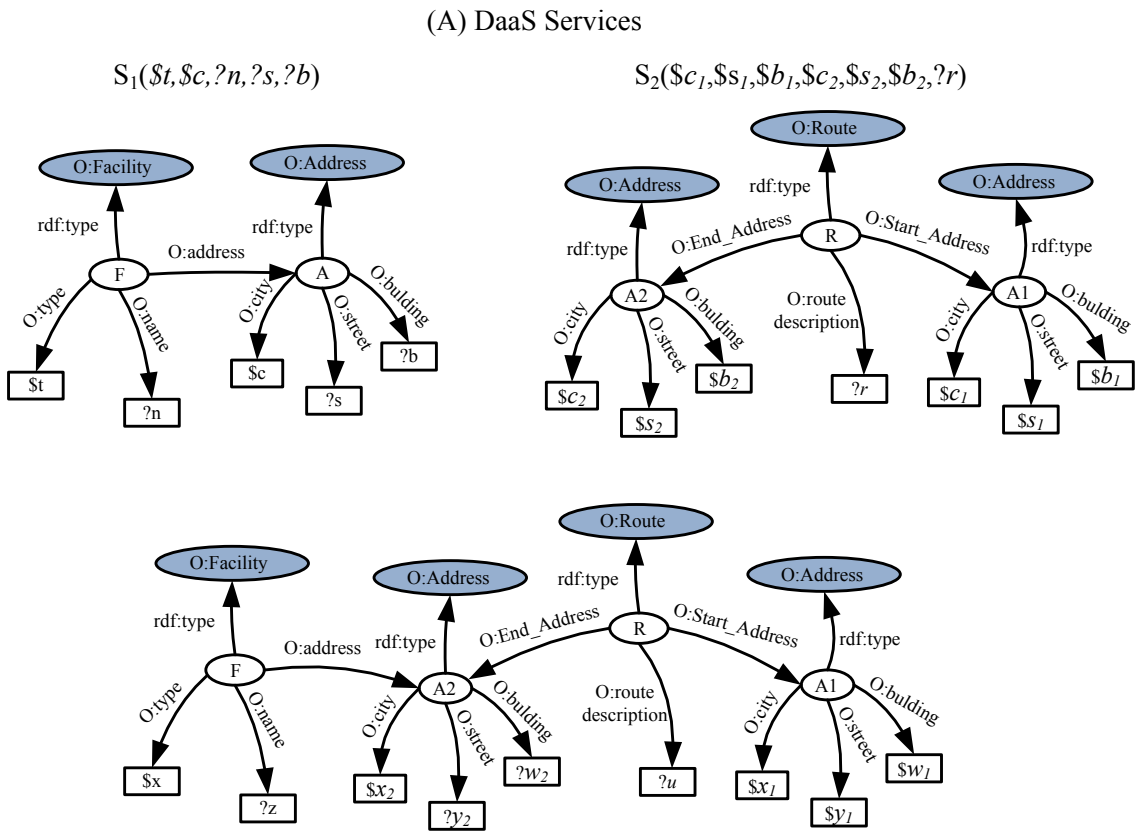


Figure 2: (A) the RDF views of services in the running example; (B) the user mashup query formulated on domain ontologies.

We will see in subsequent sections that they are still able to select the services participating in the resulting composition.

IV. COMPOSING DAAS SERVICES BY QUERY REWRITING

Our proposed composition approach relies on an RDF query rewriting algorithm (presented in [1]) to resolve the users' composition queries. Specifically, users' queries are matched against the RDF views of available services. These RDF views can be retrieved from the services description files (e.g., WSDL files). In the matching process, our matching algorithms identify the RDF sub-graphs of the query that can be covered by individual DaaS services. For example, as we can see in Table 1, the service S_1 covers the following nodes of the query: *F($x,?z)*, *A2($x$_2$,?y$_2$,?w$_2$)* and the object property linking the two *address(F,A2)*. The service S_2 covers the following nodes of the query: A2(x_2$,?y$_2$,?w$_2$),R(?d), A1($x$_1$,?y$_1$,?w$_1$) and the object properties : *end_address(R, A2)*, *start_address(R,A1)*.

Service	Covered sub-graphs
S_1($x,?z,$x$_2$,?y$_2$,?w$_2$)	F($x,?z), A2($x$_2$,?y$_2$,?w$_2$), address(F,A2)
S_2(x_1$,$y$_1$,w_1$,$x$_2$,y_2$,$w$_2$, ?u)	A2(x_2$,?y$_2$,?w$_2$), R(?d), A1($x$_1$,?y$_1$,?w$_1$),end_address(R, A2), start_address(R,A1)

Table 1: the query's sub-graphs that are covered by services in the running example

If these two services are combined together, the whole nodes and object properties sets of the query will be covered. Therefore, our composition algorithm will combine both of these services and consider the combination as a rewriting of the query as follows:

$$Q(?z,?y_2,?w_2,?u) :- S_1(\$x,?z,\$x_2,?y_2,?w_2) \times S_2(\$x_1,\$y_1,\$w_1,\$x_2,\$y_2,\$w_2,?u)$$

The composition algorithm will then orchestrate the used DaaS services in the rewriting to produce the composition execution plan that will be displayed to the user for further customization (if desired).

Figure 3 (A) shows the interface to the composition system. Users formulate their composition queries in the query panel using SPARQL language and submit the query to the system. The composition system will compose the DaaS services and present the user with composition plan in Figure 3 (B), where users can refine the composition by selecting the desired services among the possible ones and validate the composition. The composition system will then present the user with an interface where the users can specify specific values for the mashup parameters and invoke it. Figure 3 (A) shows the composition inputs values and the obtained outputs for the running example.

V. RELATED WORKS

Since the DaaS services composition research problem is relatively new, there has been only a small amount of research work addressing it. In the following, we review the most prominent ones of these works.

A considerable body of recent work addresses the problem of composition (or orchestration) of multiple web services to carry out a particular task, e.g., [15][16]. In general, that work is targeted more toward workflow-oriented applications (e.g., the processing steps involved in fulfilling a purchase order), rather than applications coordinating data obtained from multiple DaaS services, as addressed in this paper. Although these approaches have recognized the importance of automating the composition process, they have not, as far as we are aware, addressed the DaaS services.

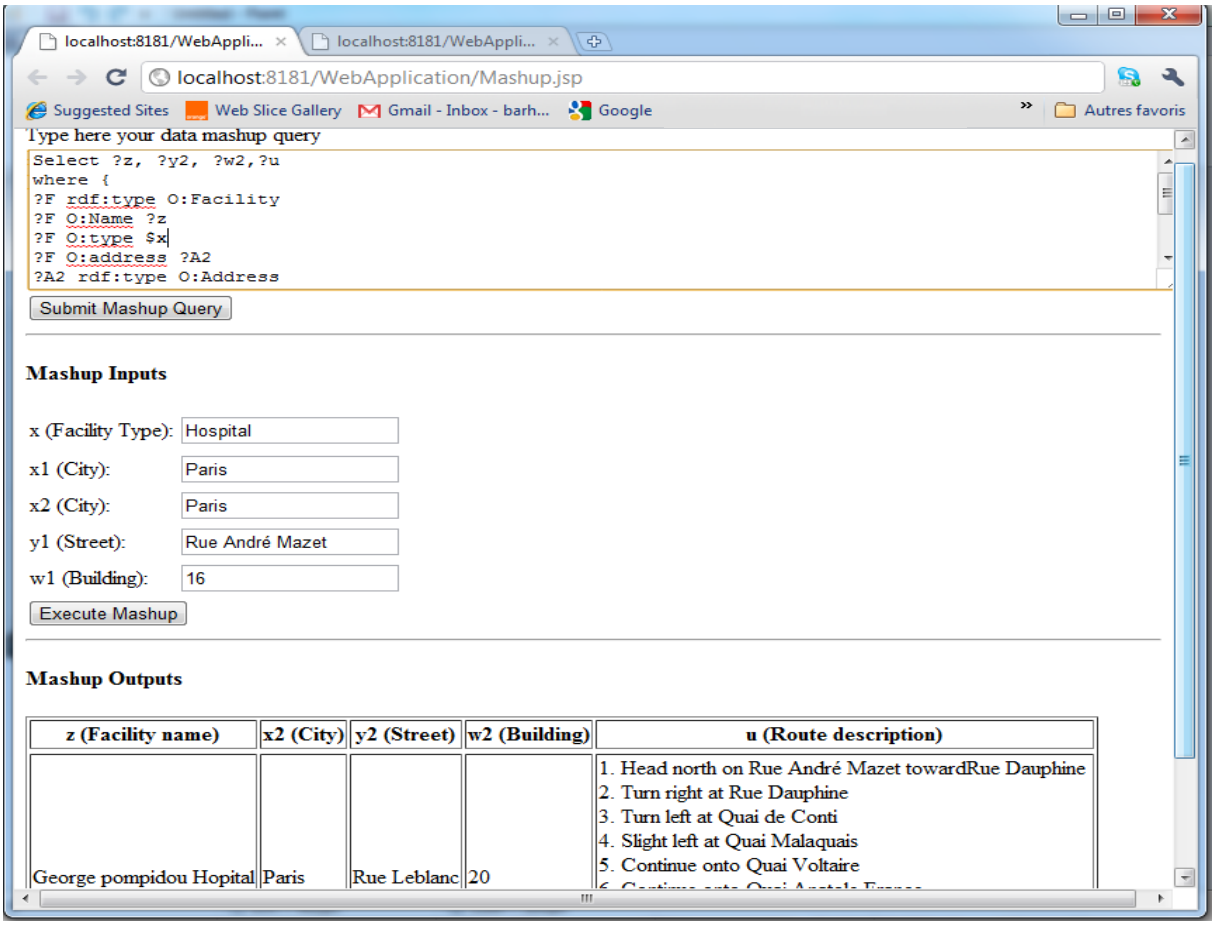
The Web Service Mediator System WSMED [9] allows users to mashup data services by defining relational views on top of them. Users can then query data by formulating their mashup queries over defined views. Users can also enhance defined views with primary-key constraints which can be exploited to optimize the mashups. The main drawback of the WSMED system is its high reliance on users; i.e. users are supposed to import the services relevant to their needs; define views on top of them and enhance the views with primary key constraints. The latter task requires from users to have a good understanding of the services' semantics. In our system, DaaS Web services are modeled as RDF views over domain ontologies where *primary key* constraints are defined explicitly by the concepts' skolem functions, thus the discussed *Primary key* based optimizations are included by default in our query processing model.

In other academic mashup systems [4][7][10][11], data mashup users are required to select the data services manually (which assumes they are able to understand their semantics), figure out the execution plan of selected services (i.e. the services *orchestration* in the mashup) and connect them to each other and drop code (in JavaScript) to mediate between incompatible inputs/outputs of involved services. This prevents average users from mashing up DaaS services at large. Our composition system addresses this limitation by proposing a declarative composition approach, where users need only to focus on the required data and the system will find and compose the services for them.

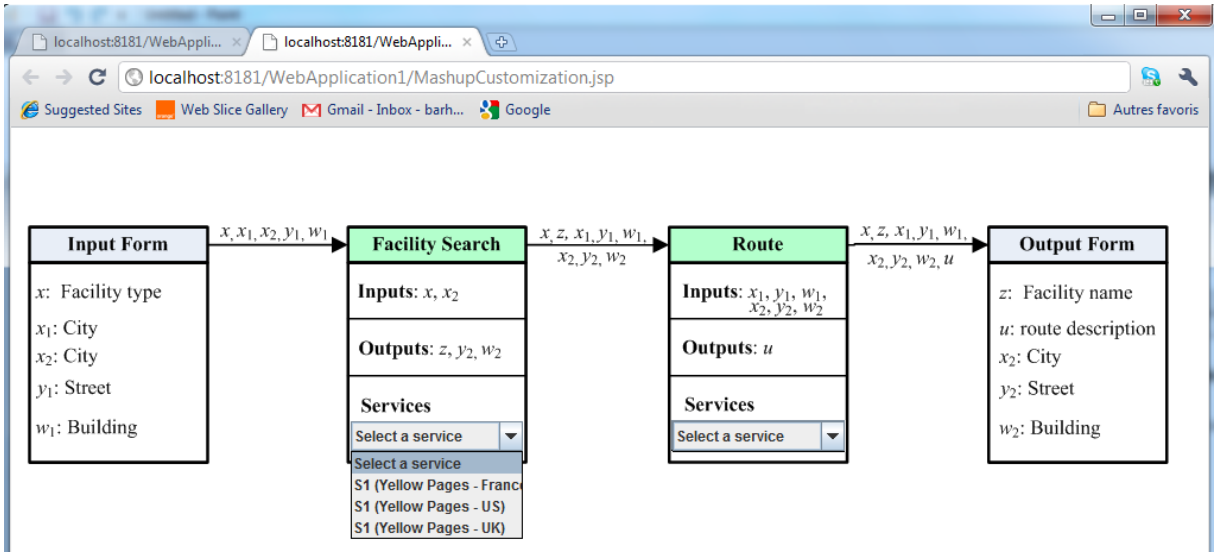
VI. CONCLUSION

In this paper, we presented an approach that caters for on-demand data integration for cloud business's data needs. We presented an ontology-based semantic modeling for cloud DaaS services. The proposed modeling makes it possible to automatically combine heterogeneous DaaS services and resolve the different types of data heterogeneity that would arise when data needs to be exchanged between composed services. We also validated our approach with a prototype. As a future work, we intend to contextual data heterogeneities between composed services (i.e., when composed services have different interpretation contexts for the data they exchange).

 ISBN: 978-1-61208-153-3



(Figure 3-A): The Mashup Interface: users type their mashup queries in the query panel, they will be presented then with the interface "*Mashup Inputs*" that is used to specify the values of input parameters to execute the mashup



(Figure-3-B): The Mashup Customization Interface MCI: the MCI allows users to select the desired services among the possible ones.

REFERENCES

[1] Mahmoud Barhamgi, Djamal Benslimane, and Brahim Medjahed, "A Query Rewriting Approach for Web Service Composition," *EEE Transactions on Services Computing (TSC)*, pp. 206-222, 2010. http://www.computer.org/portal/web/csdl/doi/10.1109/TSC.2010.4

[2] Michael J. Carey, "Data delivery in a service-oriented world: the BEA aquaLogic data services platform.," in *SIGMOD Conference*, 2006, pp. 695-705.

[3] Asit Dan, Robert Johnson, and Ali Arsanjani, "Information as a Service: Modeling and Realization," in *International Conference on Software Engineering (Workshop on Systems Development in SOA Environments)*, 2007, pp. 2-10.

[4] Hazem Elmeleegy, Anca Ivan, Rama Akkiraju, and Richard Goodwin, "Mashup Advisor: A Recommendation Tool for Mashup Development," in *2008 IEEE International Conference on Web Services (ICWS 2008)*, Beijing, China, pp. 337-344.

[5] Mike Gilpin et al., "Information-As-A-Service: Waht's Behind This Hot New Trend?," Forrester Research, Research Report 2007. http://www.forrester.com/rb/Research/information-as-a-service_whats_behind_this_hot_new_trend/q/id/41913/t/2, accessed on 29 June, 2011.

[6] Hector Gonzalez et al., "Google fusion tables: data management, integration and collaboration in the cloud," in *SoCC*, 2010, pp. 175-180.

[7] Anne H. H. Ngu, Michael Pierre Carlson, Quan Z. Sheng, and Hye-young Paik, "Semantic-Based Mashup of Composite Applications," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 2-15, 2010.

[8] Raghu Ramakrishnan, "Data Management in the Cloud," in *ICDE 2009*, pp. 5, 2009.

[9] Manivasakan Sabesan and Tore Risch, "Adaptive Parallelization of Queries over Dependent Web Service Calls," in *1st IEEE Workshop on Information & Software as Services, WISS 2009*, Shanghai, China, 2009.

[10] Junichi Tatemura, "UQBE: uncertain query by example for web service mashup," in *SIGMOD Conference*, Vancouver, Canada, 2008, pp. 1275-1280.

[11] Junichi Tatemura, "Mashup Feeds: : continuous queries over web services," in *SIGMOD Conference*, 2007, pp. 1128-1130.

[12] Hong-Linh Truong and Schahram Dustdar, "On Analyzing and Specifying Concerns for Data as a Service," in *The 2009 Asia-Pacific Services Computing Conference (IEEE APSCC 2009)*, Singapore, 2009, pp. 7-11.

[13] Qi Yu, Xumin Liu, Athman Bouguettaya, and Brahim Medjahed, "Deploying and managing Web services: issues, solutions, and directions," *VLDB Journal*, vol. 17, no. 3, pp. 537-572, 2008.

[14] Qi Zhang, Lu Cheng, and Raouf Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7-18, 2010.

[15] Mazen Shiaa, Jan Ove Fladmark, and Benoit Thiell, "An Incremental Graph-based Approach to Automatic Service Composition" Proc. of the Int. Conf. on Services Computing (SCC'08), Honolulu, pp. 212-220, 2008.

[16] Patrick Hennig and Wolf-tilo Balke, "Highly Scalable Web Service Composition Using Binary Tree-Based Parallelization," Proc. of the Int. Conf. on Web Services (ICWS'10), Los Alamitos, pp.123-130, USA, 2010.

[17] http://www.w3.org/TR/wsdl, accssed on 29 June, 2011

[18] http://www.w3.org/TR/rdf-sparql-query/, accssed on 29 June, 2011

 ISBN: 978-1-61208-153-3

UnaCloud: Opportunistic Cloud Computing Infrastructure as a Service

Eduardo Rosales, Harold Castro, Mario Villamizar
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá D.C., Colombia
{ee.rosales24, hcastro, mj.villamizar24}@uniandes.edu.co

***Abstract*—This paper presents UnaCloud: an opportunistic cloud computing Infrastructure as a Service (IaaS) model implementation, which provides at lower cost than dedicated cloud infrastructures, basic computing resources (processing, storage and networking) to run arbitrary software, including operating systems and applications. The IaaS model is provided through the opportunistic use of idle computing resources available in a university campus. UnaCloud deals with the problems associated to use commodity, non-dedicated, distributed, and heterogeneous computing resources that are part of different administrative domains. We propose an IaaS architecture based on two strategies: an opportunistic strategy that allows the use of idle computing resources in a non-intrusive manner, and a virtualization strategy to allow the on-demand deployment of customized execution environments. The proposed solution was implemented and tested through the provision of an opportunistic IaaS model, evidencing high efficiency in the deployment of virtual machines for academic and scientific projects.**

Keywords; grid computing; cloud computing; desktop grid; infrastructure as a service; unacloud; unagrid.

I. INTRODUCTION

Grid computing and cloud computing appear to be the two latest and most promising computing paradigms [1]. Grid computing is considered a paradigm in production, which surged as a vanguard technology for supporting the development of different scientific projects at a global scale [2]. In contrast, cloud computing is still an evolving paradigm. Its definitions, use cases, underlying technologies, issues, risks, and benefits will be refined in a spirited debate by the public and private sectors. These definitions, attributes, and characteristics will evolve and change over time [3]. However, cloud computing is considered the grid computing successor [4], because it represents a disruptive evolution, aimed at the customization and delivery of computing services. These services hide most of the complexities associated with the underlying infrastructure administration, can be deployed on demand and are accessed remotely via Internet [1].

There are high expectations about cloud computing paradigm for the next 1-5 years [5]. Cloud computing is attracting a lot of attention around the world [6], not only of experts in ICTs, but also academics, scientists, researchers, businessmen and common people, who are attracted by the delivery of on-demand computing services. However, there are a few cloud computing implementations, most of them exclusively based in the IaaS model, due in part to the complexity associated to the different cloud computing service delivery models (IaaS, Platform as a Service – PaaS, and Software as a Service - SaaS). Furthermore, all IaaS implementations (open source or commercial) require expensive, dedicated, robust and high performance underlying infrastructures, so they are unviable in organizations and countries with low financial resources.

Taking into account the emerging importance of cloud computing paradigm, the need of independent investigation testing of commercial providers, the financial difficulties associated with expensive underlying infrastructures, and the different cloud computing service models, in this paper we present UnaCloud, an IaaS model implementation, which provides basic computing resources through the opportunistic use of idle computing resources available in a university campus.

UnaCloud is able to deploy, manage and deliver an opportunistic IaaS model based on preexisting, non-dedicated, distributed, and heterogeneous computing resources that are part of different administrative domains. These resources are in part, conventional desktop computers, as those daily used by employees, professors or students in a university campus. These desktop computers tend to be underutilized for significant periods, resulting in plenty of idle computing resources. Due to the large amount of available computing resources on a university campus, UnaCloud represents an economically attractive solution for constructing and deploying large scale computing infrastructures, avoiding not only underutilization of non-dedicated computing resources, but also financial investments in hardware and maintenance costs associated.

UnaCloud has been initially deployed at Universidad de los Andes and, in this work, the design and details of the implementation deployed are presented along with the results obtained. The paper is organized as follows: section 2 presents the related works to IaaS model implementations and Desktop Grids and Volunteer Computing Systems (DGVCS's). Section 3 presents the UnaCloud architecture in terms of its services. Section 4 presents UnaCloud implementation. Section 5 presents the UnaCloud testing and results. Finally, Section 6 presents the conclusions and future work.

 ISBN: 978-1-61208-153-3

II. RELATED WORK

UnaCloud represents a convergence between cloud computing and DGVCS's. The service delivery model of the cloud computing paradigm is taken into account as an objective, mainly in relation to the IaaS model. The design aspects of the DGVCSs are kept into account as a mean to provide an opportunistic underlying infrastructure to support cloud computing services at lower cost. This type of convergence has been theoretically analyzed in [7], on the perspective of software engineering principles.

In the context of IaaS models, Amazon Web Services (AWS) [8] is considered a precursor because it is in productive use since 2006, offering basic processing and storage capabilities via Internet. Amazon Elastic Compute Cloud (Amazon EC2) [9] and Amazon Simple Storage Service (Amazon S3) [10] popularized a commercial IaaS model, based on a pay-per-use contract and the provision of resizable compute capacity in the cloud. The OpenNebula [11] project was the first open source tool that extended the benefits of cloud computing technologies to data centers and clusters, transforming physical infrastructures in virtual infrastructures of high flexibility and performance. Eucalyptus [12] is the first research-oriented open source software implementation that utilizes compute clusters in order to foster community research exploration of cloud computing systems. Nimbus [13] is an open source toolkit that allows transforming clusters into an IaaS model able to interoperate with grid computing conventional tools, including: Globus Toolkit, Sun Grid Engine (SGE) or PBS.

On the other hand, in the context of DGVCS's, the Worm [14] and Condor [15] projects are pioneers in the opportunistic use of homogeneous computing resources connected by LAN infrastructures. The GIMPS [16] and SETI@home [17] projects are characterized by their unique purpose, Internet scalability and the ability to leverage non-dedicated, distributed and heterogeneous computing resources (at the hardware, system and administrative domain level).

The Distributed.net [18] and BOINC [19] projects are characterized by an approach not limited to a unique purpose, being able to support multiple distributed scientific research projects. The last four projects described, are based on lightweight, portable and easy to install agents/clients that are continuously running as a background process in low priority, leveraging idle computing resources in a non-intrusive manner. Finally, projects like Bayanihan Computing. NET [20], OurGrid [21], Integrade [22] and UnaGrid [23], offer specialized support to cluster and grid computing initiatives with large processing demands, assuming the deployment of middleware and workload management systems to process multiple jobs. In the Nebulas project [24], different requirements and possible solutions to build customizable clouds (called Nebulas) using distributed voluntary resources are proposed; however, they are neither implemented, nor evaluated.

Unlike the commercial and academic IaaS model implementations, UnaCloud does not require large financial investments to purchase and maintain cluster architectures composed by multiple nodes exclusively dedicated to the provision of the virtual machines resources. In contrast, UnaCloud uses a commodity and non-dedicated underlying infrastructure, implementing opportunistic design concepts broadly studied in the context of DGVCS's.

UnaGrid is the first on-demand opportunistic Desktop Grid [23]. It uses virtualization technologies to deploy Customized Virtual Clusters (CVC) based on an opportunistic underlying infrastructure. Due to the above, the UnaGrid infrastructure is able to support cloud computing service models, even though UnaGrid functionalities are currently focused on cluster and grid computing technologies.

To the best of our knowledge, our work is the first to analyze the prospect and performance of using an opportunistic underlying infrastructure to support an IaaS model.

III. UNACLOUD ARCHITECTURE

UnaCloud began as a research effort to explore and obtain the innovative features and advantages of cloud computing paradigm. This effort is aimed at the provision of computing infrastructures for the development of e-Science projects and to support computing related activities. To achieve this, one of our most important limitations is the funds to purchase the dedicated computing resources required by all IaaS model implementations (even open source IaaS model implementations).

Therefore, the UnaCloud objectives require the extension of DGVCS's design concepts to provide an opportunistic underlying infrastructure able to support an experimental IaaS model at lower cost. In spite of the multiple problems related to use a non-dedicated infrastructure, functionalities included in UnaCloud are supposed to be similar to those available in conventional IaaS models. However the availability of the computing resources is dependent on the behavioral pattern of their currently owners, so it is normally not effective to ensure any type of QoS or SLA. Thus, UnaCloud works on a best-effort basis.

UnaCloud architecture is based on the integration of an information system with an underlying computing infrastructure, that is, a Web portal capable of coordinating information and communications on opportunistic infrastructures to provide basic computing services, operating systems and applications through a cloud computing IaaS model. The UnaCloud architecture overview is illustrated in Fig. 1.

As shown in Fig. 1, there are four types of UnaCloud users. An IaaS user demands the IaaS model without specifying the deployment location. IaaS users access the UnaCloud Web interface to customize and/or deploy virtual machines with general-purpose configurations (e.g., virtual machines used to support academic activities). Grid users demand IaaS model specifying the deployment location on specific underlying infrastructure computers. Grid users access the UnaCloud Web interface to customize and/or deploy suitable execution environments for e-Science applications (e.g., cluster, grid or cloud computing environments). IaaS-Grid users can take any of the above

 ISBN: 978-1-61208-153-3

roles. Additionally, Administrators access to all available Web interfaces, with all privileges and get exclusive access to administration services.

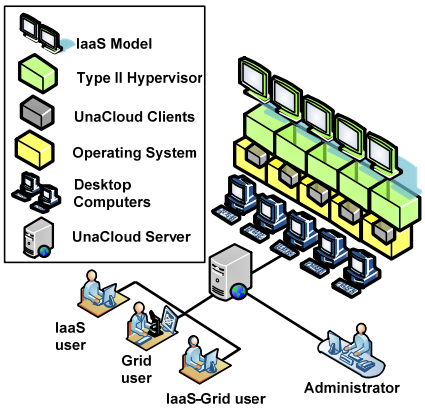


Figure 1. UnaCloud architecture overview.

UnaCloud architecture is divided into two main components: UnaCloud Server and UnaCloud Client. These components are implemented using open source and loose coupling information and communication technologies, which promote the UnaCloud interoperability and extensibility, and are appropriated to the special conditions of a commodity opportunistic underlying infrastructure.

A. *UnaCloud Server Architecture*

UnaCloud Server is a Web application whose main function is to provide an entry to all UnaCloud services, including the provision of customization, deployment, access, management and monitoring interfaces. As shown in Fig. 2, UnaCloud Server is composed of three layers:

- *Interface layer*: is a Web portal that supports a user Web Interface (WI), which supports the presentation for accessing and consuming all available UnaCloud services. This interface provides an IaaS model based on self-service. This layer is also responsible for managing the user information, including secure access through authentication and authorization mechanisms. The Web portal is available via Internet and so, can be accessed using any Web browser.
- *Core layer*: is responsible for processing all user requirements and deliver solutions in the form of UnaCloud services. The first service supported is the Customized Environment Manager (CEM), which processes and prepares orders related to all of customization settings (made through WI), including availability verifications of the computing resources. The availability verifications are performed through a virtual and physical resources information database that is managed by a service named Persistence Manager (PM). PM is also responsible for managing the operations used to provide basic IaaS traceability reports. The next core layer service is Virtual Machine Manager (VMM), which works in conjunction with PM to manage the virtual machine information. VMM is also responsible for preparing hypervisor orders to operate all the IaaS virtual machines, including: start, stop, restart and monitor operations. Finally, Physical Infrastructure Manager (PIM) service, works in conjunction with PM to manage the physical machines information. PIM is also responsible for preparing operating system orders to operate the entire underlying infrastructure, including basic operations such as: turn off, restart, logout and monitoring.

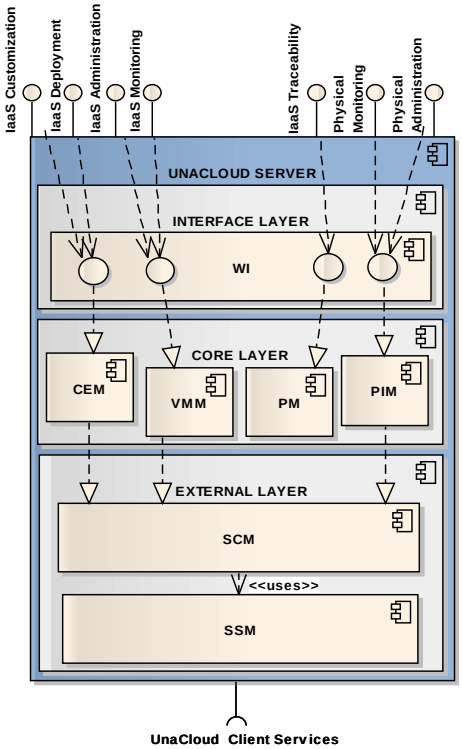


Figure 2. UnaCloud server architecture.

- *External layer*: is responsible for managing the communication services on the server side to deliver all the UnaCloud Server orders to the UnaCloud Clients. The first service supported is Server Communication Manager (SCM), which supports the connection, disconnection and message passing between UnaCloud Server and UnaCloud Client. SCM works in conjunction with Server Security Manager (SSM) service, which is responsible for managing the security schema in communications, including confidentiality and non-repudiation mechanisms.

B. *UnaCloud Client Architecture*

UnaCloud Client is a lightweight, highly portable and easy to install client which is installed and run directly on the underlying opportunistic infrastructure. This Client is based on the design concepts of agents/clients implemented on

 ISBN: 978-1-61208-153-3

DGVCS's such as: GIMPS, Distributed.net and SETI@home (studied in Section 2). These design concepts proposed the execution of background and low priority processes to use idle computing resources in a non-intrusive manner. UnaCloud Client incorporates these concepts, but apply them to the virtual machine execution processes, facilitating not only the deployment of an opportunistic IaaS model, but also the continuous and optimized utilization of the underlying infrastructure in a time-sharing hardware model.

UnaCloud Client is responsible for receiving and processing all of UnaCloud Server orders to provide a dynamic and on-demand IaaS model. To achieve this, as shown in Fig. 3, UnaCloud Client is composed of two layers:

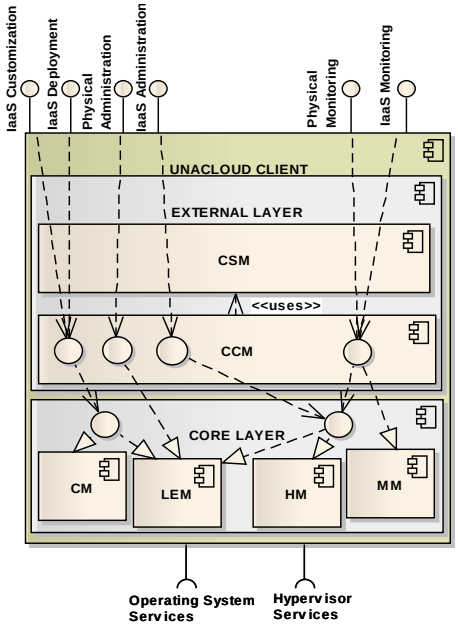


Figure 3. UnaCloud client architecture.

- *External layer*: is responsible for managing the communication services on the client side. The first service supported is Client Communication Manager (CCM), which supports the connection, disconnection and message passing between UnaCloud Client and UnaCloud Server. CCM works in conjunction with the Client Security Manager (CSM) service, which supports confidentiality and non-repudiation mechanisms for secure message passing.
- *Core layer*: is responsible for attending and meeting UnaCloud Server orders through local operating system and hypervisor invocations. The first service supported is Context Manager (CM), which is the counterpart in the client of CEM, and is responsible for adapting the virtual machine execution context to all of customization settings required by an end-user through WI. The next service is Local Executor Manager (LEM), which executes multiple commands using invocations to the local operating system services. LEM executes all commands required to meet the orders sent by VMM and PIM from the UnaCloud Server side. The next service is Hypervisor Manager (HM), which executes multiple commands using invocations to the local hypervisor. HM executes all commands required to meet the orders sent by VMM from the UnaCloud Server side. Finally, Monitoring Manager (MM) service is responsible for monitoring the state of CPU, RAM and SWAP memory, hard disk, network and operating system variables on the physical machine where UnaCloud Client is running.

UnaCloud Client can be installed on any desktop computer or server using Windows, Linux or Mac operating systems. It supposes a horizontal scaling model, based on the easy aggregation of single desktop computers or entire computer laboratories.

IV. UNACLOUD IMPLEMENTATION

To meet UnaCloud objectives, its implementation of an opportunistic IaaS model is able to provide the following services:

- *IaaS customization*: UnaCloud allows the customization of execution environments through five settings: software, hardware, quantity, location (optional) and time. Software settings allow customizing the type of operating system, its version and all applications installed on it, after the deployment new applications can be installed on the VMs. Hardware settings allow customizing hard disk and RAM memory sizes, and the CPU cores number. Quantity setting allows choosing the instances number to deploy. Location setting allows choosing the IaaS model deployment location on specific underlying infrastructure computers. The last setting only applies to Grid users who desire to optimize and document the use of the opportunistic infrastructure. Finally, time setting allows configuring the IaaS execution time. For users who want to skip the full IaaS customization process, settings only involve the selection of the IaaS deployable image name, the instances number to deploy, the location (only for Grid users) and the execution time of the deployment.
- *IaaS deployment*: UnaCloud allows the on-demand deployment of the execution environments, customized in the previous service. The IaaS deployment includes the provision of necessary data for its remote access, using standard mechanisms such as: Remote Desktop, VNC or SSH. The remote access data provided includes: the virtual machine name and IP address, the remote access mechanism name and port and, the guest operating system root user and password (UnaCloud deliver virtual machines with root privileges).
- *IaaS administration*: UnaCloud allows operating virtual machines, including basic operations such as start, stop, restart, change execution time and monitoring.

 ISBN: 978-1-61208-153-3

- *IaaS traceability*: UnaCloud allows checking the IaaS model traceability at user level. UnaCloud delivers a basic report that includes information associated with deployed virtual machines, the underlying infrastructure used on the deployment, IaaS customization settings chosen by the UnaCloud user and the execution period selected.
- *Physical infrastructure administration*: UnaCloud allows operating physical machines that compose the underlying infrastructure, including basic operations such as: turn off, restart, logout and monitoring. This functionality is only available for administrators.

V. UNACLOUD TESTING AND RESULTS

UnaCloud Client has been deployed in three computer laboratories (Waira I, Waira II and Alan Turing) at Universidad de los Andes. Each laboratory has 35 computers with Intel Core 2 Duo (1.86GHz) processors, 4GB of RAM and Windows XP as their main operating system. In addition, UnaCloud Server was deployed on a virtual machine running on a server, which is located in the data center (for availability reasons) of the Department of Systems and Computing Engineering. As illustrate in Fig. 4, the networking infrastructure is based on three switches and a multilayer switch interconnected via a GigE LAN.

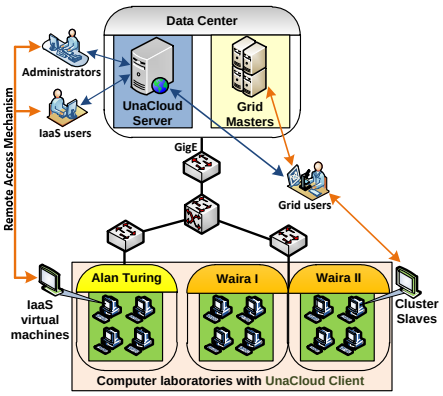


Figure 4. UnaCloud deployment.

UnaCloud Client runs only one virtual machine per desktop computer, mainly to avoid resource competition between virtual machines. Due to the fact that the opportunistic underlying infrastructure is not capable of type I hypervisors, the virtualization operations request the assistance of type II hypervisors suitable for desktop computer based on x86 architectures. Due to the above, each desktop computer has installed the VMware Workstation type II hypervisor, which assists the virtual machines operations to deploy the opportunistic IaaS model. All hypervisor services are accessed through the VMware platform with VIX libraries.

A. Cloud evaluation

As show in Fig. 4, in order to test the UnaCloud services for Grid users, some grid computing components were used, including a master node that has assigned two CPU cores and 2GB of RAM memory, and 35 slave nodes that have assigned two CPU cores and 1GB of RAM memory. This virtual infrastructure supports the e-Science experimentation of the Department of Biological Sciences, which is developing projects that analyze the coffee, cassava and potatoes genome, to improve production affected by biological organisms [25], [26] and [27].

Figure 5. UnaCloud IaaS located deployment.

As illustrate in Fig. 5, the virtual infrastructure deployment was assisted by UnaCloud following an IaaS located deployment. Grid users deployed the 35 grid slave nodes in about 7 seconds. The average time that each virtual machine took in parallel to load the guest operating system (Debian 4) and to enabling network services (to be accessed via SSH) was approximately 4 minutes, time in which a slave is ready to receive jobs from its cluster master.

As show in Fig. 4, in order to test the UnaCloud services for IaaS Users, some IaaS virtual machines were used, including software development and data mining, customized execution environments. This virtual infrastructure supports the academic activities of students of the Department of Systems and Computing Engineering. As illustrate in Fig. 6, the virtual infrastructure deployment was assisted by UnaCloud following a non-located IaaS deployment. IaaS users deployed 70 virtual machines in about 13 seconds. The average time that each virtual machine took in parallel to load the guest operating system (Windows XP) and to enabling network services (to be accessed via Remote Desktop) was approximately 5 minutes.

Figure 6. UnaCloud IaaS non-located deployment.

Both case studies demonstrate how UnaCloud provides an opportunistic IaaS model and validates all of its services. The validation process shows that UnaCloud incorporates relevant features in the cloud computing context. These features are summarized in Table 1. As mentioned before, concepts like SLA or QoS are not part of the UnaCloud initial scope.

TABLE I. UNACLOUD CLOUD COMPUTING FEATURES

Feature	*Description*
Usability	UnaCloud provides Web interfaces, whose operation is almost intuitive, requiring basic IT knowledge.
Self-service	UnaCloud users can unilaterally consume basic computing resources on a self-service model.
Broad network access	UnaCloud provides basic computing services that are available over the network and are consumed through standard secure remote access mechanisms.
On-demand services customization	UnaCloud provides services to customize execution environments required on demand by the end-user. This customization is able to meet large scale computational requirements.
Time-sharing hardware	UnaCloud incorporates an opportunistic strategy that allows the use of idle computing resources in a non-intrusive manner. This strategy allows the simultaneous opportunistic use of the underlying infrastructure by multiple users.
Virtualization	UnaCloud uses a virtualization strategy to allow the on-demand deployment and assign of customized execution environments.
Scalability	UnaCloud uses an opportunistic commodity horizontal scaling infrastructure and is based on a private cloud deployment model.
Interoperability and loose coupling	UnaCloud is based in loose coupling and interoperability services that can operate over highly heterogeneous, distributed and non-dedicated infrastructures.
Extensibility	UnaCloud is based in open source tools, broadly diffused in order to facilitate its extensibility.
Delegated administration	UnaCloud hides the underlying infrastructure complexity to end-users and provides services to support common administration tasks.
Security	UnaCloud uses authentication, authorization, confidentiality and non-repudiation mechanisms to secure the IaaS model deployment.
Measured service	UnaCloud records and reports the IaaS model traceability at user level.

B. Performance degradation perceived by the owner user

In order to analyze the performance impact perceived by resource owners due to the simultaneous execution of the virtual machine as a background and low priority process, three tests were performed. In the first tests we evaluated the performance when a virtual machine executes intensive processing applications. To achieve this, the execution of a CPU intensive application was performed by the resource owner, using three different environments: without executing the virtual machine in background and executing the virtual machine (making intensive use of processing) having one core and two cores assigned, respectively; the results of the tests are shown in Table 2. The results show that the execution of the processing virtual machine in background affected the performance perceived by resource owners by less than 1%.

TABLE II. CPU PERFORMANCE IMPACT

Environment/Test	**Task Completion Time (seconds)**			
	Test 1	***Test 2***	***Test 3***	***Test 4***
Without VM	53,94	81,01	108,05	134,99
With a VM (1 Core)	54,16	81,42	108,39	135,58
With a VM (2 Cores)	54,21	81,46	108,58	135,60

In the second set of tests, the performance impact on the resource owners was evaluated when they execute storage intensive applications (I/O). To achieve this, file compression operations of different sizes were executed by resource owners. These tests were executed within the same environments as the first tests and the results are shown in Table 3. The results evidence a low impact, less than 3%, in the performance perceived by resource owners. It is justified in the operating systems default mechanisms to manage the local processes priority. These mechanisms ensure computing resources to higher priority processes, while dynamically reducing the computing resources allocated to lower priority processes. The third tests confirm this fact.

TABLE III. I/O PERFORMANCE IMPACT

Environment/Test	**Task Completion Time (seconds)**			
	Test 1 200 MB	***Test 2 500 MB***	***Test 3 1 GB***	***Test 4 2 GB***
Without VM	104,10	259,85	521,16	1041,42
With a VM (1 Core)	105,66	262,43	526,63	1060,75
With a VM (2 Cores)	106,02	263,03	527,06	1063,07

A third set of tests were executed in order to monitoring the processor usage from both the resource owner processes, and the background and low priority virtual machine processes, which had two CPU cores assigned. Intensive processing tasks were executed within both environments. The results are shown in Fig. 7.

In the test, after measuring CPU usage with no virtual machine running, we initiate (time 3) a virtual machine using nearly 50% of the CPU, and at 5, we increase its computational requirements to nearly 100%. We then modify the CPU need from the resource owner. Between 7 and 8 the resource owner demands a 50% of the CPU and the virtual machine load automatically decreases to 50%. Between 9 and 10 the resource owner increases the consumption to about 100% and the virtual machine automatically reduces their consumption to a minimum. Between 11 and 12 the resource owner goes back to a 50% demand, and after 12, the resource owner leaves the physical machine.

The results show that the virtual machine only consumes idle processor cycles, or all cycles in the case of a fully available resource (not temporarily used or a dedicated resource). This fact guarantees a very low impact on the performance perceived by the resource owners. Based on the tests results, we conclude that virtualization and opportunistic strategies incorporated in UnaCloud represent a non-intrusive solution for deploying large scale virtual

 ISBN: 978-1-61208-153-3

infrastructures, encouraging the use of idle computing resources and providing an efficient solution to preexisting resources underutilization problem.

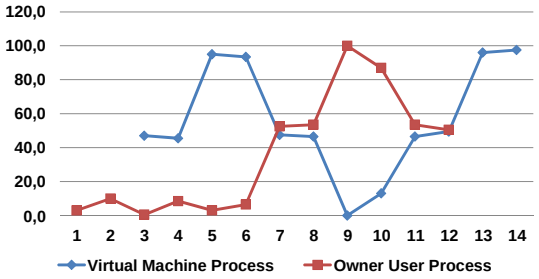


Figure 7. CPU usage for virtual machine and resource owner.

VI. CONCLUSIONS AND FUTURE WORK

UnaCloud is an opportunistic cloud computing IaaS model implementation, which provides at lower cost, basic computing resources (processing, storage and networking) to run arbitrary software, which include operating systems and applications. UnaCloud deals with the problems associated to the use of commodity, non-dedicated, distributed, and heterogeneous computing resources that are part of different administrative domains. To achieve this, we proposed an IaaS architecture based on two main strategies: a virtualization strategy to allow the on-demand deployment of customized execution environments and, an opportunistic strategy based on the validation and extension of DGVCS's design concepts to provide a commodity, non-dedicated, and heterogeneous underlying infrastructure.

Our IaaS architecture supposed a convergence between cloud computing paradigm and DGVCS's. The results not only demonstrate the convergence viability, but also offer promising opportunities to meet customized computational requirements thought the use of open source, low cost, extensible, interoperable, efficient, scalable and opportunistic IaaS model. In addition, UnaCloud represents an economically attractive solution for constructing and deploying large scale computing infrastructures, avoiding not only, underutilization of non-dedicated computational resources, but also financial investments in hardware and costs associated with physical space, temperature-controlled environment, maintenance process, etc.

UnaCloud cloud computing features are promising to reduce the development cycle and the generation of results time of any activity or project depending on the agile provision of computing resources, including academic, scientific and even commercial initiatives.

New challenges will have to be faced in order to improve the IaaS model offered: a requirement is to analyze how to guarantee statistic QoS, improving the best-effort scheme currently in use. Future work also includes UnaCloud extension to provide networking on-demand customization, creation of an API that allows that new services or applications can be incorporated to UnaCloud, compatibility with other type II hypervisors, PaaS and SaaS service models, and public, community and hybrid cloud computing deployment models. We also are preparing the UnaCloud solution as an Open Source project that will be released on 2012.

REFERENCES

[1] R. Buyya, Y. Shin, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," Proc. 10th IEEE International Conference on High Performance Computing and Communications, IEEE Press, 2008, pp. 5-13, doi:10.1109/HPCC.2008.172.

[2] Carl Kesselman and Ian Foster, The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, Nov. 1998.

[3] P. Mell and T. Grance, "NIST Definition of Cloud Computing". National Institute of Standards and Technology (NIST), 2009.

[4] M. Vouk, "Cloud computing - Issues, research and implementations," Proc. 30th International Conference on Information Technology Interfaces, IEEE Press, 2008, pp. 31-40, doi:10.1109/ITI.2008.4588381.

[5] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," Lecture Notes in Computer Science, vol. 5931, 2009, pp. 24-44, doi:10.1007/978-3-642-10665-1_4.

[6] Google Trends Labs, "Cloud Computing", [Online], http://www.google.com/trends?q=cloud+computing&ctab=0

[7] V. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, "Applying Software Engineering Principles for Designing Cloud@Home," Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), May 2010, pp. 618-624, doi:10.1109/CCGRID.2010.76.

[8] Amazon Web Services, LLC, "Amazon Elastic Compute Cloud (Amazon EC2)", [Online], http://aws.amazon.com/, Aug. 10, 2011.

[9] Amazon Web Services, LLC, "Amazon Elastic Compute Cloud (Amazon EC2) ", [Online], http://aws.amazon.com/ec2, Aug. 10, 2011.

[10] Amazon Web Services, LLC, "Amazon Simple Storage Service (Amazon S3) ", [Online], http://aws.amazon.com/s3, Aug. 10, 2011.

[11] OpenNebula.org, [Online], http://opennebula.org, Aug. 10, 2011.

[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," Proc. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), IEEE Press, May 2009, pp. 124-131, doi:10.1109/CCGRID.2009.93.

[13] Nimbus, [Online], http://workspace.globus.org/, Aug. 10, 2011.

[14] J. Shoch and J. Hupp, "The "worm" programs-early experience with a distributed computation," Communications of the ACM, vol. 25, March 1982, pp. 172-180, doi: 10.1145/358453.358455.

[15] M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," Proc. 8th International Conference of Distributed Computing Systems, IEEE Press, June 1988, pp. 104-111, doi:10.1109/DCS.1988.12507.

[16] Mersenne Research, Inc, "GIMPS: Great Internet Mersenne Prime Search", [Online], http://www.mersenne.org, Aug. 10, 2011.

[17] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home An Experiment in Public-Resource Computing," Communications of the ACM, vol. 45, Nov. 2002, pp. 56-61, doi:10.1145/581571.581573.

[18] Distributed.Net, [Online], http://www.distributed.net, Aug. 10, 2011.

[19] D. Anderson, "BOINC: A System for Public-Resource Computing and Storage," Proc. 5th IEEE/ACM International Workshop on Grid, IEEE Press, Nov. 2004, pp. 4-10, doi:10.1109/GRID.2004.14.

[20] L. Sarmenta, S. Chua, P. Echevarria, J. Mendoza, R. Santos, S. Tan, and R. Lozada, "Bayanihan Computing .NET: Grid Computing with XML Web Services," Proc. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Press, 2002. pp. 434-435, doi:10.1109/CCGRID.2002.1017182.

 ISBN: 978-1-61208-153-3

[21] B. Francisco and M. Rodrigo, “The OurGrid Approach for Opportunistic Grid Computing,” Proc. First EELA-2 Conference, Feb. 2009, pp. 11-19.

[22] A. Goldchlegery, F. Kon, A. Goldman, M. Finger, and G. Bezerra, "InteGrade: object-oriented Grid middleware leveraging the idle computing power of desktop machines," Concurrency and Computation: Practice and Experience, vol. 16, 2004, pp. 449-459, doi:10.1002/cpe.824.

[23] H. Castro, E. Rosales, M. Villamizar, and A. Miller, "UnaGrid - On Demand Opportunistic Desktop Grid," Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Press, June 2010, pp. 661-666, doi:10.1109/CCGRID.2010.79.

[24] A. Chandra and J. Weissman, "Nebulas: Using Distributed Voluntary Resources to Build Clouds," Proc. 9th Workshop on Hot Topics in Cloud Computing (HotCloud09), USENIX, June 2009.

[25] A. González, H. Castro, M. Villamizar, N. Cuervo, G. Lozano, S. Orduz, and S. Restrepo, "Mesoscale Modeling of the Bacillus thuringiensis Sporulation Network Based on Stochastic Kinetics and Its Application for in Silico Scale-down", Proc. HIBI '09. International Workshop on High Performance Computational Systems Biology, IEEE Press, Oct. 2009, pp. 3-12, doi=10.1109/HiBi.2009.17.

[26] S. Restrepo et al., “Computational Biology in Colombia,” PLOS Computational Biology, vol. 5, Oct. 2009, doi:10.1371/journal.pcbi.1000535.

[27] A. Vargas et al., “Characterization of Phytophthora infestans Populations in Colombia: First Report of the A2 Mating Type,” Phytopathology, Sep. 2009, pp. 82-88, doi:10.1094/PHYTO-99-1-0082.

 ISBN: 978-1-61208-153-3

Making VM Consolidation More Energy-efficient by Postcopy Live Migration

Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi
National Institute of Advanced Industrial Science and Technology (AIST)
Central 2, Umezono 1-1-1, Tsukuba, Japan 305-8568
Email: t.hirofuchi@aist.go.jp, hide-nakada@aist.go.jp, satoshi.itoh@aist.go.jp, s.sekiguchi@aist.go.jp

***Abstract*—Dynamic consolidation of virtual machines (VMs) is a promising technology for reducing energy consumption of data centers. Existing studies on VM consolidation, however, are based on *precopy live migration*; it is difficult to optimize VM locations aggressively due to its long and undeterminable migration process. In this paper, we propose an energy-efficient VM consolidation system exploiting *postcopy live migration*, which always allows quick live migration for any VMs. The consolidation system can optimize VM locations and server power states more frequently than those of using precopy live migration. In our previous work, we implemented postcopy live migration for KVM, and in this paper, we developed the prototype of our consolidation system, where excessive hardware nodes were suspended by means of ACPI S3 and all power usages were monitored with watt meters. Our experiments showed that our consolidation system with postcopy live migration eliminated more excessive power consumption than that of using precopy live migration. Postcopy live migration allowed the prototype system to eliminate 11.8% energy overheads of actively-running VMs, which was improved by approximately 50% from precopy live migration.**

***Keywords*-Virtual Machine; Live Migration; Consolidation; Data Center; Energy Saving.**

I. INTRODUCTION

Dynamic consolidation of virtual machines (VMs) is a promising technology for reducing energy consumption of data centers. The number of power-on server nodes is kept to a minimum at any time, so that the excessive power used for running idle server nodes can be eliminated. The locations of VMs are continuously reoptimized in response to resource requirements of VMs. When there are many idle VMs, a management system consolidates them onto fewer server nodes, and temporarily shuts down the rest of the server nodes. When these idle VMs become active, the system wakes up power-off server nodes, and relocates VMs onto them.

Live migration of VMs greatly contributes to realizing dynamic consolidation. A VM is relocated onto a new server node without any visible disruption. It should be noted that power consumption incurred by live migration itself is a relatively small value, compared to power saving gains by consolidation. As discussed in Section II, in our experiment environment, making an idle server to the suspend state of ACPI reduces 40W and more, and the network traffic and CPU overhead of a live migration consumes approximately only 7W. This means that a management system is required to perform live migrations as many times as possible in order to maximize the energy-efficiency of VM consolidation.

Widely-used live migration mechanisms, however, are not suitable for dynamic consolidation, which cannot relocate VMs frequently due to their long migration duration. These live migration mechanisms are known as *precopy* live migration; all states of a VM are completely copied to a destination node before the execution host is switched to the destination. Until the whole migration process is completed, the VM is still running on a source node. Updated memory pages during previous page copies are repeatedly transferred to the destination. This iteration process results in a long and undeterminable migration time for actively-running VMs.

On the other hand, there are also *postcopy* live migration mechanisms, performing memory page copies after the execution host is switched. This migration does not need iterative memory copies. A migrating VM updates memory pages at a destination node, not at a source node, which do not generate additional data to be transferred. The total amount of transferred data is smaller than precopy; the whole live migration process is shorter and determinable.

We believe postcopy live migration enables more energy-efficient VM consolidation than precopy live migration. To the best of our knowledge, however, all existing studies on VM consolidation are based on precopy live migration. There are open questions regarding how postcopy live migration contributes to power savings of data centers.

In this paper, we propose an energy-efficient VM consolidation system exploiting postcopy live migration. Postcopy live migration enables the consolidation system to aggressively control VM locations and server power states. The proposed system achieves more frequent live migrations and server power state changes. This fine-grained optimization allows the system to eliminate excessive energy consumption much more than using precopy live migration.

The contribution of this paper is clear; this study is the first work that applies postcopy live migration to an energy-saving VM consolidation system. Although postcopy migration techniques themselves have been discussed in research papers ([1], [2]), these implementations have not been seen in publicly-available VMMs. We therefore developed a postcopy live migration mechanism [3] for KVM [4]. In our previous work [5], we discussed the advantages of our postcopy live migration from the viewpoint of performance

 ISBN: 978-1-61208-153-3

assurance for VM consolidation. In this paper, we address the remaining questions of how much energy savings are possible with our postcopy live migration. We have developed a consolidation system using the ACPI S3 mode and evaluated the effectiveness of postcopy live migration through various experiments.

Section II explains how VM consolidation systems basically work, and summarizes why postcopy live migration has great advantages for energy savings. Section III presents our VM consolidation system. Section IV discusses its evaluation. Section V describes related work. Finally, Section VI concludes this paper.

II. Background

Energy saving technologies are keys to success in the data center business, which allow service providers to reduce daily running costs. The recent processors technologies, such as Dynamic Voltage and Frequency Scaling (DVFS) and ACPI C State [6], contribute to reducing energy consumption of running server nodes. However, these technologies cannot cut the excessive power usage of other hardware components, such as a power supply unit and a mainboard. A study on a large data center mentioned servers were operating most of the time at between 10 and 50 percent of their maximum utilization levels; however, the energy efficiency of server hardware in these utilization levels is less than half at peak performance [7]. Although recent data center facilities, such as direct current power supply systems, mitigate this issue, the deployment of these technologies requires large modifications to existing server platforms and facilities. This results in high implementation costs in most data centers.

Emerging virtualization technologies allow VM-based server consolidation for data centers. A consolidation system monitors resource usage of VMs and continuously optimizes VM locations. The system packs VMs onto the fewest possible server nodes and powers off unused server nodes. When detecting the overloading of a server node, the system powers up unused server nodes and relocates some VMs onto them. Even though most VMs are operating at lower utilization levels, the utilization levels of power-on server nodes are always kept high by packing all VMs onto them. Ideally, the energy consumption of all server nodes is proportional to the total resource usage of all VMs. VM consolidation allows service providers to eliminate excessive energy consumption that are not used for customers' computations.

Figure 1 illustrates the overview of our consolidation system. Load Monitor collects resource usage data every one second and put it into a database. Relocation Planner periodically calculates optimal locations for VMs from the latest resource usage histories in the database. VM Controller requests live migration to server nodes according to the results from Relocation Planner. Although consolidation systems

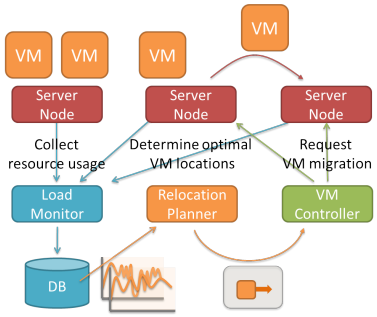


Figure 1. System components of our consolidation system

Table I
Specification of Server Node and Network Switch

Server Node	Dell Optiplex 960 CPU: Intel Core2 Q9400, RAM: DDR3 16GB HDD: ST380815AS Seagate 80GB GbE NIC: Intel 82567LM-3 GbE NIC: Broadcom NetXtreme BCM5721
Network Switch	Planex FXG-24IRM (GbE, 24 port)

have different design details, the above system overview is basically common to most consolidation systems.

Next, we explain the energy consumption breakdown of our VM consolidation system, and then point out why postcopy live migration is suitable for VM consolidation.

A. Energy Consumption Breakdown of VM Consolidation

1) Power Consumption of a Server Node: Before discussing requirements for energy-efficient VM consolidation, we measured power consumption of a server node in our cluster. The specification of the server node is summarized in Table I.

We use a customized Dell Optiplex 960, which supports the ACPI S3 mode and an out-of-band hardware management system (Intel AMT) [8]. Our consolidation system requires a hardware mechanism that allows VM Controller to wake server nodes up via a network. We first tried to use the Wake-On-LAN (WOL) feature, which is widely supported by most network interface cards. However, we found that WOL was not reliable enough to be used in a server cluster. The WOL message is transferred by a UDP datagram, which is likely dropped in congested networks. In practice, if the consolidation system is deployed on a large server cluster, each server node also needs to support more powerful remote hardware management than WOL; the hardware and software settings of all server nodes should be reconfigurable from a remote administrative program. Intel AMT (Active Management Technology), working in the firmware level, allows powerful remote management including power status control, console redirection, and OS installation. Intel AMT exploits TCP connections for its RPCs, making remote control more reliable than other UDP-based remote management mechanisms (e.g., IPMI [9]).

 ISBN: 978-1-61208-153-3

Table II
SERVER ENERGY CONSUMPTION (POWER ON)

CPU Usage (%)	C-State	Watt
100	Enabled	100
100	Disabled	100
0	Enabled	53
0	Disabled	64

Table III
SERVER ENERGY CONSUMPTION (POWER OFF/SUSPENDED)

State	Intel AMT	Watt
Power Off	Enabled	6
Power Off	Disabled	0
Suspended	Enabled	7
Suspended	Disabled	7

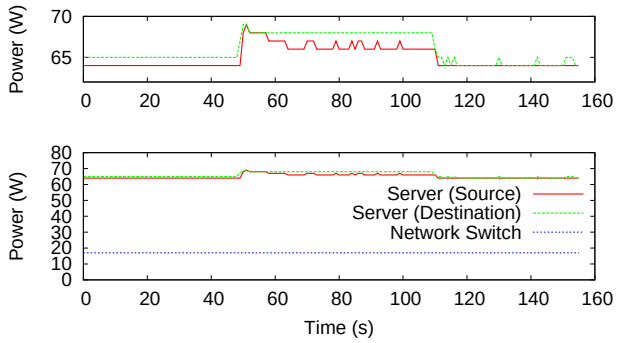


Figure 2. Energy consumption of server nodes and a network switch (A live migration is performed from 45 seconds to 105 seconds. The upper graph shows details around 65W.)

Tables II and III show energy consumption of a server node in its various states and settings. The server node, running at its full CPU utilization, consumes approximately 100W. The idle server node consumes 64W without the power saving feature. The ACPI C State, which enables an idle CPU to stop its clock cycle, contributes to reducing only 11W. Even though the server node is idle, it still consumes approximately half of the power usage at the maximum utilization level. It should be noted, DVFS (i.e., scaling up/down CPU's clock frequency), cannot reduce idle CPU power additionally; the clock cycle is already stopped by the C State feature.

When the server node is suspended, its power consumption is only 7W; this is a much smaller value than an idle power-on state. An interesting finding is that when Intel AMT is enabled the powered-off sever node still consumes 7W. Even though an operating system has been shut down, the firmware OS of Intel AMT is still running. When Intel AMT is disabled, the power consumption is approximately 0W. However, as mentioned before, this feature is required to control server power states remotely.

The results are summarized as follows: First, the contribution of CPU's power saving features is much smaller than making a server node shutdown. Second, because the recent out-of-band management technology, Intel AMT, requires its firmware OS to keep always running, the power consumption in the power-off state is not zero; in our experiments, it is approximately 7W, which is equal to the suspended state.

2) Power Consumption of a Live Migration: Figure 2 shows energy consumption of server nodes and a network switch when a live migration was performed in our experiment environment. An idle VM with 2GB memory was migrated between two server nodes via a GbE network. The normal live migration mechanism of KVM was used. It took approximately 60 seconds to be completed. While the live migration was being performed, the power consumption of each server node increased by 3W or less; this was mainly caused by the CPU overhead of the live migration. Although more than 2GB data was transferred via the network switch, its energy consumption did not show a visible increase. It should be noted that the power consumption of the network switch is nearly invariable while being powered on; the power consumption does not depend on how much data is being transferred now.

In our experiment environment, the additional power consumption incurred by a live migration is approximately 0.08Wh, which is calculated by integrating the power increase during the migration period. This value is much smaller than that of continuing to run a server node; 0.08Wh is corresponding to the power consumption of running an idle server node only in 5 seconds.

B. Requirements for Energy-Efficient VM Consolidation

These results have pointed out design criteria for energy-saving VM consolidation. First, to get the maximum energy saving, a VM consolidation system should exploit the ACPI S3 feature to turn off idle server nodes. The amounts of power consumption at the S3 state and power-off state are the same in our experiment environment. By using the ACPI S3 feature, the consolidation system can turn off/on a server node only in 5 seconds or less. This is much shorter than powering off/on the server node. To power off the server node, it takes approximately 20 seconds after the `shutdown` command is invoked. After the power-on command is invoked via Intel AMT, the VMM on it becomes operational approximately in 60 seconds. These long transitional periods result in increasing excessive power usage, which is not consumed by actual computations of VMs.

Second, the VM consolidation system should repack VMs as aggressively as possible to make excessive server nodes temporarily sleep. As discussed previously, at the viewpoint of power consumption, the overhead of a live migration is far less than that of continuing to run an excessive node; although the power consumption during the transition period of a suspend (e.g., 5 seconds) is also considered, the repacking overhead with one migration and

 ISBN: 978-1-61208-153-3

one node suspend incurs only 0.15Wh (i.e., corresponding to approximately 10 seconds power consumption of an idle node). This means, to get the maximum energy saving, the consolidation system should be designed to be able to optimize VM locations at shorter intervals than one minute. Existing studies concerning VM packing have not addressed this kind of frequent optimization at such short intervals. On the other hand, we aim to establish fine-grained, aggressive optimization at the level of every 10 seconds, not in daily and weekly cycles.

C. Limitation of Precopy Live Migration

Prior studies regarding VM consolidation are based on precopy live migration, which is already available in widely-used VMMs (e.g., Xen [10], KVM, and VMware [11]). We believe, however, precopy live migration is not suitable for energy-efficient VM consolidation, because of its undeterminable (and possibly large) migration time.

It reconstructs a VM's memory image at a destination host **before** switching its execution node ([12], [13], [14]). After live migration is initiated, this basically works as follows.

1: Start dirty page logging at a source host. This mechanism detects updates of memory pages during the following memory copy steps. **2:** Copy all memory pages to the destination. Since the VM is running at the source host, memory pages are being updated during this period. **3:** Copy dirtied memory pages to the destination again. Repeat this step until the number of remaining memory pages is small enough. **4:** Stop the VM at the source. Copy the content of virtual CPU registers, the states of devices, and the rest of the memory pages. **5:** Resume the VM at the destination host.

The problem of precopy live migration is caused by the third step; dirtied pages must be iteratively copied to the destination again and again. If the VM is running a memory-update-intensive workload, numerous dirty pages are created and transferred continuously. The total time of precopy live migration basically becomes much larger than that of *cold migration* (i.e., stop the VM, send its state to a destination, and restart the VM). In the worst case, live migration is never completed; i.e., a workload dirties VM memory faster than network bandwidth can accommodate.

This large migration time prevents a consolidation system to optimize VM locations frequently. It is not possible to maximize energy efficiency of VM consolidation.

III. Energy-Efficient VM Consolidation with Postcopy Live Migration

We propose an energy-efficient VM consolidation system exploiting postcopy live migration. In this section, we explain the advantage of using postcopy live migration, and describe the design and implementation of our VM consolidation system.

A. Postcopy Live Migration

In previous work [3], we developed a postcopy live migration mechanism for KVM. In contrast with precopy migration, memory pages are transferred **after** a VM is resumed at a destination host. The key to postcopy migration is an on-demand memory transfer mechanism, which traps the first access to a memory page at the destination and copies its content from a source host. Postcopy migration basically works as follows:

1: Stop the VM at the source host. Copy the content of virtual CPU registers and the states of devices to the destination. **2:** Resume the VM at the destination without any memory content. **3:** If the VM touches a not-yet-transferred memory page, stop the VM temporarily. Copy the content of the memory page from the source. Then, resume the VM.

The third step is repeated until all memory pages are transferred to the destination. In addition, in parallel with the on-demand page retrievals, a background copy mechanism works to make bulk copies of not-yet-transferred pages. Because on-demand page copy may not cover all ranges of VM memory in a short period of time, the background copy mechanism gets rid of dependency on a source host as soon as possible. The background copy mechanism analyzes important memory areas with page fault statistics, and starts to deal with hot-spot memory pages for current VM workloads. On-demand memory page retrievals over a network are reduced by this mechanism. These mechanisms are transparent to the users of the VM. Our postcopy live migration mechanism supports any guest operating systems without any modifications to them.

A postcopy live migration is always completed in $Ramsize/Bandwidth$ seconds, which is much shorter than precopy. On the other hand, a precopy live migration requires $Ramsize/Bandwidth + \alpha$ seconds to be completed. α depends on the memory update speed of the guest operating system; if a VM intensively updates memory or a network is congested, α becomes larger, and in the worst case the migration is never completed.

The possible downside of postcopy migration is the risk to failure of VMs. A migrating VM depends on not-yet-transferred memory pages on its source host. If the source host is unexpectedly terminated during the migration, the VM cannot continue running anymore. However, considering that IaaS data centers do not assure 100% reliability of their services, we believe that this trivial downside does not adversely affect the feasibility of postcopy migration. As explained in the later sections, postcopy migration greatly improves energy efficiency of dynamic consolidation, which results in great benefits for service providers.

B. VM Consolidation System

Figure 3 shows the design overview of our VM consolidation system. Broadly speaking, there are 3 types of physical

 ISBN: 978-1-61208-153-3

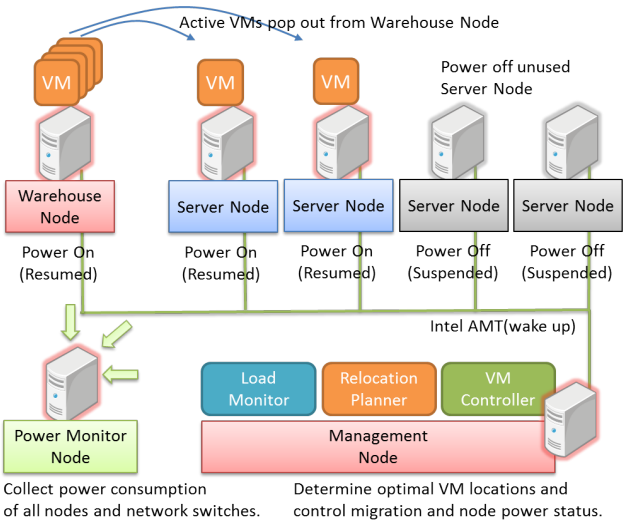


Figure 3. Design overview of our consolidation system

Figure 4. Power Measuring System

nodes in our server cluster. **Management Node** periodically determines optimal VM locations and controls migration and node power status, where the aforementioned software components are running (See also SectionII). **Power Monitor Node** collects power consumption of all nodes and network switches. **Host Nodes** (i.e., Warehouse/Server Nodes) launch VMs and execute live migrations of VMs.

1) Management Node: Load Monitor receives resource usage statistics from each host node, such as CPU usage, network I/O, and disk I/O of both a host node and the VMs running on it. This information is retrieved from `/proc/` of the host Linux operating system and the monitor interface of KVM. All the collected statistics are stored in an SQLite database. In order to support hundreds of host nodes, the latest statistics are temporarily cached in the memory of Load Monitor, thereby reducing database requests.

Relocation Planner retrieves resource usage histories from the database, determines whether a host node is overloaded or not, and calculates a relocation plan. We carefully designed this component to be independent from the others, so that it is possible to implement various consolidation algorithms.

VM Controller executes live migration according to the relocation plan. We use XML-RPC to control VMs on host nodes remotely; three request messages (e.g., `CREATE_VM`, `MIGRATE_VM`, and `DESTROY_VM`) are defined to create, migrate, and destroy the requested VM. On each host node, there is a server daemon handling these XML-RPC requests. VM Controller also executes the suspend/resume of host nodes. When all VMs on a host node are removed from it, VM Controller requests the host operating system of the node to invoke the `pm-suspend` command. When a suspended host node is required to run a VM, VM Controller requests the firmware of the host node to wake it up via Intel AMT.

2) Power Monitor Node: We developed a power measuring system of our server cluster, which periodically collects power consumption of host nodes and network switches individually. The current, voltage, and active power of a target component are measured by a customized watt meter; we use Watt Checker (MWC-01) of Osaki Electric Co, Ltd. The accuracy of active power is $\pm 2\%$. The measurement interval of the watt meter is one second. All watt meters are connected to a monitoring server (i.e., Power Monitor Node) via USB interfaces. It is possible to measure power consumption of 120 target components. Figure 4 is a photo of a part of our power measuring system; a 2U rackmount measuring board for 8 target components is installed into a 19-inch rack.

3) Host Nodes (Warehouse and Server Nodes): To consolidate VMs efficiently, our consolidation system introduces two types of host nodes, Server Nodes and Warehouse Nodes. Actively-running VMs are assigned onto Server Nodes, and idle VMs are packed into Warehouse Nodes. If a VM running at a Server Node becomes idle (i.e., consuming few CPU resources), the system migrates the VM to a Warehouse Node, and suspends the Server Node if there are no VMs anymore.

This design choice is made by considering hardware costs and use cases. To pack idle VMs into the minimum host nodes, the system should have a special host server with a large amount of physical memory; a Warehouse Node is dedicated to hosting as many idle VMs as possible. On the other hand, Server Nodes have a small amount of memory to host a few active VMs. Because active VMs will make substantial impacts on CPU and I/O resources, these VMs should be located on other nodes than Warehouse Nodes.

C. Packing Algorithm

Our consolidation system is designed to be independent of packing algorithms. It is possible to implement any kinds

 ISBN: 978-1-61208-153-3

of packing algorithms. In the first prototype system, we implemented a simple heuristic algorithm that determines near-optimal locations swiftly. An active VM is exclusively assigned to a Server Node; on the other hand, idle VMs share a Warehouse Node.

First, all the VMs are launched at one of the Warehouse Nodes, and then the following steps are iterated every second.

Distribution Phase: When the latest 10-seconds CPU load average of a Warehouse Node reaches 90% (i.e., is regarded as overloaded), the most CPU-consuming VM is migrated to a Server Node. By using usage statistics measured in outside of VMs, it is difficult to accurately determine the amount of a CPU resource is actually required. Therefore, simply, we pick up the VM that is probably in a 'race-to-halt' state. A target Server Node is selected from sleeping Server Nodes, and then resumed to accept the VM. The VM is migrated to the Server Node. Finally, if there are no VMs on the Warehouse Node, the consolidation system suspends it.

Consolidation Phase: The system does not move the migrated VM for at least 20 seconds after the migration ends, in order to avoid overreaction. After that, the resource monitoring daemon of the VM is started to periodically check whether the latest CPU load average of the VM is under a return threshold value (50%). If the load average is under the threshold, the monitoring daemon tries to move the VM back to one of the Warehouse Nodes; it tries to find the Warehouse Node that has sufficient CPU and memory resources for the VM. If the Warehouse Node is suspended, the consolidation system resumes it. An *admission ticket* to a Warehouse Node is given to the VM on a 'first come, first served' basis, in order to serialize migrations to the Warehouse Node. If a Warehouse Node with sufficient resources is found, the VM is migrated to it. Otherwise, the VM remains at the Server Node; the daemon pauses at one second intervals and tries the above steps again.

It should be noted that the algorithm is currently based on only CPU usage statistics, not including disk and network I/O data. At the time this paper is being written, KVM does not support live migration for paravirtualized devices, such as VirtIO Block Device and VirtIO Network Device. All the VMs on our consolidation system must use fully-virtualized devices incurring relatively high CPU overheads.

IV. Evaluation

In our testbed cluster, we performed experiments to evaluate the effectiveness of our consolidation system; our consolidation system with postcopy live migration was compared with that of using precopy live migration. We measured energy consumption of our consolidation system with simple and complex workload scenarios.

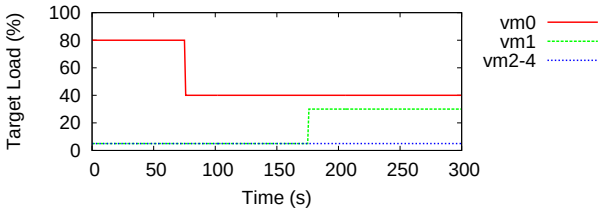


Figure 5. The CPU Load Changes of VMs in a Simple Consolidation Scenario

A. Experiment Settings

Our testbed cluster includes 6 host nodes of the specification in Table I; one node is used for a Warehouse Node, and other 5 nodes are used for Server Nodes. Each host node is connected to a shared disk server, which is required to perform live migrations among different host nodes. Additionally, as mentioned in Section III-B, a Management Node controls VM consolidation, and a Power Monitor Node collects power consumption. These nodes are connected to a private network segment. The host nodes are also connected to a migration network segment, which is intended to isolate busty migration traffic from other management traffic. In our experiments, our consolidation system controls 5 VMs; each VM has one virtual CPU core and 1 GB RAM.

We developed a workload generator program running on a guest operating system. The packing system consolidates VMs in response to their CPU loads. Live migrations are deeply affected by their memory update speeds. To identify characteristics of our consolidation system, therefore, the program can generate any specified CPU loads and memory update intensities by interlacing short busy loops and sleeps. It is designed to emulate a server-type workload like web/mail applications. A small computational task is periodically generated at a calculated average rate conforming to the Poisson distribution; as is well known, the arrival rate of a new request to a network application is basically explained by the Poisson distribution.

B. Simple Scenario

First, we evaluate the basic effectiveness of using postcopy live migration for dynamic consolidation. In this evaluation, we used a simple load change scenario as shown in Figure 5. The load of VM0 is first set to 80%, and then reset to 40% at 75 seconds. The loads of other VMs are first set to 0.05%, and then the load of VM1 is reset to 30% at 175 seconds. The memory update intensity of workloads is set to 0.6; with this value, the memory update speed at a 100% CPU usage is approximately 200MB/s.

Figure 6 shows the CPU usage of host nodes and VMs. Figure 7 shows the power consumption of host nodes and network switches.

The left side of the figures shows the case of using postcopy live migration. At 85 seconds, the consolidation

 ISBN: 978-1-61208-153-3

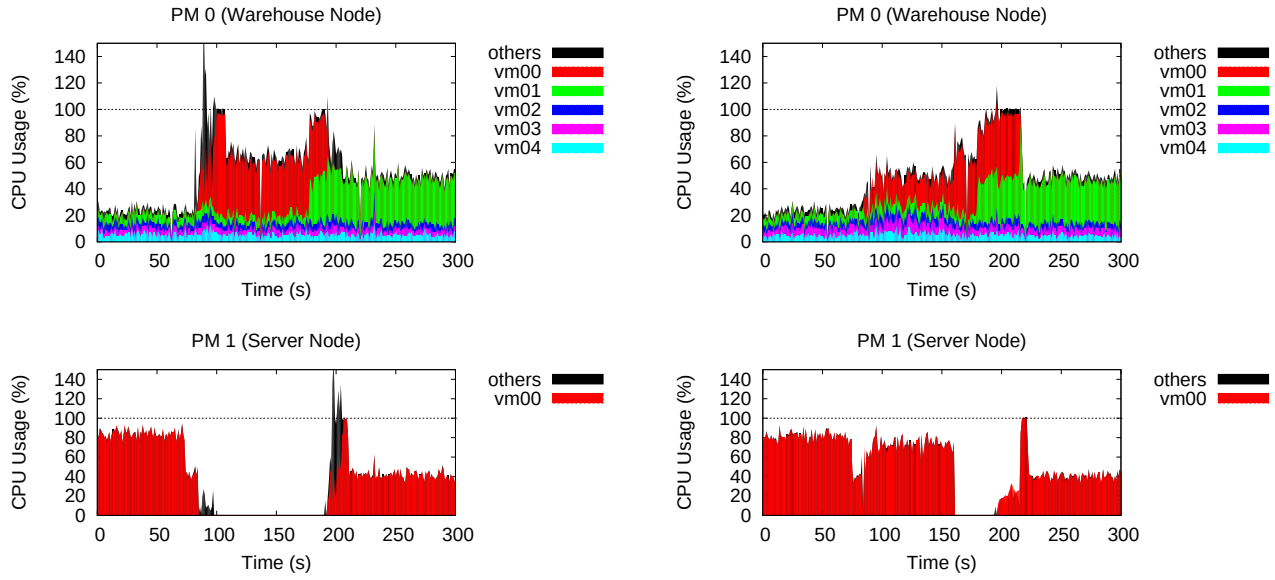


Figure 6. The CPU usage of Host Nodes and VMs (left: using postcopy, right: using precopy)

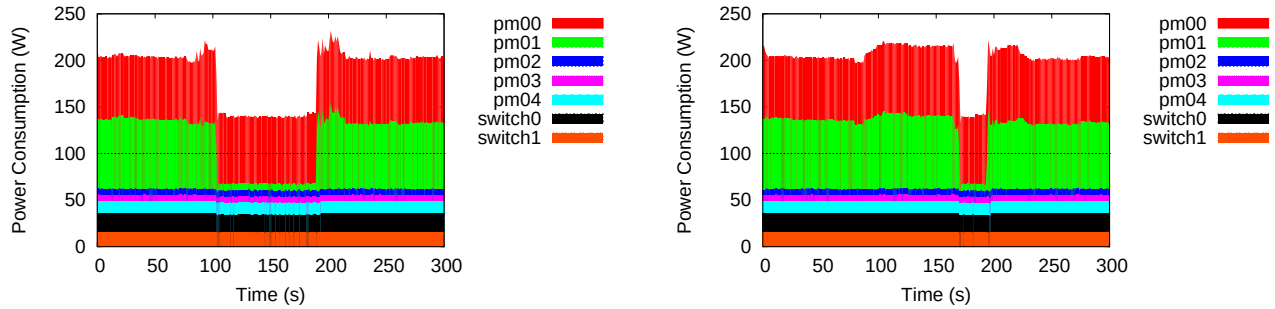


Figure 7. The Power Consumption of Host Nodes and Network Switches (left: using postcopy, right: using precopy)

system decided to consolidate all VMs into Warehouse Node (PM0), and then started to relocate VM0 to it. This live migration finished approximately at 100 seconds, and then Server Node (PM1) was suspended. As shown in Figure 7, the total power consumption was reduced by approximately 60W. It should be noted that the live migration incurred energy overheads (i.e., 20W or less) until completed; however, the overheads were far less than the power consumption saved by this dynamic consolidation. At 175 seconds, VM1 started consuming 40% CPU usage. After detecting the overloading of Warehouse Node (PM0), the consolidation system resumed Server Node (PM1) again, and relocated the most CPU-consuming VM (VM0) to it.

As shown in the right side of the figures, the consolidation system with precopy live migration started to relocate VM0 to Warehouse Node (PM0) at the same time as using postcopy (i.e., at 85 seconds). In this case, however, the live migration did not finish until 160 seconds. The memory update speed of VM0 was over 80 MB/s during the migration, which was relatively close to the available bandwidth (approximately 120 MB/s) of the migration network. Because precopy live migration needs to transfer updated memory pages repeatedly, the consolidation system could not promptly relocate VM0 to Warehouse Node (PM0) as performed with postcopy live migration. Resultingly, Server Node (PM1) was suspended only in 20 seconds (i.e., approximately 25% of using postcopy). In addition, energy overheads of the live migration were higher than using postcopy. A 20W increase of power consumption continued until the migration was completed, which was involved by dirty page tracking of the migrating VM.

Through these experiments, we confirmed that our consolidation system with postcopy live migration successfully worked, which dynamically optimized VM locations and server power states. In comparison with precopy live migration, postcopy live migration allowed the consolidation system to eliminate excessive power usage more aggressively for memory-intensive VMs.

 ISBN: 978-1-61208-153-3

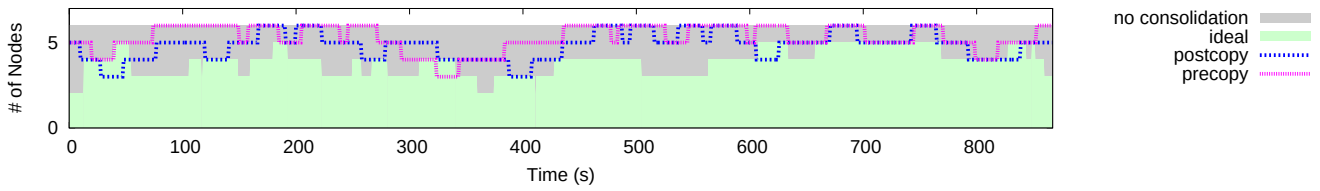


Figure 8. The Number of Active Host Nodes

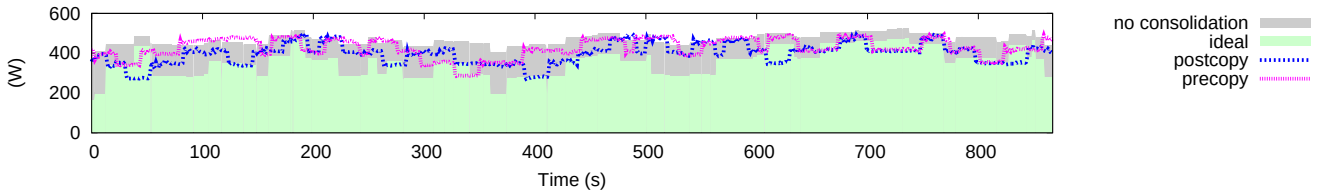


Figure 9. The Total Power Consumption of Host Nodes and Network Switches

C. Complex Scenario

Next, we evaluated our consolidation system with a compound load change scenario. We randomly generated an approximately 15-minutes scenario with the following rules, considering *race-to-halt*-like workloads. A workload on each VM changes its state between active and idle modes at 75% and 25% probabilities, respectively. A new mode continues for a random duration between 30 and 60 seconds. The workload generates a random CPU load between 70% and 100% in the active mode, and between 0% and 30% in the idle mode. The memory update intensity of workloads is set to 0.6, the same value as the previous experiments.

Figure 8 shows the number of active host nodes. **ideal** shows the theoretical number of host nodes required to pack all VMs at each time step, which is calculated from the load change scenario by using First Fit Algorithm; this number assumes that all migrations finish instantaneously at any time.

The consolidation system with postcopy live migration basically used fewer active host nodes than that of using precopy live migration. In the case of using precopy, a live migration sometimes prevented other following migrations from being started for a long time; VM locations were not sometimes optimized in response to load changes.

As shown in Figure 9, the consolidation system with postcopy live migration more closely fits to the ideal total power consumption, which is estimated on the assumption that the power consumption of a host node is proportional to the total CPU loads generated by VM workloads on it [1].

Table IV summarizes the total power consumption accumulated during the load change scenario. The power consumption was reduced by 11.8% with postcopy live migration, and by 5.2% with precopy live migration. It should be noted that the consolidation system addresses energy consumption overheads between the ideal case and no consolidation case; the consolidation system with postcopy live migration eliminated approximately half of the energy overheads, which is improved by approximately 50% from that of using precopy live migration.

[1]From Table III, the power consumption of a host node is roughly estimated to be at $53 + (100 - 53) * L$, where L is the total CPU loads generated by VM workloads on it.

Table IV
ACCUMULATED ENERGY CONSUMPTION

	Energy (Ws)	Saved Energy (%)
no consolidation	390175	-
ideal	294033	24.6
postcopy	344204	11.8
precopy	369877	5.2

V. RELATED WORK

A. Postcopy Live Migration

SnowFlock [1] provides a VM cloning system enabling developers to easily program distributed systems. A postcopy technique is used to rapidly copy the state of a master VM to worker VMs. It is required to modify the memory management code of the Xen's hypervisor and the paravirtualized Linux system. A study [2] developed a postcopy live migration mechanism for the paravirtualization mode of Xen, which extends the swap-in/out code of the Linux kernel for on-demand memory transfer. A special device driver is required to be installed into the guest Linux system.

As described in our previous work [3], we have developed a postcopy live migration mechanism for KVM. In comparison with the above work, our mechanism supports guest operating systems without any modifications to them (i.e, no special device drivers and programs are needed in VMs); all guest operating systems including Windows, Linux, and *BSD are supported. It is implemented as a lightweight

extension to KVM. It is not required to modify critical parts of the VMM code. We named our postcopy migration mechanism as Yabusame, and are now preparing to publish its source code under an open source license [15].

B. Dynamic VM Consolidation using Precopy Live Migration

To the best of our knowledge, this paper is the first work exploiting postcopy live migration for energy-efficient VM consolidation. The following studies regarding VM consolidation are based on precopy live migration.

Sandpiper [16] is a consolidation management system that dynamically optimizes VM locations in order to remove host overloading. This study showed that using workload-specific activity data, such as request arrival rates and response time, makes more optimized relocations possible; resource demand of VMs is estimated and predicted by queuing theory and autoregression analysis. In [17], a consolidation system uses a threshold value of resource usage to trigger VM repacking; if the CPU usage of a host exceeds this value, the system reoptimizes VM locations, so that mitigates the risk that application response times (e.g., service level agreement in this study) are adversely affected. The study [18] exploits an anomaly detection technique based a stochastic model, which determines the VMs and hosts subject to significant state changes. This study argues that a threshold-based algorithm incorrectly detects overloading and mistakenly determines a reconfiguration plan. The study [19] discusses the way of finding turning points of resource demands, where reconfiguration of VM locations is advisable. This technique aims to determine whether repacking is required or not with small calculation cost. Entropy [20] is a VM packing management system exploiting constraint programming techniques. It first determines the minimum number of nodes that are necessary to host all VMs, and then computes an optimal order of migrations to minimizing the overall reconfiguration time. The study [21] presents a network-aware migration scheduling algorithm, which tries to minimize the bandwidth usage while holding migration deadlines.

We consider that these techniques are also applicable to our consolidation system. We can extend our current packing algorithm with the above techniques, thereby improving scalability of our consolidation system for large-scale data centers. In future work, we will discuss the advantage of postcopy migration with other packing algorithms. In our previous work [22], we experimentally developed a genetic algorithm that determines near-optimal VM locations quickly. We have a plan to apply this algorithm to our consolidation system.

VI. Conclusion

In this paper, we have proposed an energy-efficient VM consolidation system exploiting postcopy live migration. Postcopy live migration greatly contributes to eliminating excessive power consumption, which allows our consolidation system to aggressively optimize VM locations. In postcopy live migration, the whole migration process finishes much more quickly than precopy live migration. We developed the prototype of our consolidation system, where excessive hardware nodes were suspended by means of ACPI S3 and all power usages were monitored with watt meters. Our experiments showed that our consolidation system with postcopy live migration eliminated more excessive power consumption than that of using precopy live migration. Postcopy live migration allowed the prototype system to eliminate 11.8 % energy overheads of actively-running VMs, which was improved by approximately 50% from precopy live migration.

In future work, we have a plan to integrate our consolidation mechanism into an open source cloud management system such as Eucalyptus [23] and OpenStack [24]. In addition, we are now preparing to apply our consolidation mechanism to a large-scale data center composed of hundreds of physical hosts. Further details will be reported in our upcoming papers.

Acknowledgment

This work was partially supported by KAKENHI (20700038 and 23700048) and JST/CREST ULP.

References

[1] H. A. Lagar-Cavilla, J. A. Whitney, A. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, “SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing,” in *Proceedings of the fourth ACM european conference on Computer systems*, Apr 2009, pp. 1–12.

[2] M. R. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning,” in *Proceedings of the 5th International Conference on Virtual Execution Environments*, Mar 2009, pp. 51–60.

[3] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, “Enabling instantaneous relocation of virtual machines with a lightweight VMM extension,” in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, May 2010, pp. 73–83.

[4] A. Kivity, Y. Kamay, D. Laor, and A. Liguori, “kvm: the Linux virtual machine monitor,” in *Proceedings of the Linux Symposium*, Jul 2007, pp. 225–230.

[5] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, “Reactive consolidation of virtual machines enabled by postcopy live migration,” in *Proceedings of the 5th International Workshop on Virtualization Technologies in Distributed Computing*, Jun 2011, pp. 11–18.

[6] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation, *Advanced Configuration and Power Interface Specification*, Apr. 2010.

 ISBN: 978-1-61208-153-3

[7] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, pp. 33–37, Dec 2007.

[8] K. Cline, L. Grindstaff, S. Grobman, and Y. Rasheed, "Innovating above and beyond standards," *Intel Technology Journal*, vol. 12, no. 04, pp. 255–268, 2008.

[9] Intel Corporation, Hewlett-Packard Company, NEC Corporation, and Dell Computer Corporation, *-IPMI- Intelligent Platform Management Interface Specification Second Generation v2.0*, Feb. 2004.

[10] 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 nineteenth ACM symposium on Operating systems principles*, 2003, pp. 164–177.

[11] J. Sugerman, G. Venkitachalam, and B.-H. Lim, "Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor," in *Proceedings of USENIX Annual Technical Conference*, 2001, pp. 1–14.

[12] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proceedings of USENIX Annual Technical Conference*, 2005, pp. 25–25.

[13] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, 2005, pp. 273–286.

[14] A. Mirkin, A. Kuznetsov, and K. Kolyshkin, "Containers checkpointing and live migration," in *Proceedings of the Linux Symposium 2008*, Jul 2008, pp. 85–92.

[15] AIST Cloud Computing Research, http://grivon.apgrid.org/; Last accessed: June 30, 2011.

[16] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th Symposium on Networked Systems Design and Implementation*, 2007, pp. 229–242.

[17] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, Apr 2006, pp. 373–381.

[18] M. Andreolini, S. Casolari, M. Colajanni, and M. Messori, "Dynamic load management of virtual machines in a cloud architectures," in *Proceedings of the IEEE Conference on Cloud Computing*, Oct 2009.

[19] T. Setzer and A. Stage, "Decision support for virtual machine reassignments in enterprise data centers," in *Proceedings of the 5th IEEE/IFIP International Workshop on Business-driven IT Management*, Apr 2010.

[20] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. L. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of the 5th International Conference on Virtual Execution Environments*, 2009, pp. 41–50.

[21] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009, pp. 9–14.

[22] H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh, "Toward virtual machine packing optimization based on genetic algorithm," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, ser. Lecture Notes in Computer Science, vol. 5518, Jun 2009, pp. 651–654.

[23] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 124–131.

[24] The OpenStack Project, http://www.openstack.org/; Last accessed: June 30, 2011.

 ISBN: 978-1-61208-153-3

Deterministic Execution of Multiprocessor Virtual Machines

Junkang Nong, Qingbo Wu, Yusong Tan
School of Computer, National University of Defense Technology,
Changsha 410073, Hunan, China
e-mail: {njk.jackson,wu.qingbo2008,yusong.tan}@gmail.com

***Abstract*—Deterministic execution offers a lot of benefits for debugging, fault tolerance, security of multiprocessor systems. Most previous work to address this issue either depends on custom hardware or needs to recompile the program. Some others combine the hardware and software technologies. Our goal in this work is to provide deterministic execution and repeatability of arbitrary, unmodified, multiprocessor systems without custom hardware. To this end, we propose a new abstraction of a multiprocessor virtual machine named *Deterministic Concurrency State Machine* (DCSM). With the virtual private memory model, the multiprocessor virtual machine can execute deterministically as a DCSM. With the replay of the DCSM, better debugging methods and intrusion analysis can be obtained to improve the availability and security of the whole system, not only a program. We implemented DCSM on the Kernel-based Virtual Machine (KVM) and the performance cost can be acceptable if some parameters and optimization strategies are chosen correctly based on the preliminary evaluation results.**

Keywords-availability; concurrency; deterministic execution; security

I. INTRODUCTION

Nowadays, cloud computing is accelerating the market for parallel software development. However, the concurrency in multithreaded programs brings the problem of non-determinism. This non-determinism makes a concurrency system produce different outputs, even given the same input. Such weak repeatability may not ensure that a server running in the cloud can rerun to the previous state right before the physical machine crashed. Then, the wrong results may be sent to clients.

Determinism is the foundation of replay, debugging, fault tolerance and auditing. Many intrusion analysis tools assume that the system can enforce determinism even on malicious code designed to evade analysis [1]. The replicated state machine technology [2] is also based on the assumption that the virtual machine can execute deterministically.

To address the issue of non-determinism, some work that depends on custom hardware [4-5] can obtain a good performance. For software-only solutions, some of them need to recompile the program [6, 12]. When referring to the non-determinism of the whole system, previous software-only solutions [7] primarily focus on pure record-and-replay technology, which incurs high overheads and large space costs. Other software-only technologies [15-16] are tailored to specific classes of programs, but they do not notice that the environment of the program can also induce an unexpected bug (e.g., one Mozilla bug cannot be triggered unless another program modifies the same file concurrently with Mozilla [3]).

In order to provide deterministic execution of arbitrary, unmodified, multiprocessor systems without custom hardware support, we propose *Deterministic Concurrency State Machine* (DCSM). The deterministic execution of DCSM is enforced by our modified hypervisor or virtual machine monitor which we call dVMM. This solution can ensure the repeatability of the environment-caused bug, improving the ability of debugging. Given an external input, this DCSM will make a deterministic state transition based on current execution state. Therefore, the record-and-replay technology is used on this DCSM to ensure the repeatability of the execution of a multiprocessor virtual machine.

This paper makes the following contributions. First, we propose the virtual private memory model and relative scheduling algorithm. With this model and algorithm, the multiprocessor virtual machine that encapsulates multithreaded programs can execute deterministically. As a result, the controllability can be obtained. Second, we define the Deterministic Concurrency State Machine. With this DCSM, the record-and-replay technology can be used to improve the repeatability of the whole virtual machine's execution. Meanwhile, it eliminates the large space costs due to recording the interleaving of CPUs.

The outline of this paper is as follows. In Section 2, we define the DCSM, propose the virtual private memory model and describe how the dVMM ensures the deterministic execution with a scheduling algorithm and record-and-replay technology. Section 3 describes some implementation issues. Section 4 provides some evaluation results. Section 5 discusses relevant issues. Section 6 outlines related work and Section 7 concludes.

II. DETERMINISTIC EXECUTION AND REPLAY OF MULTIPROCESSOR VIRTUAL MACHINES

A. The Problem of Non-determinism

Figure 1 [3] shows a concurrency bug in Mozilla. In this figure, if thread 2 writes the variable io_pending after thread 1 writes it, there will be an expected correct execution path. But if thread 1 writes the variable io_pending after thread 2 writes it, the expectation of the program will be violated. By default, thread 1 should initialize the variable before they execute concurrently.

This is a common concurrency bug, which makes contribution to the non-determinism of multithreaded

 ISBN: 978-1-61208-153-3

programs. If a multiprocessor system is intruded, this non-determinism will make the replay of the intrusion more difficult and result in inaccurate intrusion analysis.

Another example is shown in Figure 2, which is mentioned in DMP [4]. The table inside the figure shows the frequency of the outcome of 1000 runs on a Intel Core 2 Duo machine [4].

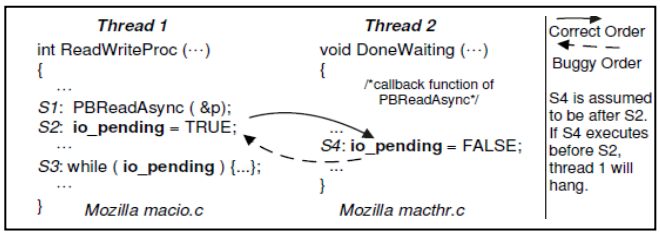


Figure 1. An order violation bug in Mozilla [3].

The result demonstrates that the underlying parallel architecture can affect the result of a system. The Symmetric MultiProcessor (SMP) model is used widely nowadays. But, Figure 2 shows the case of non-determinism in a SMP system. The non-determinism in parallel is caused by the data races of concurrent memory accesses in a SMP system. Our DCSM solution is a software-only approach to solve such a system-level issue. In addition to DCSM, external non-determinism is considered with the record-and-replay technology, which can improve the repeatability of the whole system's execution.

```
int race = -1; // global variable

void thread0() {
  if ( doClearCache )
    clearCache();
  barrier_wait();
  race = 0;
}

void thread1() {
  if ( doClearCache )
    clearCache();
  barrier_wait();
  race = 1;
}
```

clear cache?	thread 0 wins
no	83.2%
yes	33.9%

Figure 2. A simple program with a data race between two threads and runs 1000 times [4].

Next section describes the characteristics of DCSM, while its building methods are described in two following sections. The record-and-replay section specifies the method to deal with external non-determinism outside DSCM.

B. Deterministic Concurrency State Machine

Bocchino Jr. et al. [9] argue that parallel programming must be deterministic by default. But many programs are coded serially. They can reach parallel with the support of other tools such as compiler. Further more, people are used to thinking serially, which will probably result in buggy programs. Therefore, some measures must be taken to make the multithreaded programs execute deterministically. Those measures must also consider the environments influence on the programs. To meet these demands, a deterministic multiprocessor virtual machine is used to encapsulate the multithreaded programs and their environments. This kind of virtual machines ensures deterministic execution of the whole system. And their execution is controlled by dVMM to ensure a deterministic execution path. Such a deterministic multiprocessor virtual machine is called a *Deterministic Concurrency State Machine* (DCSM).

Figure 3 depicts the behaviors of a DCSM. A DCSM can be represented by a tuple {(V, M), I, A}, where V is the set of cpus' states, M is the set of memory states, I is the set of inputs, A is the set of actions. Given the initial state (V0, M0) and certain input, the DCSM will take a subset of actions in A and reach a deterministic state, thus produce a deterministic result. During the actions, DCSM will not receive any external inputs. The actions DCSM takes are the concurrent instructions; the size of these instructions is further determined and controlled to realize the DCSM in the following sections.

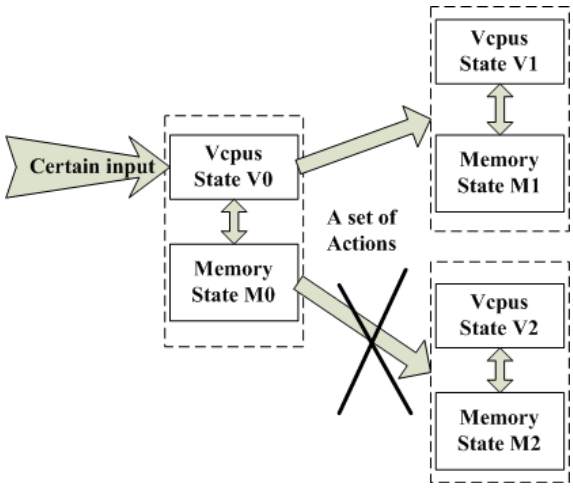


Figure 3. State transitions of a DCSM. The DCSM will deterministically take certain actions to reach state (V1, M1) if given certain input and certain initial state.

C. Virtual Private Memory Model

In a multiprocessor VM, if one virtual cpu (vcpu) is hung up after it acquires a lock, then other running vcpus that want to acquire this lock will waste their time in trying to get the lock. In that case, it is very difficult to control the concurrency for deterministic synchronization because of its complexity. Therefore, the basic scheduling strategy is that all vcpus in the multiprocessor VM must be running concurrently on physical cpus. Otherwise, they must be paused at the same time.

Figure 4 shows the virtual private memory model. There are two main stages-when virtual memory is created and when it is merged or synchronized.

This basic scheduling strategy makes the situation simpler. Based on that strategy, an algorithm can be designed to ensure a multiprocessor VM's deterministic execution. To be deterministic, the concurrent execution must be synchronized at some critical points. So, our algorithm is mainly based on the idea of quantum, which is composed of certain quantity of instructions. These quanta are the actions that DCSM will take to reach the next deterministic state. At

system level, data races are happened on the shared memory. So the virtual private memory model is proposed to provide each vcpu an illusion that each has its own memory. This illusion makes each vcpu executes concurrently without considering the memory interactions. As a result, without the interference from external non-determinism, an isolation of the executions of the quanta is provided before the synchronization stage. So this memory model provides the concurrent stages to fully execute and the synchronization stages to make the result of previous executions deterministic. Algorithms can use these two stages to get fully parallel execution and make the result deterministic when synchronizing.

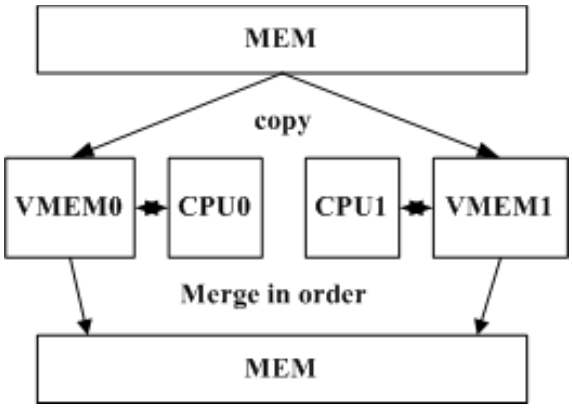


Figure 4. Virtual private memory model.

D. DMP-VPM Algorithm

At the virtualization level, the virtual private memory model ensures the virtual memory isolation of concurrent executions. According to the characteristics of virtualization level, our quantum based algorithm utilizes the virtual private memory model and privatizes the shared memory. And it synchronizes the concurrent quanta to make the result deterministically. This algorithm is called DMP-VPM (Deterministic Shared Memory Multiprocessing based on Virtual Private Memory). Figure 5 tells how a vcpu behaves in DCSM.

Figure 5 describes how the algorithm works. Each vcpu executes after obtaining its virtual private memory. When the quantum finishes, it is time to merge the private memory in order. So for the sake of the sequence guarantee at the merging or synchronization stage, the idea of token ring is utilized. The token is passed in a deterministic sequence among vcpus. A vcpu with the token has the right to merge its private memory and create a new virtual private memory. Otherwise, it must wait for its turn. In the algorithm, when a vcpu with the token wants to merge, it must make sure that the vcpu has not read the pages merged or written by the previous vcpus in this memory version. Otherwise it will create its new virtual memory and re-execute the quantum. After merging successfully, the vcpu with the token will create its new virtual memory, pass the token to the next vcpu and execute its next quantum. Such a deterministic sequence in accessing or modifying memory will result in a deterministic memory state. Note that the design of DCSM does not consider the external non-determinism which will change the execution sequence of a quantum. Under such a condition, vcpus can also reach a deterministic state.

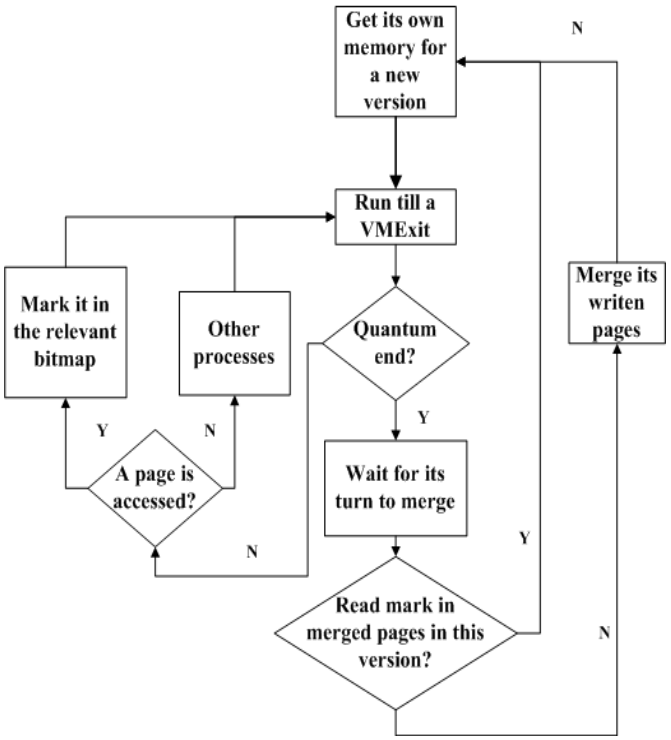


Figure 5. Each vcpu's execution diagram with virtual private memory.

E. The Record and Replay of DCSM

A deterministic executing VM can be regarded as a deterministic state machine. Because given current state and some external input, such a VM can produce a deterministic result, namely make a transition to another deterministic state. Then this VM can be replayed with the initial state and the recorded external inputs. This is the main characteristic of the DCSM. Although the record-and-replay technology has been used in many fields, the record and replay of the DCSM is kind of different.

To replay a DCSM, external non-determinism must be injected during replay at the exactly right time when the injection will not break the execution sequence of a quantum. Which vcpu needs the injection must be recorded. Some information about the non-determinism must also be recorded. When recording the DCSM, the external injections are controlled to happen at the beginning of each quantum, making the replay easier. Such an injecting method further improves the isolation between the quantum and external world and makes sure that the DCSM will not receive any input when taking actions. As a result, the quantum's result is deterministic. But some special instructions like RDTSC must be treated differently. The results of such instructions

 ISBN: 978-1-61208-153-3

and the exact occurrence time must be recorded for the replay.

III. IMPLEMENTATION ISSUES BASED ON KVM

The hardware virtual technology VT, which is developed in some Intel's cpus, makes the implementation easier and will be of great help in performance. KVM is a module in linux and makes the whole operating system a hypervisor based on VT. It utilizes the kernel's memory management mechanism to provide the guest virtual machine with a fake contiguous guest physical memory. To implement the DCSM and its record-and-replay, the interface VMEntry/VMExit is a good place for coding. The implementation framework is shown in Figure 6. According to this figure, some implementation issues are described as follows.

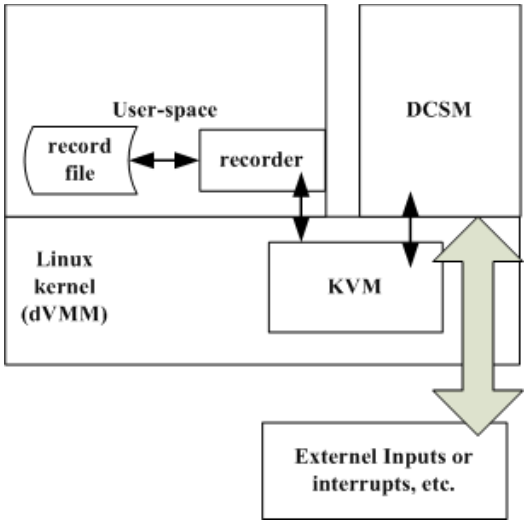


Figure 6. The framework of the dVMM.

A. Implementation of DCSM

When implementing DCSM, we have to take into account the implementation of the virtual private memory model. The implementation of that model will be described along with the DMP-VPM algorithm. The EPT (Extended Page Table) mechanism in intel's VT technology can be used to implement that model. In EPT's page table, each entry has several relevant control bits, namely readable bit, writable bit and executable bit. To provide each vcpu an isolated virtual private memory, we utilize the copy-on-write technology.

In KVM, each EPT violation will cause a VMExit which will call the relevant handler handle_ept_violation(). Then the function kvm_mmu_page_fault() is called to process this violation. In this function, the key memory process function tdp_page_fault() is called. Therefore, the process in this function can be changed to realize our goal. Note that each vcpu is designed to have its own page table. During the creation of the EPT page table, lazy allotment strategy is used. Once the guest accesses a page not present in the EPT page table, we first identify whether it is a write or a read. If it is a read, the EPT page is allotted, set as readable and not writable and the read action on this page is recorded. And if it is a write, a new host physical page is allocated and the original page's data are copied to the new page. Then dVMM will make the relevant EPT page entry redirect to the new page, set corresponding control bits and record this redirection for merging. Again, if it is a write on a page that has the EPT page present and not writable, it will be checked whether it is caused for recording. If so, dVMM will do the same thing for the write. All the records are produced during the first attempt to read or write for the sequential merging; these records are not written to the log file. After the first access to the page, the same subsequent accesses will not be interposed. So, there are only limited times of the control actions of dVMM.

B. Implementation of DCSM's record and replay

To record and replay the DCSM, some external non-determinism must be treated differently. For instructions like RDTSC, the exact logical time of the result delivery after the instruction's execution must be logged and replayed. For other external non-determinism, signals are delivered at the beginning of the quantum.

To record the exact logical time of non-determinism's occurrence, a tuple <eip, bc, ecx> is used to represent the logical time, where eip is the instruction counter, bc is the performance counter and ecx is a register used for string operations. As in figure 6, the recorder program in user space will communicate with KVM by forwarding custom commands through the ioctl() interface of the kvm device. With these commands, users can run an assigned VM as a DCSM and replay it if needed. During recording, the recorder is wakened to read the records from KVM. Then the recorder writes the records in the log file. During replay, the recorder keep extracting the records from the log file and passing them to KVM with the ioctl() interface. After receiving enough records, the assigned VM is able to run. If KVM does not have any records for replay, it will check whether the recorder has marked that all records have been sent. If not, the assigned VM will be paused until new records arrive. Otherwise, the VM continues running.

IV. EVALUATION

To get the performance evaluation of DCSM, we have to know exactly the overheads caused by the VMEXITs of the quanta for synchronization. In our virtual environment, we ran the SPLACH2 benchmark suite [17] to evaluate the design of parallel processors. For some applications, we chose input parameters to make them run for around 60 seconds so that the actual workload can be distinguished. The tests we ran were fmm, ocean, water-spatial, lu and radix. The modified virtual machine monitor KVM ran on a machine with a dual Intel Core (TM) 2 64-bit processor (2 cores total) clocked at 2.93 GHz, with 4GB of memory running linux 2.6.38.5.

Figure 7 shows the overheads of a two processor KVM guest that ran the tests in it. In the experiment, the guest's processors didn't re-execute their quanta even the quanta visited the same page. And we didn't record anything to a

 ISBN: 978-1-61208-153-3

log file. Therefore, the result mainly does not include the overheads caused by the re-execution of the quanta, nor can the record size be gained. The results of different quantum sizes as 8K, 32K, 128K were compared with that of the normal execution of the guest.

The results show that a larger quantum can reduce the VMEXITs and page faults' frequency. With a quantum size larger than 32K, the overheads are less than 3x. So a larger quantum can have a better performance in the experiment. But if quanta's re-execution is taken into account, a larger quantum will generate a longer re-execution time. In this case, the quantum size must be chosen carefully.

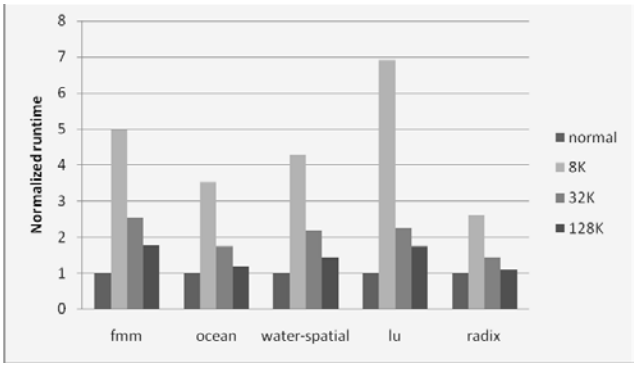


Figure 7. Overhead of DCSM for a two processor KVM guest without quanta's re-execution.

V. DISCUSSION

In the cloud, virtual machine monitors like KVM can provide users with some virtual resources based on the vast physical resources. Many servers are developed as parallel programs and can run in a virtual machine in the cloud. They may encounter an intrusion or a bug when providing service. Our solution can be applied to replay the execution and help developers fix those problems.

Our solution is quite similar to dOS [8], while dOS is implemented in an operating system and our dVMM can enforce the whole guest operating system. Based on our virtual private memory model, many other scheduling algorithms can also be applied. All evaluation experiments of dOS are done on 8-core 2.8GHz Intel Xeon E5462 machines with 10GB of RAM. Without recording the internal non-determinism due to interleaving of threads, dOS produces about 1000 times smaller logs than SMP-ReVirt [7]. The log size of dVMM depends on the communication between the virtual machine and external environment. When dealing with the entire system, dVMM can eliminate much more logs due to the interleaving of CPUs. So dVMM can have a smaller log size than SMP-ReVirt. Since dVMM have to deal with all the processes in an operating system, it will produce more logs than dOS if more processes communicate with external environment.

However, the main overhead of dVMM is due to the communication between quanta, the same as dOS. According to the evaluation of dOS, the overhead of Chromium with a scripted user session opening 5 tabs and 12 urls is about 1.7x on average. The main factors causing the overhead are the quantum size, single-stepping and the communication between quanta. Due to the cost of VMExit of each quantum for synchronization, the slowdown of dVMM is no more than 7x in our tests. But it will become much smaller if the multithreaded program has a good locality of or only a few memory accesses, as well as a suitable quantum size chosen.

There are also many optimizations that can be used to improve dVMM's performance. First, to reduce the probability of re-execution of a quantum, some methods like forward in DMP [4] can also be applied. Second, some binary translation technologies can be used to pre-process the instructions and pre-allocate some shadow pages for the vcpus to reduce more page faults and VMExits in future execution. Other optimizations can also be applied to improve dVMM's performance.

VI. RELATED WORK

At language level, parallel languages such as SHIM [10] and DPJ [9, 11] can enforce determinism, but require rewriting the code. Determinator [1] is implemented on a microkernel and proposes a new programming model. The whole new programming type may not be suitable for some common used applications and it is only implemented on a microkernel now. dOS [8] proposes Deterministic Process Groups (DPG) to ensure the concurrency determinism and a shim layer to replay DPG, which is a solution only implemented in linux. But our solution does not need a whole new programming model and supports different multithreaded programs in different operating systems.

Some hardware-based system such as DMP [4] and Calvin [5] can obtain a good performance, but require custom hardware support. DMP provides different methods to gain determinism. Some of the methods utilize the transactional memory, which is similar to our solution. However, their implementations need custom hardware support, which may not be suitable for the community hardware in the cloud. Some technologies like RCDC [12] and CoreDet [6] use a combination of hardware and software support. They not only use custom hardware, but also need software support like compliers, which forces an application to be recompiled before running. On the contrary, dVMM is a software only solution and can support arbitrary, unmodified software.

There are also many technologies focused on record and replay of a multithreaded program. Like dVMM, SMP-ReVirt [7] can replay the whole system which encapsulates multithreaded programs and their environments. Unfortunately, it has high overheads and large space costs, which is largely owing to the recording of the execution interleaving. However, DCSM does not have to record the interleaving of CPUs compared with SMP-ReVirt. PRES [13] and ODR [14] log a subset of shared memory interactions, reduce the log size, but have increased costs in replay when doing the search of the execution space. dVMM utilizes the hardware VT technology and enforces the deterministic execution with very few controls. Therefore, without logging shared memory interactions, dVMM can have much smaller

 ISBN: 978-1-61208-153-3

log size than SMP-ReVirt. Without searching the execution space, dVMM can perform better in replay than PRES and ODR.

VII. CONCLUSION

This paper proposed a virtual private memory model. Based on this model, DMP-VPM algorithm is introduced to control the deterministic execution of the multiprocessor virtual machine. This controlled virtual machine is called deterministic concurrency state machine. And a record-and-replay scheme for this DCSM is designed. With the internal determinism and the record of external non-determinism, the repeatability of the whole system can be ensured, providing support for debugging, intrusion analysis, etc. Without quanta's re-execution and quantum size no less than 32K, the DCSM generates overheads less than 3x. With a carefully chosen quantum size, the DCSM is supposed to have acceptable overheads.

REFERENCES

[1] A. Aviram, S. Weng, S. Hu, and B. Ford. Efficient System-Enforced Deterministic Parallelism. In OSDI. 2010, pp. 193-206.

[2] J. R. Douceur and J. Howell. Replicated Virtual Machines. Technical Report MSR TR-2005-119, Microsoft Research, Sep 2005.

[3] S. Lu, S. Park, E. Seo, and Y. Zhou. Learning from Mistakes-A Comprehensive Study on Real World Concurrency Bug Characteristics. In ASPLOS. 2008, pp. 329-339.

[4] J. Devietti, B. Lucia, L. Ceze, and M. Oskin. DMP: Deterministic Shared Memory Multiprocessing. In ASPLOS. 2009, pp. 85-96.

[5] D. Hower, P. Dudnik, D. Wood, and M. Hill. Calvin: Deterministic or Not? Free Will to Choose. In HPCA. 2011, pp. 333-334.

[6] T. Bergan, O. Anderson, J. Devietti, L. Ceze, and D. Grossman. CoreDet: A Compiler and Runtime System for Deterministic Multithreaded Execution. In ASPLOS. 2010, pp. 53-64.

[7] G. W. Dunlap, D. G. Lucchetti, P. M. Chen, and M. A. Fetterman. Execution Replay for Multiprocessor Virtual Machines. In VEE. 2008, pp. 121-130.

[8] T. Bergan, N. Hunt, L. Ceze, and S. Gribble. Deterministic Process Groups in dOS. In OSDI. 2010. pp. 177-191.

[9] R. L. Bocchino Jr., V. S. Adve, S. V. Adve, and M. Snir. Parallel Programming Must Be Deterministic by Default. In HotPar. 2009, pp. 4-4.

[10] S. A. Edwards, N. Vasudevan, and O. Tardieu. Programming shared memory multiprocessors with deterministic message-passing concurrency: Compiling SHIM to Pthreads. In DATE. 2008, pp. 1498-1503.

[11] R. Bocchino, V. Adve, D. Dig, S. Adve, S. Heumann, R. Komuravelli, J. Overbey, P. Simmons, H. Sung, and M. Vakilian. A Type and Effect System for Deterministic Parallel Java. In OOPSLA. 2009, pp. 97-116.

[12] J. Devietti, J. Nelson, T. Bergan, L. Ceze, and D. Grossman. RCDC: A Relaxed Consistency Deterministic Computer. in ASPLOS. 2011, pp. 67-78.

[13] S. Park, W. Xiong, Z. Yin, R. Kaushik, K. Lee, S. Lu, and Y. Zhou. Do You Have to Reproduce the Bug at the First Replay Attempt? - PRES: Probabilistic Replay with Execution Sketching on Multiprocessors. In SOSP. 2009, pp. 177-192.

[14] G. Altekar and I. Stoica. ODR: Output-Deterministic Replay for Multicore Debugging. In SOSP. 2009, pp. 193-206.

[15] J. Huang, P. Liu, and C. Zhang. LEAP: Lightweight Deterministic Multi-processor Replay of Concurrent Java Programs. In FSE. 2010, pp. 385-386.

[16] E. Berger, T. Yang, T. Liu, and G. Novark. Grace: Safe Multithreaded Programming for C/C++.. In OOPSLA. 2009, pp. 81-96.

[17] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In ISCA, 1995, pp. 24-36.

 ISBN: 978-1-61208-153-3

A Generalized Approach for Fault Tolerance and Load Based Scheduling of Threads in Alchemi .Net

Vishu Sharma, Manu Vardhan, Shakti Mishra, Dharmender Singh Kushwaha
Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology, Allahabad
Allahabad, India
Email: {cs0916, rcs1002, shaktimishra, dsk}@mnnit.ac.in

***Abstract*— Computational grids can be best utilized by the divide and conquer approach, when it comes to executing a large process. In order to achieve this, building multithreaded application is one of the efficient approaches. The threads are scheduled on different computational nodes for execution. One of the frameworks that support multithreaded applications is Alchemi, but it does not incorporate any load based scheduling and fault tolerance strategy. In Alchemi, a manager node uses first come first serve (FCFS) scheduling to schedule threads on executors (node that execute independent thread), but it does not consider any CPU load on which the executors are running. Moreover if an executor fails in between, then the manager node reschedules the thread on other executor node. One solution for the above problem is to save intermediate results from each thread and reschedule these threads on another executor. We propose an approach that provides fault tolerance in Alchemi by using Alchemi Replica Manager Framework (ARMF), where the manager node will be replicated on one of its executor node. The proposed algorithm is 6-16 percent more efficient than FCFS, when implemented in Alchemi.**

Keywords-ARMF; FCFS; fault tolerance; load based scheduling.

I. INTRODUCTION

A computational grid provides distributed environment in which user jobs can be executed either on local or on remote machines [2]. In grid, user jobs are considered as applications that contain the tasks to be executed. Further, each independent task is represented by a single thread. Whenever a user is having a job which contains multiple individual tasks it is better to use multithreading environment because thread creation and management is easier and faster than process creation. Threads provide following advantages over processes [20]:

- Thread creation takes less time because it uses the address space of process that owns it.
- Thread termination is easier than process
- There is less communication overhead between threads because address space is shared.

Figure 4 shows the architecture of Alchemi. It shows a manager connected with four executors. Alchemi provides API's that are used to create grid applications. In Alchemi, Gthread class is used to implement the multithreading [13]. Figure 1 shows the Gthread class and its structure. It contains an abstract method start ().Each thread is given a priority by a user. Alchemi .NET has the 5 priority levels from lowest to highest. Each application consists of several threads. The manager node is responsible for the scheduling of threads on different executors and collects the results from these executors after successful completion. The two issues related with Alchemi are scheduling of threads and fault tolerance.

The first issue is that of scheduling, where the manager node uses FCFS [17] policy for scheduling. It stores the threads according to their priority and schedules the highest priority thread on next available executor. It does not consider the CPU load of the processors on which the executors are running. If more than one executor is available at a time, it might happen that a thread is scheduled on a more loaded executor which can degrade the performance.

```
public abstract class GThread : MarshalByRefObject
{
public abstract void Start();
/* method is overridden by the class that inherits the
Gthread class*/ }
```

Figure 1. Structure of Gthread class.

Second issue is that of Fault tolerance, this helps system to recover from faults [4]. In case of Alchemi grid, if a thread is scheduled on an executor and due to some reasons, the executor crashes, the thread running on this executor also crashes. In such a case, the manager reschedules this thread on another executor and the thread is restarted from the scratch. Moreover there may be the case when the Alchemi manager can crash and all the executors currently registered with the manager will come to halt.

One solution to the above problem is discussed in [5]. The authors have used a file based implementation in which a file stores the intermediate results and if thread crashes it is rescheduled on another executor and resumes its execution from last successful result, without starting from the scratch. It reads the last successful result from the stored file.

The second limitation in [5] is that all the fault tolerance code overhead is on the user who submits the application. The Alchemi manager is not responsible for any kind of activity. Thus we came across the following issues that are yet to be resolved in Alchemi .NET.

- If a thread execution fails in between, then how the values produced by this thread (till the point of failure) can be saved at manager node and how the

 ISBN: 978-1-61208-153-3

remaining work of the failed thread is assigned to other thread. Approach given in [5] does not talk about how this kind of fault tolerance mechanism can be implemented in manager node. It completely relies on user. Neither have they discussed about the possibility of Alchemi manager failure.

- If the more than one executor is available at same time and the CPUs on which these executors are running might be overloaded then how to schedule threads to get a better solution.

To address above mentioned issues, a generalized approach is proposed as under, in which fault tolerance is provided for computational applications [12] running on a global grid.

- To provide a kind of check pointing scheme which stores the intermediate results produced by threads and the Alchemi manager node is incorporated with the facility to control the execution of failed threads and reschedule these threads on other available executors. In case of Alchemi manager failure the ARMF is proposed, which will provide the backup in such cases.
- To choose the best available executor on the basis of the load of CPU.

For more complex scientific application this approach may not work well as it requires users input. Hence, the proposed approach is confined to the computation intensive processes.

Rest of the paper is organized as follows. Section 2 describes the existing work done in fault tolerance and scheduling in grids. Section 3 shows the proposed approach. Section 4 shows the case study using the proposed algorithm and Section 5 derives the conclusion.

II. RELATED WORK

In load-based scheduling [18], load information can't be exchanged much frequently due to network communication overheads [2]. It is desirable to exchange the load information only when it is needed.

In a system, fault tolerance is achieved by means of some redundancy that could be hardware, software or time redundancy [19].

Vladimir et al. [7] discuss about the scheduling of divisible load applications, where the resources are selected dynamically, based on the intermediate results. In this approach, application specific requirement also plays a vital role in selecting the resources. But this approach is applied at application level and does not concentrate on multi-threaded grid [15] environment.

Zeljko et al. [8] discusses an improved scheduling strategy in Alchemi. This approach still relies on a static strategy for selecting the executors and adds nothing to fault-tolerance. To achieve fault tolerance, a file based technique is proposed in [5]. First problem with this approach is that it places the burden of creating and manipulating the file on the user who creates the application and the manager does not contribute in any kind of fault tolerance activity. Second problem is that for each thread there is a single file, means incurring more overhead on the manager node. This approach [5] has been shown only for one application. Authors have not discussed how other applications can be implemented using this approach.

One of the characterization techniques is given in [10]. In this technique, individual machine faults are defined as, resource level fault and faults in global environment of grid are considered as service level faults. This paper does not elaborate much about the resumption of jobs from the point where it was crashed.

Another improved approach is given in [11]. Fault tolerance is achieved at job level but as each job can be divided into individual tasks using multithreading so several issues like which thread got faulted, how to combine the results from faulted threads etc remain unhandled.

An approach for thread scheduling is shown in [16], where different threads are scheduled to download files from different servers. But in this approach if a thread fails to execute, it is rescheduled after all threads complete their execution.

All the above discussed literature work motivated us to put efforts for providing a novel solution to fault tolerance and load based scheduling in Alchemi .NET.

III. PROPOSED APPROACH

In our approach two concepts, first is fault tolerance and second is scheduling of threads, based upon CPU load are integrated into single algorithm. We first discuss about the fault tolerance approach followed by the thread scheduling based on CPU load. The proposed approach did not consider the manager load, as the thread will always execute on the executor node, not on the manager node. There may be the case of manager failure, which we have discussed below.

A. Fault Tolerance Approach

In Alchemi .NET the applications are divided into individual threads and these threads are scheduled on currently available executors. If a thread execution stops in between then the work done by that thread till that point will be lost.

In [5], an approach is proposed in which file is created for each thread which keeps track of thread execution. This approach puts extra burden of creating and using the file over the application programmer who creates the application.

We propose an approach that enhances this idea [5] by incorporating the manager with the capability of creating and maintaining the file. Each application, submitted by a different user is different and hence the intermediate results (variables) would be different. We try to generalize this approach so that different kind of applications can be executed in the same way. To support this kind of dynamicity, we are using the XML-file. As the application is submitted, the manger node creates an XML-file with relevant information loaded into it. This information is responsible for resuming a crashed thread.

A big challenge in this approach is how to identify these variables. In our approach these variables are supplied by the user who submits the application because the user knows what and where the values must be stored. During the thread

 ISBN: 978-1-61208-153-3

execution, the executor is responsible for saving these values into the XML-file that is on manager. Whenever a crashed thread is rescheduled on different executor the manager node will extract the values from that XML-file and will pass it to the thread so that it can resume its operations.

```
Public class table: Gthread /* user code */ {  table (
int starting_number, int last_number)
    {/* constructor initializes the values in XML file */
    /* initialization of values done by manager */
    }  Public void start()
    {for(num=starting_number;num<=last_number;
number++)
        for( int i=1; i<= 10; i++)
            { result=num*i; }
    savetofile(num, result);
    }}  Savetofile(values )/* method runs on executor */
    {  /* sends intermediate values to the manager node
*/ }
```

Figure 2. Proposed structure of thread implementation.

Figure 2 shows the structure of the threaded class that a user implements. This class extends the Gthread class given in Figure 1. The Structure of the XML file is given in Figure 3. This file contains the values for threads for which processing has been successful.

```
<file  application_id= " "><thread>
<init><thread_id> 123</thread_id>
<first number>1</first number>
<last number>5<last number>
<completed>yes</completed></init></thread>
<thread>
<init><thread_id>163</thread_id>
<first number>6</first number>
<last number>10<last number>
<completed>no</completed>
</init>
</thread> </file>
```

Figure 3. Structure of XML file.

In the existing file-based fault tolerance approach [5], fault tolerance is supported at user end. Fault tolerance is completely dependent on application user. In our proposed approach, fault tolerance is supported by the Alchemi manager, application user need not to concern about its implementation.

Next, in Alchemi architecture, there is no provision for handling the situation where manager can fail. Under these circumstances all the Executors registered with the failed manager will stop executing, and the whole system will come to halt. There should always be some backup / replica manager, so that single point failure can be avoided.

Alchemi manager which is responsible for managing the execution of grid applications can be replicated. This can be achieved by replicating the Alchemi manager at its one of the Executor, which is currently registered with this manager.

Figure 4 describes the whole scenario. The manager node is connected with four executors. Each executor executes an independent thread. User application is containing 3 threads.

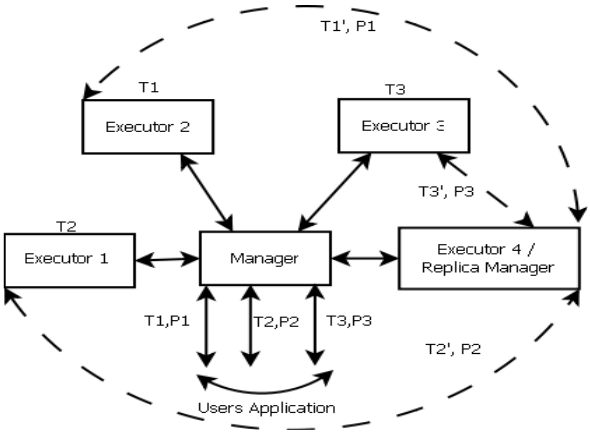


Figure 4. Architecture of Alchemi and Alchemi Replica manager.

P1, P2, P3 are the thread priorities assigned by the user for the respective thread. T1', T2', T3' are the thread associated with the Replica manager which is on Executor 4.

The information that needs to be transferred to the Executor node, so that the Alchemi manager can continue functioning from the point of failure and not from the scratch, is stored in a XML file with the manager. This XML file needs to be replicated to that Executor node, which is acting as a replica of Alchemi manager. Periodic updation of this XML file is required, so as to maintain the consistency of the system.

The information that needs to be transferred to the executor node is stored in a XML file with the manager, so that the Alchemi manager can continue functioning from the point of failure and not from the scratch. This XML file needs to be replicated to that Executor node, which is acting as a replica of Alchemi manager. Periodic updation of this XML file is required, so as to maintain the consistency of the system.

In the present Alchemi framework, an executor can register itself only with one manager. Issue associated here, from the developers/programmers perspective is "how the Executor will register itself with the new manager i.e., the replicated manager in case of manager failure". With the present framework, if the manager fails, the new replica manager needs to inform all the executors, registered with the failed manager, to get them registered with the new replica manager. Or there should be some provision by which an executor can register it with more than one manager.

B. Modified Scheduling Algorithm

Alchemi .NET provides its grid API that is used to develop grid applications to be submitted to the Alchemi. Each application contains threads. Number and priority of

 ISBN: 978-1-61208-153-3

threads are defined by the application programmer. It does not consider current performance of the CPU on which the executor is running. If at the same time two executors are available and one of these is overloaded whereas other is not, so it might happen that a highest priority thread is scheduled on an executor that is overloaded. In those cases when the higher priority thread execution duration is large, this overloaded executor might degrade the performance.

In the proposed approach, an executor does not send its load information periodically, rather it sends it whenever an executor finishes execution of a thread and it is ready to receive a new thread from the manager. We assume that no thread is interrupted during its execution due to the load information on its machine.

In Figure 5 default mechanism of selecting the executors is shown.

```
Step1: Thread=gethighestprioritythread();
Step2: Executor=Getnextavailableexecutor()
Step3: create new schedule with executor and thread.
Step4: Schedule(dedicateschedule);
```

Figure 5. Default scheduling mechanism in Alchemi.

Figure 6 shows the modified algorithm, if more than one executor is available at the same time our algorithm selects the best one.

```
Step1: Thread= Gethighestprioritythread();
Step2: Execut_available[]=Getcurrent_avail_executor()
       Executor= Executoravailable[].getleastloaded().
Step3: Create new schedule with executor and thread.
Step4: schedule(dedicateschedule);
```

Figure 6. Modified mechanism.

C. Algorithm

The algorithm combines both the approaches discussed above. Its theoretical description is given in Figure 7.
The architecture of the proposed approach is shown in Figure 8. A ft_thread is added at manager and executor nodes. At manager node the ft_thread is running continuously and is responsible for receiving the intermediate values from the ft_thread running on executors. It writes the intermediate values into the XML file and reads them in case a faulty thread needs to be rescheduled.

1. Get the highest priority thread from the database.
2. Create the entry in XML file for that thread.
3. Get the available executors check their load factor and if more than one executor is available get the minimally loaded executor.
4. Receive the intermediate values sent by the executor for that thread.
5. Replace the existing value in XML file with the recently received values.
6. If executor gets disconnected then check the thread status allocated to that executor. If it is not completed create new thread with the same thread id that was executing on the crashed executor.
7. Supply the last successful results to that newly created thread so that it can resume its execution.
8. Get the minimally loaded executor and assigned this thread to that executor.
9. Repeat steps 1 to 8 until the thread database is empty.

Figure 7. Proposed algorithm.

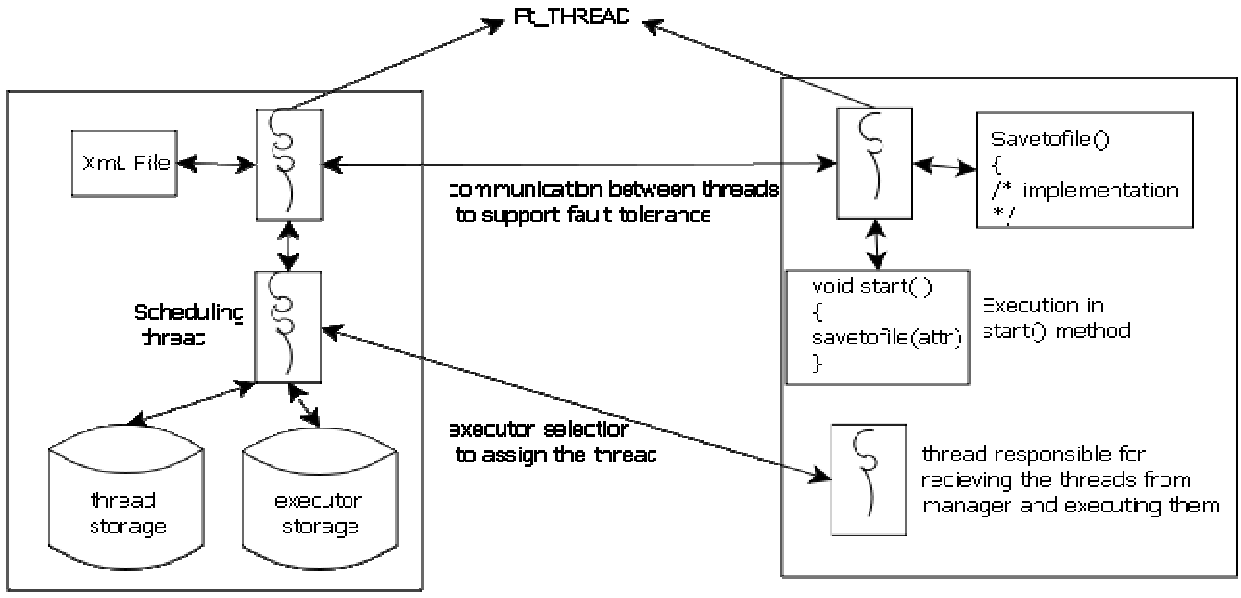


Figure 8. Architecture of Fault Tolerant Alchemi.

IV. CASE STUDY

We evaluate the scenario where an overloaded executor might be a bottleneck for the performance. In Figure 9, we show an example with **t**hree executors on which threads are scheduled. We assume that all executors that are not overloaded execute the threads in approximately same time.

In Figure 9, an executor is marked as overloaded and it takes more time to execute a thread as compared to an average loaded or underloaded executor.

An average loaded or underloaded executor takes 4 units of time to execute a high priority thread and 2 units of time to execute a low priority thread whereas an overloaded executor takes 6 units of time for high priority thread and 3 units of time for low priority thread. Hence the completion time for this application according to FCFS scheduling is 9 units of time.

In Figure 10, we see another arrangement of threads on the executors. In this low priority threads are scheduled on overloaded executor and all high priority threads are scheduled on less overloaded executors. The completion time of the application is 8 units of time.

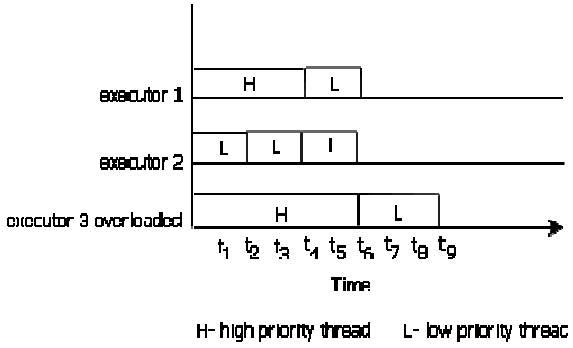


Figure 9. Arrangement of threads on executors according to default mechanism.

Load information collected from the executor also helps in selecting the best available executor whenever a thread is rescheduled after a crash. In our approach we assume that if at any point of time two executors are available we select one which is less loaded.

In the simulated environment we analyze the behavior of proposed application with different applications. These applications are included in random. In Alchemi, different executor nodes are connected to manager node. From these available executor nodes some are overloaded in comparison to others.

Table I shows five applications, number of high and low priority threads for each application. In this table, column name A.N. stands for application number, N.T. for Total number of threads in an application, N.H.P for Number of high priority threads, N.L.P. for Number of low priority threads and E.E.T. for Expected execution time on normal executor. In Table II, completion time for FCFS and proposed algorithm is shown. The total number of threads in a single application is shown in Table I. The execution time for a thread is shown on a normal executor. We assume that an overloaded executor takes 50% more time to execute a thread. In Table I application number 4 has threads of same type, i.e., all the threads are having same priority. In this case also, our proposed algorithm performs well.

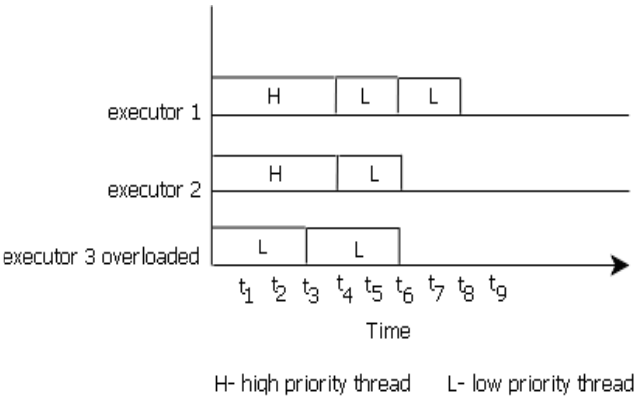


Figure 10. Arrangement of threads on executors according to proposed algorithm.

TABLE I. APPLICAION CHARACTERISTICS. H RPRESENTS THE HIGH PRIORITY THREAD AND L REPRESENTS THE LOW PRIORITY THREAD

A.N.	N.T.	N.H.P	N.L.P	E.E.T.	
				H	*L*
1	7	2	5	4	2
2	14	2	12	6	4
3	11	2	9	10	6
4	9	9	0	6	-
5	6	4	2	10	5

Figure 11 shows the results obtained from FCFS and proposed algorithm in simulated environment. It shows that our proposed algorithm gains better completion time. Figure 8 also shows that for a given application set, our proposed algorithm is 6-16 % more efficient in comparison to FCFS algorithm. In case where all the threads have same priority, it is 11% more efficient than the FCFS algorithm.

V. CONCLUSION

An approach that achieves fault tolerance supported by manager node of Alchemi is presented in this paper. In comparison to other approaches, the scheduling of threads on various nodes after the crash requires no user intervention. Rather the proposed approach implements fault tolerance in system by using manager node and executor node. We also propose an Alchemi Replica Manager Framework (ARMF) and a scheduling algorithm based on the load information of executor nodes. ARMF replicates the XML-file, which is maintained by the manager node and stores all the required information about the threads executing on the executors, to one of its executor, which will be acting as the replica manager in case of manager failure. Our proposed algorithm selects the executors depending upon the load information of currently available executors. This helps Alchemi manager to select best executor (least loaded for a high priority

 ISBN: 978-1-61208-153-3

thread) amongst available ones. In performance study, it has been found that the proposed approach is 6 – 16 % more efficient than FCFS, when implemented in Alchemi. Alchemi Replica Manager Framework (ARMF) provides a mechanism to replicate manager node to one of its executor.

TABLE II. COMPARISON BETWEEN PROPOSED ALGORITHM AND FCFS

Application number	No.of exec-utors	No.of over loaded CPUs	FCFS Comp-letion time	Proposed algorithm completion time
1	3	1	9	8
2	4	2	21	18
3	3	1	33	28
4	3	1	18	16
5	3	1	22.5	20

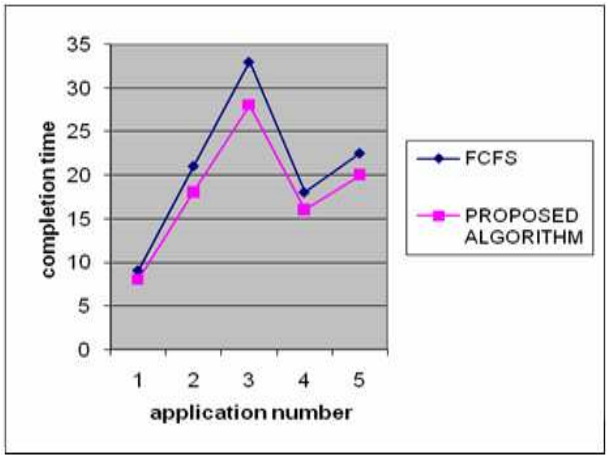


Figure 11. Performance study of both algorithms.

REFERENCES

[1] Sunita Bansal, Gowtham K, and Chittrnjan Hotta: Novel adaptive scheduling Algorithm for computational grids. Proceeding IMSAA'09 Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications, pp. 1-5, 2009.

[2] Ruchir Shah, Bhardwaj Veeravalli, and Manoj Misra: On the Design of Adaptive and Decentralized load balancing algorithms with Load estimation for computational grid environments, IEEE transactions on parallel and distributed systems, vol. 18, no. 12, pp. 1675-1685, 2007.

[3] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal: Alchemi: A .NET-based Grid omputing Framework and its Integration into Global Grids. In: Grid Computing and Distributed **S**ystems (GRIDS), Technical Report, GRIDS-TR-2003-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, pp. 1-17, 2003

[4] William C. carter: Fault-Tolerant Computing: An Introduction and a Viewpoint, IEEE TRANSACTIONS ON COMPUTERS, vol. C-22, no. 3, pp. 225 – 229, 1973.

[5] Md. Abu Naser Bikas, AltafHussain, Abu Awal Md. Shoeb, Md. Khalad Hasan, and Md. Forhad Rabbi: File Based GRID Thread Implementation in the .NET-based Alchemi Framework, Multitopic ConferenceI, NMIC. IEEE Intern., pp. 468-472, 2008.

[6] Veeravalli Bharadwaj, Debashish Ghose, and Thomas G. Robertazzi: Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems, Cluster Computing 6, pp. 7–17, 2003, 2003.

[7] Vladimir V. Korkhov, Jakub T. Moscicki, and Valeria V. Krzhizhanovskaya: The User-Level Scheduling of Divisible Load Parallel Applications With Resource Selection and Adaptive Workload Balancing on the Grid, IEEE systems journal, vol. 3, no. 1, pp. 121-129, 2009.

[8] Zeljko Stanfel, Goran artinovic, and ZeljkoHocenski: Scheduling Algorithms for Dedicated Nodes in Alchemi Grid. IEEE International Conference on Systems, Man and Cybernetics, pp., 2531 – 2536, SMC 2008.

[9] Gracjan Jankowski, Radoslaw Januszewski, and Rafal Mikolajczak.: Improving the fault-tolerance level within the GRID computing environment - integration with the low-level checkpointing packages, CoreGRID Technical Report Number TR-0158, June 16, 2008.

[10] Jes´us Montes CeSViMa, Alberto S´anchez, and Mar´ıa S. P´erez.: Improving grid fault tolerance by means of global behavior modeling, Ninth International Symposium on Parallel and Distributed Computing, pp. 101-108, 2010.

[11] HwaMin Lee1, DooSoon Park1, Min Hong1, Sang-Soo Yeo2, SooKyun Kim3, and SungHoon Kim4.: A Resource Management System for Fault Tolerance in Grid Computing, International Conference on Computational Science and Engineering, pp. 609-614, 2009

[12] Nirmalya Roy and Sajal K. Das: Enhancing Availability of Grid Computational Services to Ubiquitous Computing Applications, IEEE transactions on parallel and distributed systems, vol. 20, no. 7, pp. 953-967, 2009.

[13] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal: Alchemi: A .NET-based Enterprise Grid Computing System, 6th International Conference on Internet Computing. Las Vegas, pp. 1-10, 2005.

[14] Sandeep Singh Rawat and Dr. Lakshmi ajamani: Experiments with CPU Scheduling Algorithm on a Computational Grid, IEEE International Advance Computing Conference (IACC 2009), pp. 71-75, India, 2009

[15] Jos´e Augusto Andrade Filho, Rodrigo ernandes de Mello, and Evgueni Dodonov: Toward an efficient Middleware for Multithreaded Applications in Computational Grid, 11th IEEE International Conference on Computational Science and Engineering, pp. 147-154, 2008.

[16] Suvarna N. A and Dinesh Chandra: Evaluation of Improvement Algorithms for dynamic Co-Allocation with respect to parallel downloading in Grid Computing, First International Conference on Integrated Intelligent Computing, pp. 79-83, 2010.

[17] U. Schwiegelshohn and R. Yahyapour: Analysis of first-come-first serve parallel job scheduling, Proceedings of the ninth annnal ACMSIAM symposium on Discrete algorithms (SODA'98), pp. 629-638, 1998.

[18] Cui Zhendong and Wang Xicheng.: A Grid Scheduling Algorithm Based on Resources Monitoring and Load Adjusting, Knowledge Acquisition and Modeling Workshop, 2008, KAM Workshop, pp. 873-876, 2008.

[19] Nils Mullner, Abhishek Dhama, and Oliver Theel: Deriving a Good Trade-off between System Availability and Time Redundancy, Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, pp. 61-67, 2009.

[20] http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/threads.htm accessed on 10-05-2011.

 ISBN: 978-1-61208-153-3

Reducing the Human Cost of Grid Computing With glideinWMS

Igor Sfiligoi, Frank Würthwein,
Jeffrey Michael Dost, Ian MacNeill
University of California San Diego
La Jolla, CA 92093, USA
email: isfiligoi@ucsd.edu, fkw@ucsd.edu,
jdost@ucsd.edu, imacneill@ucsd.edu

Burt Holzman, Parag Mhashilkar
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA
email: burt@fnal.gov, parag@fnal.gov

***Abstract*—The switch from dedicated, tightly controlled compute clusters to a widely distributed, shared Grid infrastructure has introduced significant operational overheads. If not properly managed, this human cost could grow to a point where it would undermine the benefits of increased resource availability of Grid computing. The glideinWMS system addresses the human cost problem by drastically reducing the number of people directly exposed to the Grid infrastructure. This paper provides an analysis of what steps have been taken to reduce the human cost problem, alongside the experience of glideinWMS use within the Open Science Grid.**

Keywords-Grid; glideinWMS; human cost

I. Introduction

Over the past decade, the science community has been moving from dedicated, tightly controlled compute clusters to a widely distributed, shared Grid infrastructure in an effort to both increase the average equipment utilization and gather additional compute resources in times of need. One such Grid infrastructure is the US-based Open Science Grid (OSG) [1,2], an umbrella organization gluing together groups of scientists from many scientific domains. These groups are normally referred to as Virtual Organizations (VOs), since they have an internal structure. Each VO brings to the community both people and compute resources, with the understanding that their compute resources can be used by other VOs when not needed by the owning VO, and conversely that their users can access resources they don't own, when available.

This system has greatly benefited several VOs, but the early adopters have noticed that using the Grid can have a very high human cost. While the Grid is quite easy to use as long as everything works fine, when something goes wrong, it can take a significant amount of human time to debug and fix the problem. Given that the OSG currently encompasses O(100k) CPU cores distributed over O(100) geographic locations, having at least a few misbehaving nodes at any given time is pretty much a given. And with a community of O(10k) users, each broken node is likely to affect hundreds of users before being fixed. If each user were to spend even half an hour debugging the problem, the total human cost can easily exceed a week worth of time for each such event.

The glideinWMS system [3,4] attempts to reduce the human cost in two ways. It creates a dynamic overlay on top of Grid resources, thus insulating the final users from Grid problems, and it cleanly separates the VO policy handling from the actual Grid interfaces, allowing for a generic Grid-facing service, called a glidein factory, that further limits the exposure to the complexities of the Grid. To the best of our knowledge, this is the only system that supports that.

The glideinWMS has been in use on OSG with a shared glidein factory for over 2 years, and has proven to be a major success, drastically reducing the human cost of several VOs.

Section II provides an overview of the pilot paradigm, and the cost savings associated with it. Section III describes the cost savings due to the glideinWMS approach of separating VO policy from Grid submission. Finally, Section IV provides the analysis of the cost savings that OSG achieved in using the glideinWMS with a shared glidein factory.

II. Cost Advantage of Pilot Infrastructures

A pilot system [3] creates a dynamic overlay pool of compute resources on top of the Grid, as shown in Fig. 1. From the end user point of view, this overlay pool looks and feels exactly like a dedicated, tightly controlled compute cluster of the past, it is just a dynamic one, growing and shrinking depending on workloads and Grid resource availability.

Pilot infrastructures use two mechanisms to shield the users from Grid errors. The first and most important protection is provided by the pilots themselves; if a malfunctioning node kills the pilot before it is able to join the overlay pool, the users will never be aware of the existence of such node, preventing any error condition at its root. Starting the pilot is however not a sufficient condition to assure job success, since user jobs may need access to resources not needed by the pilot itself, e.g., scientific libraries, or they may need them in larger quantities, e.g., disk space. To account for that, most pilot system implementations, and in particular glideinWMS, allow for additional validation procedure to be run before joining the overlay pool; if even one test fails, the pilot aborts and never

 ISBN: 978-1-61208-153-3

joins the pool. This allows for the overlay pool to be well behaved, at least within the limits of the tested properties.

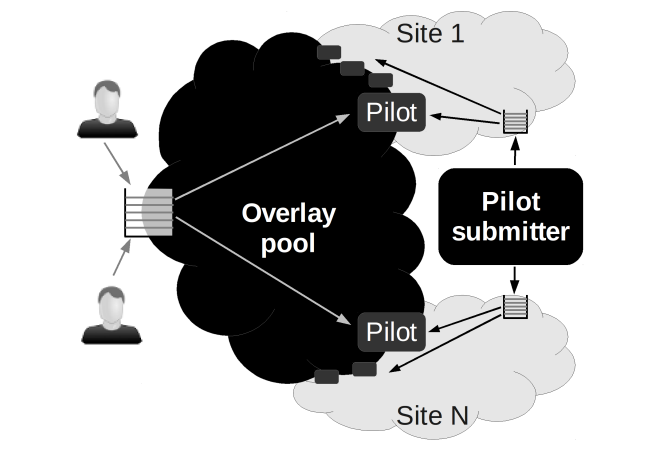


Figure 1. Schematic view of a pilot system

For the final user, the human cost of using this pool is thus comparable to using a truly dedicated compute pool. However, someone still has to create this overlay pool by submitting pilot jobs to the Grid. This pilot administrator will thus be exposed to the Grid-related errors affecting the pilot jobs themselves, and will be responsible for debugging them. While the human cost of this individual will obviously be much higher compared to the human cost of any individual user in the direct Grid submission paradigm, its cost is arguably still much smaller than the aggregate human cost of all the individuals.

There are two reasons for the cost savings. The first one is due to the difference in the type of jobs failing. Each user job is precious, so users have to spend some time recovering each and every one of them. Pilots are instead disposable, since they by themselves don't carry any useful payload, and any failure before an actual user job is started does not represent any loss of data, just reduced efficiency on the failing node. The human cost thus scales only with the number of failing nodes, not failing jobs. As shown in Table I, for a sizable OSG VO of O(1k) users running O(10M) compute jobs per month on O(1k) nodes, if even 1% of those jobs were to fail due to Grid problems, the use of a pilot infrastructure would reduce the effort from debugging O(100k) user jobs to debugging O(10) Grid nodes, thus decreasing the human cost by several orders of magnitude.

TABLE I. COMPARISON OF DEBUGGING COSTS FOR A SIZABLE OSG VO

	Direct submission	**Pilot system**
Metric (/month)	O(10M) jobs	O(1k) nodes
Error rate	O(1%)	O(1%)
Entities to debug	O(100k)	O(10)

The second reason is due to the difference in expertise. End users are typically not interested in computing, being scientists and viewing computing just as a tool, so they will likely spend a large amount of time trying to understand the occasional set of Grid-related problems. Pilot administrators can instead be IT professionals, who are well versed in debugging and fixing these kind of problems. Moreover, they will see similar errors with a much higher frequency, making the time-to-resolution dramatically shorter.

III. IMPORTANCE OF PARTIAL SHARING IN PILOT INFRASTRUCTURES

The typical way of using pilot infrastructures is for each Virtual Organization to install a completely independent instance. This has been the approach of the early adopters of pilot infrastructures, such as the LHCb [5], CDF [6] and ATLAS [7] VOs.

The net result of this approach, however, is the proliferation of pilot administrators. Given that many Grid sites provide resources to many VOs, it also likely results in duplicate effort of debugging errors for pilots that happen to land on the same malfunctioning compute nodes. Offloading the operational load of many VOs to a single operations group would thus result in significant human cost savings, for the same reasons described in the previous section.

One of the reasons why early adopters did not go for a shared solution is that while sharing of a pilot instance is in theory possible, e.g., by simply allowing users from different communities to submit to the same overlay pool, in practice VOs cherish their autonomy, and will not delegate all control to a third party. As long as pilot submission is tightly integrated with the overlay pool operations, as it was the case for the solutions referenced above, partial sharing is not an option.

The glideinWMS addresses the above problem by clearly splitting the pilot infrastructure in two logical pieces, and thus separating the pilot submission from the operation of the overlay pool itself. The pilot submission is handled by one or more **glidein factories**, while the overlay pool is handled by the Condor batch system [8,9], with an additional process, called the **VO frontend**, providing the logic for requesting pilot submission from a glidein factory. Each glidein factory, in turn, can serve multiple VO frontends. The complete architecture is summarized in Fig. 2; please note that Condor pilots are labeled as **glideins**.

Using the glideinWMS, each VO operates its own Condor batch system instance and the associated VO frontend. Since almost all the policies are implemented in this layer, the VO maintains the full control of the overlay pool, thus retaining the look-and-feel of a dedicated, tightly controlled compute cluster.

A VO could also run a glidein factory, but it can instead delegate this activity to a third party without relinquishing any control of the system. The glidein factory is effectively a slave to the VO frontends, submitting pilots on their request. The added value of a glidein factory is mostly in the insulation of a VO frontend, and through it the associated Condor batch system, from the Grid world, providing Grid site specific configuration and validation, and handling all the Grid-related monitoring and error debugging. All of these activities are completely generic, and can be shared among any number of VOs.

 ISBN: 978-1-61208-153-3

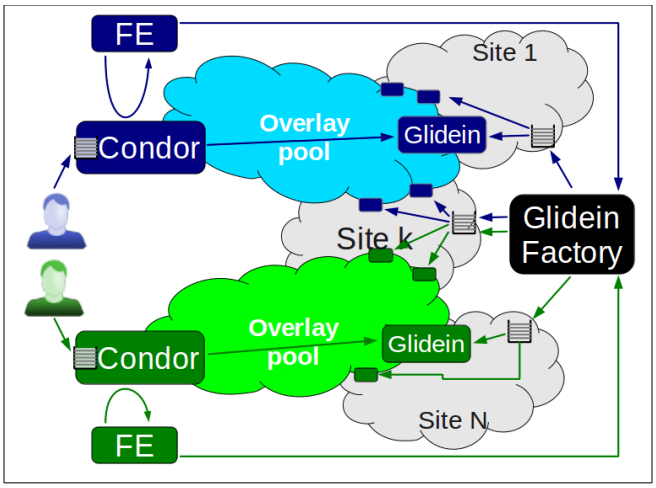


Figure 2. A glideinWMS glidein factory serving two VO frontends

One obvious concern in concentrating all operations to a single entity is that it may become the single point of failure. However, the glideinWMS architecture addresses this concern by allowing each VO frontend to be interfaced with multiple glidein factories, if so desired. While having more than one glidein factory will likely raise the overall cost of the system, it allows to hedge the risk of badly run services, scalability limits as well as complete service shutdowns.

As stated above, the cost savings of using a common glidein factory stem from the fact that many Grid sites provide resources to many VOs; pilots from many VOs will thus land on any malfunctioning or misconfigured worker node. Since the human cost scales with the number of failing nodes being debugged by a pilot administrator, having multiple pilot administrators debug the same node is obviously more expensive compared to a single team doing this task. A quantitative comparison is available in the next section.

IV. GLIDEINWMS IN OSG

The Open Science Grid has been financing the operation of a glidein factory located at University of California San Diego (UCSD) since 2009, with additional contribution coming from the CMS experiment [10]. This instance is operated by three people on part-time basis, with an average effort of little less than one FTE. This glidein factory is open to all OSG VOs, and is currently used by 12 of them, varying in size from small campus-Grid groups to large world-wide communities.

The UCSD glidein factory submits pilot jobs to about 100 Grid sites; out of these, about 30% are used by multiple VOs, as shown in Fig. 3. Grid sites are selected mostly based on which VOs they support. The glidein factory operators obtain this information from multiple sources, including Grid information systems, VO-specific information systems and community knowledge. As far as possible, all information is cross-checked and all new Grid sites validated before being advertised to the served VO frontends. This effort invested in the early validation is usually orders of magnitude smaller than the effort that would be needed to debug misconfigured or malfunctioning sites after the fact, saving precious human time.

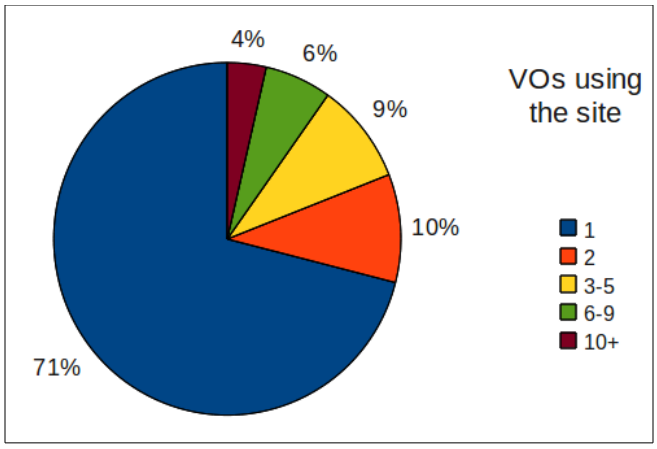


Figure 3. Fraction of OSG glidein factory Grid sites by number of VOs

As shown in Table II, in a typical week, this glidein factory submits about 200k pilot jobs, with about 130k or 65% running on shared Grid sites. Of all the submitted pilot jobs, about 25k or 12% fail the basic node validation, out of which about 22k running on shared Grid sites, yielding a slightly higher 16% error rate. About 25% of all human time is being spent on monitoring these kind of errors, identifying the root cause and collaborate with the affected Grid site administrators in resolving them. Given that significantly more than half of all failing pilots run on shared Grid sites, if each VO had to perform these functions by itself, it would have to spend at least 15% of a person's effort on this, which would result in at least 1.5FTE effort OSG-wide being dedicated to just Grid monitoring and debugging. Using a common glidein factory instance thus saves the OSG community well over a full time person time equivalent.

TABLE II. WEEKLY STATISTICS OF THE OSG GLIDEIN FACTORY

	All sites	Shared sites
Total glideins	200k	130k
Failing glideins	25k	22k

As can be seen, the major effort is currently not dedicated to day-to-day operations. Of the remaining time, about 40% is spent in helping the debugging of problems arising directly between Grid sites and the VO Condor batch system, another 20% writing tools to reduce the needed human effort in the long term, and the final 40% to help VOs to effectively use the glideinWMS. These numbers are also shown in Fig. 4.

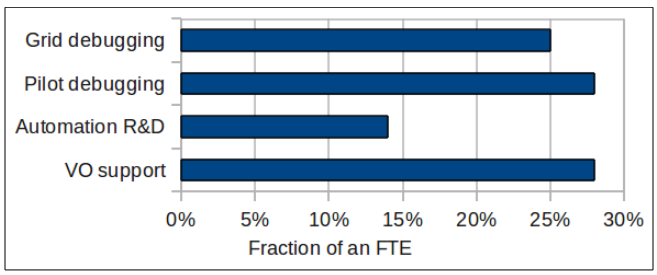


Figure 4. Allocation of effort at the OSG glidein factory

While problems arising from the use of Grid resources by the VO's Condor batch system is technically beyond the glidein factory control, the relevant error logs may not get propagated back to the VO, since the VO communication mechanisms are based on Condor itself. The glidein factory will instead always get them, since it is using the regular Grid mechanisms. The glidein factory operators are thus expected to monitor for these kind of errors as well.

The operators of the OSG-sponsored glidein factory instance also often take a leading role in solving such problems. These problems are often very similar in nature between different VOs; a typical example of such problems are firewall issues. As such, the glidein factory operators have extensive experience in debugging such errors, reducing the total human effort needed. This is especially important since these events, while relatively rare, often don't result in any obvious error messages in the logs, but require speculative thinking in order to be solved. Some of these speculative actions may be scriptable, so time is being invested into the R&D of such tools.

Finally, some of the OSG factory operators also help managing a CMS VO frontend and the related Condor batch system, so together with the experience of supporting several additional VOs from the glidein factory side, they are experts in troubleshooting every component of the glideinWMS system. As such, it is cost-effective to use these people to help all the OSG VOs in the configuration of their glideinWMS components. This does not mean they are involved in day-to-day operations, but they do advise on major configuration decisions.

The number of VOs supported by the OSG glidein factory has been gradually increasing with time. In this period, we noticed that new VOs typically require significant hand-holding, both in terms of configuration help as well as Condor problems on Grid resources during the initial setup period and during major changes in their operation mode, but require relatively little effort most of the remaining time. The human time required by the glidein factory operations team has thus been pretty much constant for all but the initial few months of the glidein factory lifetime, and is expected to significantly decrease once the influx of new VOs slows down.

TABLE III. FTE COST ESTIMATES FOR GLIDEINWMS USE IN OSG

	Shared factory	**VO provided factory**	
		Per VO	**OSG-wide** (12 VOs)
Grid debugging	25%	15%	180%
Pilot Debugging	28%	15%	180%
Automation R&D	14%	10%	120%
Total	67%	40%	480%

The actual cost savings of using a shared OSG glidein factory are difficult to measure, since most VOs using it switched directly from direct submission to the shared-factory pilot paradigm. We thus made an educated guess about the operational costs a typical OSG VO would incur by running its own glidein factory, and presented them in Table III. Given that more than half of all pilots run on shared Grid sites, we estimated that the per-VO cost of both Grid and pilot debugging would scale approximately at the same rate; the automation R&D would instead likely be almost the same as in the shared glidein factory scenario, although the shared glidein factory does need to produce more complex tools. As can be seen, we estimate that the OSG VOs would each use about 40% of an FTE, for an OSG-wide total of about 5 FTEs. This is significantly higher than the 2/3 FTE currently being used by the shared glidein factory.

V. CONCLUSION AND FUTURE WORK

Using Grid resources directly can have a high human cost. While the Grid is quite easy to use as long as everything works well, when something does go wrong, it can take a significant amount of human time to debug and fix the problem. Several OSG Virtual Organizations have thus switched to the use of glideinWMS, which allows for significant cost savings.

The major cost savings come from glideinWMS being a pilot system, i.e. creating a dynamic overlay pool of compute resources on top of the Grid. This shields the end users from Grid errors, and delegates their debugging to a dedicated team of professionals. Furthermore, to achieve savings across different VOs, the glideinWMS architecture separates the pilot submission services from the VO logic, shielding even the VO administrators themselves from the Grid, and allowing for the outsourcing of the Grid error handling to an experienced operations team.

The Open Science Grid has thus invested into a common glidein factory instance, creating an expert operations team that handles the Grid-related monitoring and debugging tasks for all the interested VOs. This allows these VOs to drastically reduce the human effort needed, resulting in global savings of several full time persons time compared to running the complete pilot infrastructure themselves. The cost savings compared to direct Grid submission can instead be counted in tens of FTE, given the thousands of scientists using the Grid resources.

Moreover, the outsourcing of Grid-related activities also contributes to a much better user experience, since most Grid-related problems are caught before the users are exposed to them, and the remaining ones get solved quickly thanks to the experience of the dedicated glidein factory operations team. This contributes to a greater usage of Grid resources by scientists who would otherwise avoid them, due to the high human cost involved.

The system has served OSG well, both in terms of effectiveness and human cost, and is expected to continue to operate in the foreseeable future, with most OSG VOs eventually using it. The only major operational change currently planned is the creation of a second glidein factory instance at a different location, for high availability reasons. While this is expected to slightly increase the operations costs, it is a highly desirable step now that a large community depends on it.

 ISBN: 978-1-61208-153-3

ACKNOWLEDGMENT

This work is partially sponsored by the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG), and the US National Science Foundation under Grants No. PHY-0612805 (CMS Maintenance & Operations), and OCI-0943725 (STCI).

REFERENCES

[1] R. Pordes et al., "The open science grid," J. Phys.: Conf. Ser., vol. 78, 012057, pp. 1-5, 2007, doi: 10.1088/1742-6596/78/1/012057.

[2] "Open Science Grid home page," http://www.opensciencegrid.org/, Accessed June 2011.

[3] I. Sfiligoi et al., "The pilot way to grid resources using glideinWMS," CSIE, WRI World Cong. on, vol. 2, pp. 428-432, 2009, doi: 10.1109/CSIE.2009.950.

[4] "glideinWMS," http://tinyurl.com/glideinWMS, Accessed June 2011.

[5] A. C. Smith and A. Tsaregorodtsev, "DIRAC: reliable data management for LHCb," J. Phys.: Conf. Ser., vol. 119, 062045, pp. 1-6, 2008, doi: 10.1088/1742-6596/119/6/062045.

[6] S. Belforte et al. "GlideCAF: A late binding approach to the grid," Proc. Comp. in High Ener. and Nucl. Phys. (CHEP2006), 2006, id 147, http://indico.cern.ch/materialDisplay.py?contribId=147&sessionId=8&materialId=paper&confId=048, Accessed June 2011.

[7] T Maeno, "PanDA: distributed production and distributed analysis system for ATLAS," J. Phys.: Conf. Ser., vol. 119, 062036, pp. 1-4, 2008, doi: 10.1088/1742-6596/119/6/062036.

[8] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," Conc. and Comp.: Practice and Experience, vol. 17, issue 2-4, pp. 323–356, 2005, doi: 10.1002/cpe.938.

[9] "Condor project homepage," http://www.cs.wisc.edu/condor/, Accessed June 2011.

[10] The CMS Collaboration et al. "The CMS experiment at the CERN LHC," J. Inst, vol. 3, S08004, pp. 1-334, 2008, doi: 10.1088/1748-0221/3/08/S08004.

 ISBN: 978-1-61208-153-3

On the Performance Isolation Across Virtual Network Adapters in Xen

Blazej Adamczyk, Andrzej Chydzinski
Institute of Computer Sciences
Silesian University of Technology
44-100 Gliwice, Poland
{*blazej.adamczyk,andrzej.chydzinski*}*@polsl.pl*

***Abstract*—Virtualization has recently become a very popular technique for utilizing hardware capabilities and lowering infrastructure and maintenance costs. However, making several virtual machines share the same resources can potentially introduce performance isolation problems. Depending on the application, proper quality of service and the performance isolation may present critical requirements for the system.**
In this paper, we focus on network performance isolation among virtual adapters in Xen. We present several experiments demonstrating how activity of one virtual machine can affect the network performance of any other. Additionally, we examine the network I/O scheduler in Xen to see if it is fair, predictable and configurable enough. Finally, we propose an idea on how to modify Xen back-end drivers to improve the network performance isolation.

***Keywords*-performance isolation; Xen; virtualization; network scheduler.**

I. INTRODUCTION

The increasing number of different IT services are making the virtualization idea a very important aspect of computer science. Virtual Machine Monitors (VMMs) bring about the dynamic resource allocation and enable full utilization even of the most powerful servers, while still maintaining good fault isolation between virtual machines (VMs). However, the services provided over the network may require a certain quality, which is not easy to ensure in a virtualized environment. Several VMs can share the same physical network interface as well as other hardware (processor, memory etc.) what likely makes one VM affect other VMs performance. Therefore, the *performance isolation* is crucial in case of some applications and has to be carefully verified.

In this paper, we focus on Xen VMM, [1], which is one of the most popular virtualization platforms and an Open Source project. Firstly, we present a study of the network performance isolation between Xen virtual machines. Different test scenarios allowed us to identify several problems. Secondly, we carefully analyze the Xen CPU scheduler and the network IO scheduler to find out their possible source and resolution method.

The remaining part of the paper is structured as follows. In Section III, Xen general architecture is overviewed. Then, a description of the Xen schedulers is presented in Section IV. Section V describes the testing environment and its parameterizations. The results and discussion on them are contained in Section VI. Finally, an idea of improving the network performance isolation in Xen is presented in Section VII. Conclusions are gathered in Section VIII.

II. STATE OF THE ART

This study verifies that there are problems related with performance and isolation of virtualized network resources. Several previous studies (see [9], [10], [11], [12], [13], [14], [15]) focus on analysis of the performance of IO operations and some of them present partial solutions. Unfortunately, these studies do not examine isolation and manageability in the field of resource sharing in considered virtualization platforms. In [7], however, the authors tried to approach the performance isolation problem focusing on all kinds of resources. Unfortunately, this study was performed on older version of Xen with an older CPU scheduler implementation. They assumed that the main source of the problem is connected with CPU assignment and scheduling. We think however, that to achieve good performance isolation across virtual network adapters the proposed CPU scheduler improvement could be used but is not sufficient. We present that even on a low CPU utilization the problem is still noticeable and is related with network scheduler itself. We have verified that applying a modified for virtualization Weighted Round Robin (WRR) network scheduler improves the performance isolation and provides better control over virtual network devices.

III. XEN VMM

Different virtualization environments have been developed throughout the years. Xen, due to its unique architecture (Fig. 1), is one of the leading solutions. The core of Xen, which is responsible for control over all virtual machines, is a tiny operating system called Xen Hypervisor. Its main tasks are CPU scheduling, memory assignment and interrupt forwarding. In contrast to other VMMs, the virtualization of all other resources is moved outside the hypervisor. Such original approach has the following advantages:

- Device drivers are not limited to the hypervisor operating system because they are installed on a virtual machine (any OS),

 ISBN: 978-1-61208-153-3

- Device drivers, as the most vulnerable software, are isolated from the hypervisor, significantly increasing the stability,
- Distributed virtualization of resources allows creation of several driver domains, eliminating the single point of failure,
- Small hypervisor operating system is much more reliable, efficient and stable.

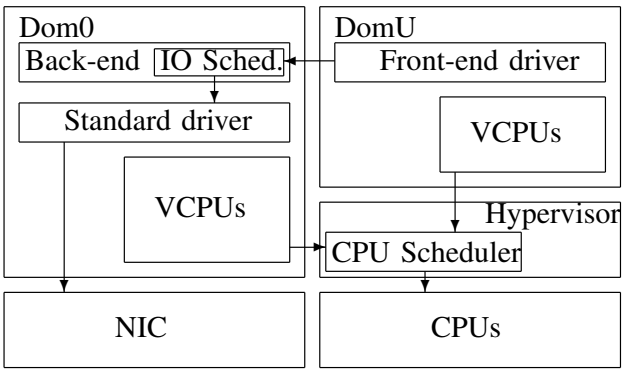


Figure 1. Xen architecture (Dom0 - Xen primary virtual machine, DomU - other Xen virtual machine, Hypervisor - main Xen operating system running directly on hardware, NIC - Network Interface Card, VCPU - virtual CPU)

There are two main virtualization methods. The first one allows to run any kind of OS and emulates all the necessary hardware to create an impression that the guest system is running on a physical machine. Second approach is to run a modified guest operating system, which is ”aware” of being virtualized. The latter, called *paravirtualization*, is much more efficient, but limited to some operating systems only. Xen provides both methods, but performs much better in the paravirtualization mode, which will be the only method used further in this paper.

To make the IO operations as fast as possible, Xen introduced also paravirtualized device drivers. Each guest domain (Xen VMs are also called ”domains”) has the front-end drivers installed. Such drivers, provided with Xen, are communicating with the back-end drivers running on a special driver domain (Dom0 in Fig. 1). All requests addressed to a certain hardware are first scheduled and processed by the back-end driver, then are sent to the standard device driver inside the driver domain and finally reach the hardware. Thanks to Xen internal page-flipping mechanism called XenBus, (see [2], [3]), such solution is much more efficient than the standard emulation technique.

IV. XEN SCHEDULERS

The main goal of this study is to examine the network performance isolation across Xen guest domains. It means to check, if activity of one virtual machine influences the network performance of any other. The resulting knowledge is of great importance from the perspective of many network-related applications.

There are two elements in Xen, which may influence such isolation, namely the CPU scheduler and the network IO scheduler [6]. In the following two sections a description of these two schedulers is given.

A. CPU Scheduler

The fundamental part of each multitasking operating system is the CPU scheduler. Its aim is to create an impression that all running processes are executed in parallel. Typically, there are much more processes than available physical CPUs and the processes have to share CPU time. The scheduler is responsible for this division.

Inside Xen VMM, the hypervisor is the main operating system running on the physical machine. It is responsible for scheduling physical CPU time among virtual machines. To make the process easier the term *virtual CPU (VCPU)* is introduced. Every VM in Xen can have multiple virtual processors. Also, every domain is running operating system with another scheduler, which divides a VCPU time among processes running inside the guest operating system. The hypervisor on the other hand, schedules the physical CPU time among VCPUs.

The newest version of Xen uses the *credit scheduler* [4] [5]. It assigns two parameters for each domain - *weight* and *cap*. The weight defines how much CPU time a domain gets comparing to other virtual machines. The cap parameter is optional and describes the maximum amount of CPU a domain can consume. Using this two parameters the number of credits can be calculated. As a VCPU runs, it consumes credits. While VCPU has existing credits, its priority is called *under* and it gets CPU time normally. When there are no credits left, the priority changes to *over*. Each physical CPU maintains its own local VCPU queue. In the first place, the VCPU tasks with priority *under* from the local queue are executed. Then, if there are no VCPUs with priority *under*, the scheduler looks for such tasks in other CPU queues. If there are no tasks with priority *under*, the tasks with priority *over* from the local queue are executed. The credit scheduler in Xen can by summarized in the following algorithm and diagram (Fig. 2):

1) Process preemption - the scheduler takes control over CPU.
2) Last taken VCPU inserted back into the local queue according to its credits number.
3) Have the highest priority VCPU from the local queue used all its credits?
   - No: Highest priority VCPU taken from the local queue.
   - Yes: SMP Load Balancing - highest priority VCPU taken from other CPU queues.
4) Switching context to the currently taken VCPU - the VCPU takes control over CPU.

 ISBN: 978-1-61208-153-3

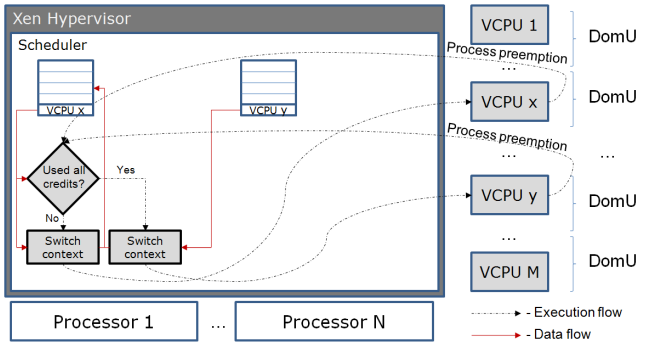


Figure 2. Xen CPU scheduler

Considering this CPU scheduler in the context of the network performance isolation, it is worth noticing that the scheduler operates on virtual CPUs only, so it should not have a strong impact on IO performance. However, it may happen that one misbehaving VM will slow down the total responsiveness and performance of other domains. Also, as it was presented in [7], the Xen CPU scheduler does not take into account the amount of CPU consumed by the driver domain on behalf of other VM. This may also have an impact on the network performance isolation, as some domains may use more CPU time than they are allowed. Furthermore, a different type of IO request (e.g., more demanding, like disk driver requests) can potentially slow down the driver domain and affect the network performance of other VMs.

B. Network IO scheduler

Looking at Xen architecture and analyzing its source code from the network performance isolation point of view, one can easily note that the most interesting part is the back-end network driver, called Netback. It contains another scheduler, responsible for gathering all IO requests sent to a certain physical network adapter. This network scheduler is not a complex mechanism and probably can be improved. Its only configuration parameter is the maximum rate (parameter *rate*) - in fact it can be perceived as the credits number in the scheduler. The administrator can specify only the maximal throughput achieved by a certain virtual network adapter. Unfortunately, there is no way to prioritize and control the quality of service in more details.

The scheduler itself counts the amount of data sent/received in given periods. If *rate* has been reached, it sets a callback to process the request in next periods. Such solution is efficient, but does not guarantee any fair share or quality. In fact, a misbehaving VM can theoretically flood driver domain with requests.

V. Experimental setup

To perform the tests, we installed Linux Gentoo with Xen 4.0.0 on Intel Quad Core 2 (2.83GHz), 2GB RAM, with hardware virtualization support. Two guest domains, each having 1 VCPU and 1GB of RAM, were created. Although there were separate physical CPU available for each VM, both VCPUs were pinned to the same physical CPU. Such configuration was used in order to check the influence of the CPU scheduler on the network performance. All network measurements were taken using *iperf* application. The UDP protocol transferring datagrams of 1500B to an external host over 100Mb link was used. We used the 100Mb link instead of 1Gb to present that the isolation problems are still present without a heavy CPU utilization. Only outgoing traffic was measured, as this was our main point of interest. The testing environment is presented in Fig. 3.

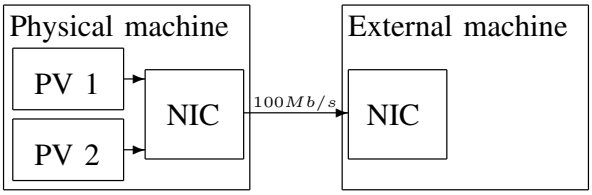


Figure 3. Testbed configuration. (PV1, PV2 - Xen paravirtualized machines, NIC - Network Card Interface)

VI. Results

In the first experiment, we observed how activity of one VM can affect the performance of another, when both VMs are configured with the same *rate* parameter. Four values of *rate* were used in different test runs: $25Mb/s$, $30Mb/s$, $35Mb/s$ and $40Mb/s$. In every run one machine started its transfer at the very beginning and the other started after 5s of delay. For every *rate* value, the experiment was repeated 10 times and the 0.95 confidence intervals were derived. The results are presented in Fig. 4.

Firstly, we can see that the actual rate is always a little smaller than *rate* parameter. As for the performance isolation, it is not too bad for low values of *rate*. However, with growing *rate*, the confidence intervals are getting larger and larger - in sample runs we can observe stronger variations of the throughput achieved by each VM. For the value of *rate* equal to 35Mb/s, the performance isolation becomes rather weak (although only about 60 percent of the total bandwidth is consumed).

Thus the only way to achieve a good isolation is to limit virtual adapters by far, which is not a satisfactory solution. Also, it is worth mentioning that having only the upper limit parameter is not enough in many cases. It would be much better to have any means to prioritize certain virtual adapter or even to have a minimum rate parameter and a scheduler satisfying these requirements.

In the second experiment, different *rate* values per each VM were used. Fig. 5 shows results for $rate = 30Mb/s$ in one VM, and $rate = 40Mb/s$ in another. The isolation

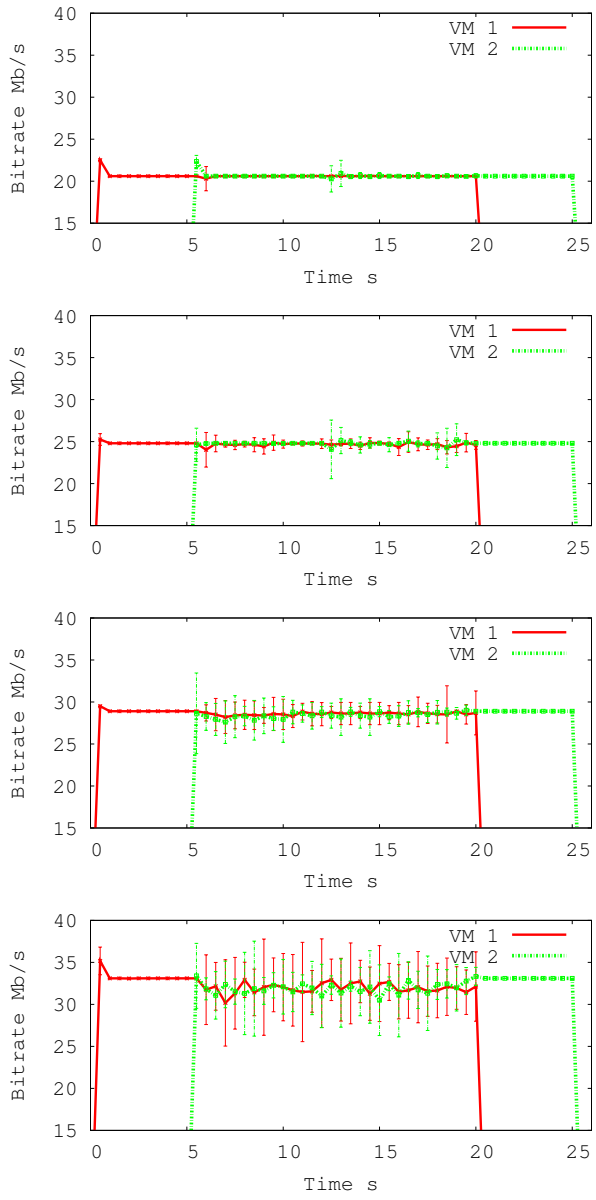


Figure 4. The throughput per VM for different values of *rate* parameter, namely for $25Mb/s$, $30Mb/s$, $35Mb/s$ and $40Mb/s$, counting from the top.

problem still remains but, what is worth noticing, both VMs affects each other similarly.

In the presented two experiments the performance isolation problem was either mild or moderate, depending on the configuration. In the following two experiments, we will demonstrate more severe performance isolation issues.

In the third experiment, we verified how Xen divides available bandwidth among two VMs when the maximal rate is not set. A sample path of the throughput achieved by each VM in time is presented in Fig. 6. Surprisingly, sometimes one virtual machine gets the total throughput and the other's

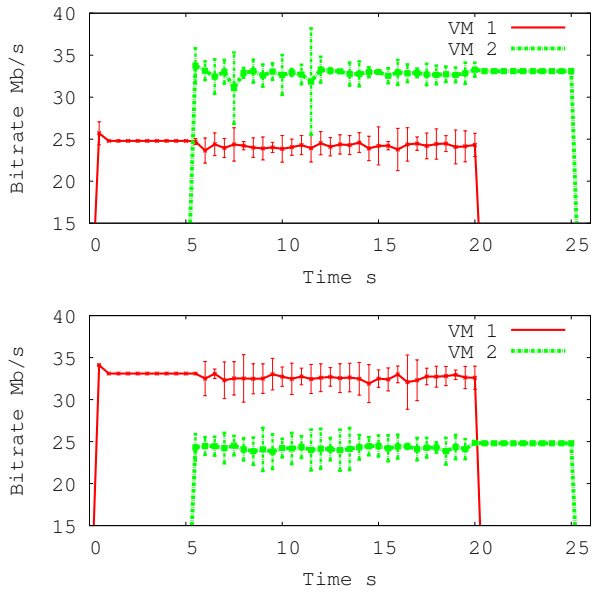


Figure 5. Total throughput per VM for different values of *rate* parameter ($30Mb/s$ and $40Mb/s$.

throughput decreases to 0. Moreover, there are long periods when one VM dominates the other by far. Therefore, we have in fact no performance isolation at all in this case.

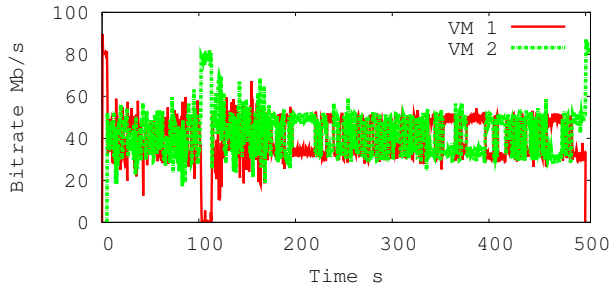


Figure 6. Sample throughput processes in time for two separate VMs without limits

In the fourth set of tests, we wanted to verify if a very abusive virtual machine can take more bandwidth then others. This time we wanted to check the performance isolation of the network IO scheduler only, therefore we pinned one physical CPU to each VM.

In the first test, one domain was trying to transfer data over one connection using full available speed, while the second domain was using two connections, both of them trying to achieve full available speed. In the next test, the second domain was using three connections at full available speed.

The results are presented in Fig. 7. As it can be observed, the more abusive domain is, the better throughput it achieves. Naturally, if the rate parameter had been set, the overactive

 ISBN: 978-1-61208-153-3

domain would never have crossed the maximum rate. In the lower ranges however, the problem remains.

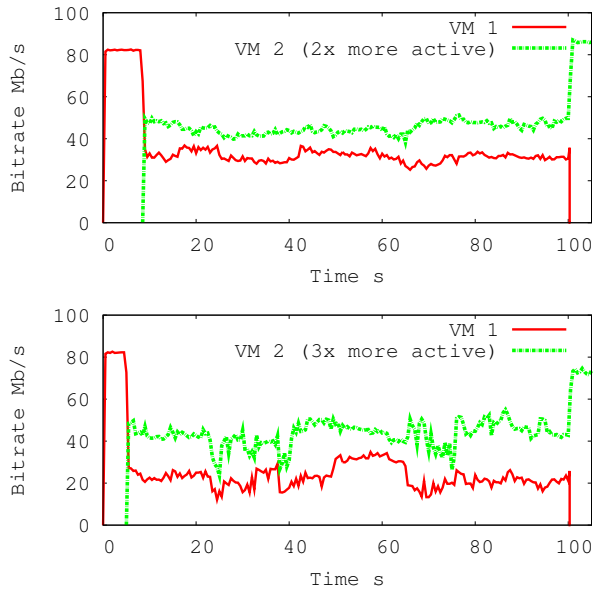


Figure 7. Bandwidth division with one overactive VM.

In the last experiment, we wanted to check if non-network IO requests can influence the network performance isolation of another domain. During the experiment one VM was constantly sending datagrams at full speed, while the second VM was performing some extensive disk operations (*fio* tool was used for this purpose). The results are presented in Fig. 8; t_0 and t_1 are points in time when the extensive disk operations were initiated and finished, respectively.

We can see that other IO request can also have a strong impact on the network performance. This is probably caused by driver domain not being able to process all the IO requests. Block device access is being handled by separate block device back-end drivers. Disk operations are much more demanding in the driver domain than the Netback drivers.

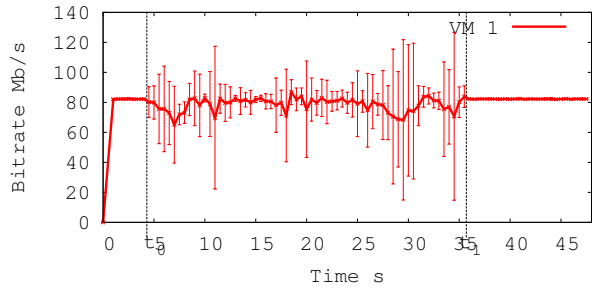


Figure 8. Disk IO influence on network performance. (t_0 - disk IO start, t_1 - disk IO finish)

VII. Improvement idea

After detailed analysis of the problem, we have gathered some ideas on how to modify Xen to improve the network performance isolation. Currently, in the driver domain several Netback kernel threads can be running, depending on the number of VCPUs. Furthermore, several virtual network adapters are mapped with one Netback kernel thread dynamically and this single Netback thread schedules the work using a simple round-robin algorithm, additionally taking into account *rate* parameter (omitting adapters, which used up all their bandwidth in the current period). Our idea is to introduce two additional parameters for every virtual adapter, namely *priority* and *min rate*. To implement the former, it would be necessary to change the round-robin mechanism to a more advanced priority based queue. Of course, we have to remember that the algorithm should not increase significantly the time complexity. The *min rate* parameter could use the same prioritization mechanism, assigning higher priorities to interfaces, which have not yet achieved the minimum rate. Depending on the results, it may be also necessary to introduce a user level application for maintaining the niceness level of each Netback thread inside the driver domain, according to actual needs.

A. Prioritization

The very first step to solve all the aforementioned problems is to introduce a prioritization mechanism into Xen's *Netback* driver. To achieve such functionality we implemented the simple *Weighted Round Robin* algorithm (see [8]). In virtualized environment where a packet passes several virtual adapters before it reaches the actual real interface and each interface has its own input buffer the WRR scheduler has to be modified to guarantee that the scheduled packets will not be dropped before they reach the wire. Dynamic and real-time priority assignment in this scheduler was created by additional Linux kernel *sysctl* parameters, i.e., *prioritize, priorities* and *delay*. The first parameter defines whether to use the *WRR* scheduler or not. Second parameter is an array of the actual priority values for each virtual adapter and the *delay* is used to define the inactivity period (i.e. a period of time after, which the *vif* is treated as inactive).

Each *vif* has a separate queue of data to transfer and a *priority*. The latter corresponds to the weight in the implemented *WRR* algorithm. Total bandwidth available at the physical link is shared proportionally between all active virtual interfaces according to their weights.

To test the prioritization we performed simple experiment where two VMs transmit data to an external host. In the meantime the priorities were changed every second. At the begging VM 1 had much bigger priority, in the end VM 2 was favored in the same proportion (i.e., $30/1$). The results are presented in Figure 9.

 ISBN: 978-1-61208-153-3

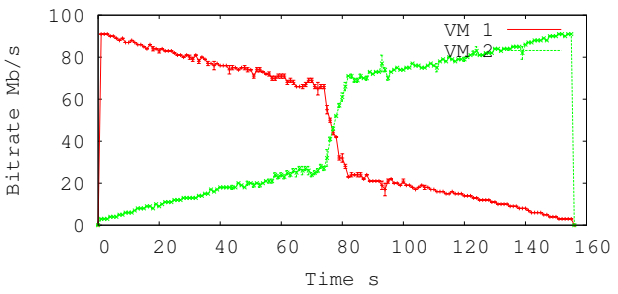


Figure 9. Results of the improved scheduler for changing priorities of each VM.

B. Further improvements

Prioritization brings a lot of new possibilities and improves the performance isolation by far. Nevertheless, in high CPU utilization scenarios it is not sufficient. Much more complicated mechanisms have to be created. Virtualization makes the problem very complex, as three different schedulers may affect the isolation: *CPU Scheduler, Domain 0 VCPU Scheduler* and *Netback IO Scheduler*. To achieve best results it might be necessary to *synchronize* all schedulers. Thus, partial solutions providing the *minimal rate* parameter for given virtual interface may prove very valuable. Finally, a modification proposed in [7] may also help to increase the performance isolation taking the aggregate CPU consumption into consideration. All these are subjects of our future study.

VIII. Conclusion

Xen is a powerful and stable virtualization platform, what accompanied with its Open Source formula makes it one of the most interesting VMMs, especially for research purposes. However, when the network virtualization is considered, the weak point of Xen is its lack of proper performance isolation. We demonstrated this using five sets of tests. The problems with isolation are caused by several factors mostly connected with CPU and IO schedulers. We proposed the *Netback* driver modification using *WRR* algorithm to provide prioritization. We have also briefly presented an idea for future improvements.

IX. Acknowledgment

This work is partially funded by the European Union, European Funds 2007-2013, under contract number POIG.01.01.02-00-045/09-00 ”Future Internet Engineering”.

References

[1] 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: Proc.of the 19th ACM SOSP, New York, 2003, Vol. 37, pp. 164–177.

[2] Y. Xia, Y. Niu, Y. Zheng, N. Jia, C. Yang, and X. Cheng: Analysis and Enhancement for Interactive-Oriented Virtual Machine Scheduling, IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008, Vol. 2, pp. 393–398.

[3] Xen Wiki, http://wiki.xensource.com/xenwiki/XenBus, 29-06-2011.

[4] L. Cherkasova, D. Gupta, and A. Vahdat: Comparison of the three CPU schedulers in Xen, SIGMETRICS Performance Evaluation Review; September 2007, Vol. 35, No. 2., pp. 42-51.

[5] G. W. Dunlap: Scheduler development update, Xen Summit North America 2010, http://www.xen.org/files/xensummit_intel09/George_Dunlap.pdf, 29-06-2011.

[6] J. Matthews, E.M. Dow, T. Deshane, W. Hu, J. Bongio, P.F. Wilbur, and B. Johnson: Running Xen: A Hands-on Guide to the Art of Virtualization; Prentice Hall; April 2008.

[7] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat: Enforcing Performance Isolation Across Virtual Machines in Xen; In Proceedings of the 7th ACM/IFIP/USENIX Middleware Conference, 2006, pp. 342-362.

[8] A. K. Parekh and R. G. Gallager: A generalized processor sharing approach to flow control in integrated services networks: The single-node case; IEEE/ACM Transactions on Networking; 1993, Vol. 1, pp. 344-357.

[9] P. Padala et al.: Adaptive control of virtualized resources in utility computing environments, ACM SIGOPS Operating Systems Review, Vol. 41, No. 3, 2007, pp. 289-302.

[10] Y. Song, Y. Sun, H. Wang, and X. Song: An adaptive resource flowing scheme amongst VMs in a VM-based utility computing, in Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on, 2007, pp. 10531058.

[11] J. Liu, W. Huang, B. Abali, and D. K. Panda: High performance VMM-bypass I/O in virtual machines, in Proceedings of the annual conference on USENIX, 2006, Vol. 6, pp. 3-3.

[12] V. Chadha, R. Illiikkal, R. Iyer, J. Moses, D. Newell, and R. J. Figueiredo: I/O processing in a virtualized platform: a simulation-driven approach, in Proceedings of the 3rd international conference on Virtual execution environments, 2007, pp. 116-125.

[13] D. Ongaro, A. L. Cox, and S. Rixner: Scheduling I/O in virtual machine monitors, in Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, 2008, pp. 1-10.

[14] G. Liao, D. Guo, L. Bhuyan, and S. R. King: Software techniques to improve virtualized I/O performance on multi-core systems, in Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, San Jose, California, 2008, pp. 161-170.

[15] S. R. Seelam and P. J. Teller: Virtual I/O scheduler: a scheduler of schedulers for performance virtualization, in Proceedings of the 3rd international conference on Virtual execution environments, 2007, pp. 105-115.

 ISBN: 978-1-61208-153-3