ICAS 2014

The Tenth International Conference on Autonomic and Autonomous Systems

ISBN: 978-1-61208-331-5

April 20 - 24, 2014

Chamonix, France

ICAS 2014 Editors

Mark J. Balas, Embry-Riddle Aeronautical University in Daytona Beach, USA

Wendy Powley, Queen's University - Kingston, Canada

ICAS 2014

Foreword

The Tenth International Conference on Autonomic and Autonomous Systems (ICAS 2014), held between April 20 - 24, 2014 in Chamonix, France, was a multi-track event covering related topics on theory and practice on systems automation, autonomous systems and autonomic computing.

The main tracks referred to the general concepts of systems automation, and methodologies and techniques for designing, implementing and deploying autonomous systems. The next tracks developed around design and deployment of context-aware networks, services and applications, and the design and management of self-behavioral networks and services. We also considered monitoring, control, and management of autonomous self-aware and context-aware systems and topics dedicated to specific autonomous entities, namely, satellite systems, nomadic code systems, mobile networks, and robots. It has been recognized that modeling (in all forms this activity is known) is the fundamental for autonomous subsystems, as both managed and management entities must communicate and understand each other. Small-scale and large-scale virtualization and model-driven architecture, as well as management challenges in such architectures are considered. Autonomic features and autonomy requires a fundamental theory behind and solid control mechanisms. These topics gave credit to specific advanced practical and theoretical aspects that allow subsystem to expose complex behavior. We aimed to expose specific advancements on theory and tool in supporting advanced autonomous systems. Domain case studies (policy, mobility, survivability, privacy, etc.) and specific technology (wireless, wireline, optical, e-commerce, banking, etc.) case studies were targeted. A special track on mobile environments was indented to cover examples and aspects from mobile systems, networks, codes, and robotics.

Pervasive services and mobile computing are emerging as the next computing paradigm in which infrastructure and services are seamlessly available anywhere, anytime, and in any format. This move to a mobile and pervasive environment raises new opportunities and demands on the underlying systems. In particular, they need to be adaptive, self-adaptive, and context-aware.

Adaptive and self-management context-aware systems are difficult to create, they must be able to understand context information and dynamically change their behavior at runtime according to the context. Context information can include the user location, his preferences, his activities, the environmental conditions and the availability of computing and communication resources. Dynamic reconfiguration of the context-aware systems can generate inconsistencies as well as integrity problems, and combinatorial explosion of possible variants of these systems with a high degree of variability can introduce great complexity.

Traditionally, user interface design is a knowledge-intensive task complying with specific domains, yet being user friendly. Besides operational requirements, design recommendations refer to standards of the application domain or corporate guidelines.

Commonly, there is a set of general user interface guidelines; the challenge is due to a need for cross-team expertise. Required knowledge differs from one application domain to

another, and the core knowledge is subject to constant changes and to individual perception and skills.

Passive approaches allow designers to initiate the search for information in a knowledge-database to make accessible the design information for designers during the design process. Active approaches, e.g., constraints and critics, have been also developed and tested. These mechanisms deliver information (critics) or restrict the design space (constraints) actively, according to the rules and guidelines. Active and passive approaches are usually combined to capture a useful user interface design.

We take here the opportunity to warmly thank all the members of the ICAS 2014 Technical Program Committee, as well as the numerous reviewers. The creation of such a 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 ICAS 2014. 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 ICAS 2014 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICAS 2014 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the fields of autonomic and autonomous systems.

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

ICAS 2014 Chairs:

Michael Bauer, The University of Western Ontario - London, Canada
Radu Calinescu, University of York, UK
Michael Grottke, University of Erlangen-Nuremberg, Germany
Bruno Dillenseger, Orange Labs, France
Mark J. Balas, Embry-Riddle Aeronautical University, USA
Alex Galis, University College London, UK
Antonio Liotta, Eindhoven University of Technology, The Netherlands
Jacques Malenfant, Université Pierre et Marie Curie, France
Mark Perry, University of New England in Armidale, Australia
Wendy Powley, Queen's University - Kingston, Canada
Nikola Serbedzija, Fraunhofer FOKUS, Germany

ICAS 2014

Committee

ICAS Advisory Chairs

Michael Bauer, The University of Western Ontario - London, Canada
Radu Calinescu, University of York, UK
Michael Grottke, University of Erlangen-Nuremberg, Germany
Bruno Dillenseger, Orange Labs, France
Mark J. Balas, Embry-Riddle Aeronautical University, USA
Alex Galis, University College London, UK
Antonio Liotta, Eindhoven University of Technology, The Netherlands
Jacques Malenfant, Université Pierre et Marie Curie, France
Mark Perry, University of New England in Armidale, Australia
Wendy Powley, Queen's University - Kingston, Canada
Nikola Serbedzija, Fraunhofer FOKUS, Germany

ICAS 2014 Technical Program Committee

Jemal H. Abawajy, Deakin University, Australia
Sameh Abdel-Naby, University College Dublin, Ireland
António Abelha, Universidade do Minho - Braga, Portugal
Nouara Achour, USTHB University, Algeria
Carl Adams, University of Portsmouth, UK
Javier Alonso, Duke University, USA
Razvan Andonie, Central Washington University - Ellensburg, USA
Richard Anthony, University of Greenwich, UK
Eva Ibarrola Armendariz, Escuela Técnica Superior de Ingeniería de Bilbao, Spain
Ismailcem Budak Arpinar, University of Georgia - Athens, USA
Tsz-Chiu Au, Ulsan National Institute of Science and Technology (UNIST), Korea
Roger Azevedo, McGill University, Canada
Mark J. Balas, University of Wyoming - Laramie, USA
Michael Bauer, The University of Western Ontario -London, Canada
Matthias Becker, University Hannover, Germany
Janusz Bedkowski, Institute of Mathematical Machines / Warsaw University of Technology, Poland
Julita Bermejo-Alonso, Universidad Politécnica de Madrid, Spain
Karsten Berns, University of Kaiserslautern, Germany
Daniel Berrar, Tokyo Institute of Technology, Japan
Philippe Besnard, IRIT - CNRS /Universite Paul Sabatier - Toulouse, France
Ateet Bhalla, Oriental Institute of Science & Technology - Bhopal, India
Karsten Böhm, Fachhochschule Kufstein, Austria
Fabienne Boyer, University of Grenoble I, France
Stainam Brandao, COPPE/Federal University of Rio de Janeiro, Brazil
Ruth Breu, University of Innsbruck, Austria
Daniela Briola, University of Genova, Italy
David W Bustard, University of Ulster, UK

Radu Calinescu, University of York, UK
Paolo Campegiani, University of Roma Tor Vergata, Italy
Sara Casolari, Università di Modena e Reggio Emilia, Italy
Fernando Cerdan, Universidad Politecnica de Cartagena, Spain
Michal Certicky, Comenius University - Bratislava, Slovakia
Lei Chen, Sam Houston State University, USA
Stéphanie Combettes, Université Paul Sabatier, IRIT, France
Peng Dai, Google Inc., USA
Carlos Roberto de Oliveira Junior, Instituto Federal do Rio de Janeiro, Brazil
Noel de Palma, University Joseph Fourier, France
Marina De Vos, University of Bath, UK
Sotirios Ch. Diamantas, University of Nebraska, Omaha, USA
M. Bernardine Dias, Carnegie Mellon University, USA
Tadashi Dohi, Hiroshima University, Japan
Changyu Dong, University of Strathclyde, UK
Thach-Thao Duong, Griffith University / NICTA, Australia
Larbi Esmahi, Athabasca University, Canada
Thaddeus Eze, University of Greenwich - London, UK
Ziny Flikop, Scientist, USA
Naoki Fukuta, Shizuoka University, Japan
Alex Galis, University College London, UK
Rodrigo Garcia Carmona, Universidad Politecnica de Madrid (DIT-UPM), Spain
Fabio Gasparetti, Roma Tre University, Italy
Joseph Giampapa, Carnegie Mellon University, USA
Andrzej M. Goscinski, Deakin University - Geelong, Australia
Dominic Greenwood, Whitestein, Switzerland
William Grosky, University of Michigan - Dearborn, USA
Michael Grottke, University of Erlangen-Nuremberg, Germany
Jordi Guitart, Universitat Politècnica de Catalunya - Barcelona Tech (UPC), Spain
Odd Erik Gundersen, Verdande Technology, Norway
Mustapha Hamerlain, CDTA, Algeria
Ronny Hartanto, German Research Center for Artificial Intelligence (DFKI GmbH), Germany
Takako Hashimoto, Chiba University of Commerce, Japan
Koen Hindriks, Delft University of Technology, Netherlands
Gerold Hoelzl, Johannes Kepler University, Austria
Wladyslaw Homenda, Warsaw University of Technology, Poland
Marc-Philippe Huget, Polytech Annecy-Chambery-LISTIC | University of Savoie, France
Tim Hussein, University of Duisburg-Essen, Germany
Yoshiro Imai, Kagawa University, Japan
Luis Iribarne, University of Almeria, Spain
Yiming Ji, University of South Carolina - Beaufort, USA
Richard Jiang, Northumbria University, UK
Yanguo Jing, London Metropolitan University, UK
María José Ibáñez, RIAM I+L Lab (GNOSS), Spain
Hamed Ketabdar, Deutsche Telekom Laboratories / TU Berlin, Germany
Fernando Koch, IBM Research Lab, Brazil
Boris Kovalerchuk, Central Washington University - Ellensburg, USA
Satoshi Kurihara, University of Osaka, Japan

Munehiko Sasajima, Osaka University, Japan
Mariano Saura, Polytechnic University of Cartagena, Spain
Christoph Schommer, University Luxemburg, Luxemburg
Paulo Jorge Sequeira Gonçalves, Polytechnic Institute of Castelo Branco, Portugal
Nikola Serbedzija, Fraunhofer FOKUS, Germany
Mohamed Shehab, University of North Carolina at Charlotte, USA
Maxim Shevertalov, Drexel University, USA
Arnab Sinha, INRIA, France
Flavio Soares Correa da Silva, University of Sao Paulo, Brazil
Nisheeth Srivastava, University of California, USA
Edward Stehle, Drexel University, USA
Claudius Stern, Opsolution GmbH, Germany
Ryszard Tadeusiewicz, AGH University of Science and Technology, Poland
Charalampos Tampitsikas, Università della Svizzera Italiana (University of Lugano), Switzerland
Lijun (Leo) Tang, Microsoft Corporation - Bellevue, USA
Yuqing Tang, Carnegie Mellon University, USA
Patricia Tedesco, Federal University of Pernambuco, Brazil
Ingo J. Timm, Universität Trier, Germany
Manachai Toahchoodee, The University of the Thai Chamber of Commerce, Bangkok, Thailand
Irina Topalova, Technical University of Sofia, Bulgaria
José Manuel Torres, University Fernando Pessoa - Porto, Portugal
Davide Tosi, Università dell'Insubria – Como, Italy
Trung Kien Tran, Institute of Artificial Intelligence - University of Ulm, Germany
Raquel Trillo Lado, University of Zaragoza, Spain
Egon L. van den Broek, University of Twente, Enschede, Karakter University Center, Radboud University Medical Center Nijmegen, Nijmegen, The Netherlands
Cristián Varas, NIC Chile Research Labs, Chile
Stavros Vassos, Sapienza University of Rome, Italy
Phan Cong Vinh, NTT University, Vietnam
Vladimir Vlassov, KTH Royal Institute of Technology, Sweden
Stefanos Vrochidis, Centre for Research and Technology Hellas - Thermi-Thessaloniki, Greece
Thomas Walsh, Massachusetts Institute of Technology, USA
Bozena Wozna-Szczesniak, Jan Dlugosz University, Institute of Mathematics and Computer Science – Czestochowa, Poland
Nanjian Wu, Chinese Academy of Sciences, China
Reuven Yagel, Ben-Gurion University, Israel
Jinfeng Yi, Michigan State University, USA
Constantin-Bala Zamfirescu, "Lucian Blaga" University of Sibiu, Romania
Andrzej Zbrzezny, Jan Dlugosz University in Czestochowa, Poland
Chenyi Zhang, Simon Fraser University, Canada
Dieter Zöbel, University Koblenz-Landau, Germany
Albert Zomaya, University of Sydney, Australia

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

MetaMAc, or What Do I Do Now?

A strategic perspective on autonomy beyond anomalies and goals

Don Perlis
Department of Computer Science
University of Maryland
College Park, MD, USA
perlis@cs.umd.edu

Michael T. Cox
Institute for Advanced Computer Studies
University of Maryland
College Park, MD, USA
mcox@cs.umd.edu

***Abstract*—In recent years, there has been strong interest in both reasoning about goal-identification and selection and metacognitive handling of anomalous situations. These two concerns are usually framed in terms of making agents more autonomous and flexible in dynamic and complex domains. Here, we wish to argue that there is a natural unifying perspective that includes both concerns and that may point the way to a yet more powerful kind of autonomy.**

Keywords—high-level autonomy; rational anomaly handling; goal reasoning; anomaly handling; monitoring, control, and management of autonomous self-aware systems

I. INTRODUCTION

An agent often has routine activities in which it is forming and/or following plans in pursuit of existing goals. And there are also situations in which it has to stop and ask itself: what do I do now? One major example of the latter is that of anomaly-handling: something seems out of the ordinary, contrary to expectation, and might indicate the need to do a form of error-correction. This has been the focus of much recent work, for instance Meta-AQUA [6], the Metacognitive Loop [2] and other similar efforts. Another example is goal-driven autonomy, in which an agent may autonomously alter or add to its goals if circumstances so warrant [1,8].

We wish to call attention to a level of processing at which an agent considers quite generally what to do: select from among several existing goals, form a plan to achieve an existing goal, continue with a current plan-in-action, alter such a plan, identify a new goal, abandon a plan or a goal, adopt new subgoals in response to unexpected events, explore opportunities for possible goals or other benefits, do a reality-check of beliefs and expectations, learn for learning's sake, and so on. This could perhaps be called the executive level of processing (borrowing that phrase from cognitive psychology), although that terminology already is in use in various cognitive architectures and so might not be the best choice. Instead, let us call this *reasoning at the strategic level*.

In what follows, we sketch some concepts related to the idea of such a processing level, argue that it usefully generalizes more traditional goal-reasoning and anomaly-handling, and outline what might be fruitful approaches to the strategic level. Section II simply states our hypothesis, and gives a key example; sections III –V describe existing work on rational anomaly-handling, goal-driven autonomy, and how they interact; VI discusses temporal issues, and VII presents conclusions.

II. METAMAC

We postulate a *metacognitive monitoring activity (MetaMAc)* that runs in parallel with an agent's normal routine activity of planning-acting in pursuit of already-identified goals. MetaMAc will be aware of such routine activities that are underway, and also of their aims and expectations, and of how (at least some) events are actually unfolding (which may or may not be as expected). As such MetaMAc represents the deliberate, conscious "self as process" monitoring and considering itself [3]. As MetaMAc processes this real-time information, it also asks itself over and over: What should I do now? What choices are there? Is there anything that would be better to do instead of (or in addition to) what I am doing? MetaMAc would normally run in the background, unless something pops into prominence in virtue of a certain salience or threshold that is reached.

We hypothesize that a MetaMac-enabled system would reveal considerable advantages over the same system without that enhancement. This could show up in many ways, but most especially in fewer errors over the long run.

Here is an example that illustrates various aspects of our idea:

A meeting is occurring, but participants are finding it hard to hear one another. One automated participant, X, identifies the problem as background noise coming from the hallway. X gets up and closes the door. Why? Because the closed door effectively blocks the hallway noise.

 ISBN: 978-1-61208-331-5

How does X come to be able to do this? While it may seem trivial, there are in fact a number of specific capabilities involved here, that illustrate our thesis:

1. X can identify a new problem on its own. This in itself is non-trivial; typically a human presents a problem to an automated system.

2. X can reason about causal relationships; but where does the causal info come from? (see item 4 below)

3. X can form new goals (problem solution ideas) on its own; but often there are multiple relevant goals, so how are they distinguished so as to pick the "best" ones (whatever that may mean)? Notice this, aside from closing the door, X could move closer to the speaker, ask the speaker to raise the volume, tape the mouths of the passersby, etc. So cost/benefit analysis is important.

4. X can and does pick up new information on a "knowledge is power" basis, independent of a specific immediate need. That is how X learned (long ago) that sound does not travel well through a closed door. But this presents complications as well: how does X decide when and what and how much to learn about "things in general?" There is an endless supply of such things, and X could easily become permanently absorbed in learning one narrow theme, or learning tiny random bits of distinct themes. Some sort of overall principles are needed here, perhaps in part guided by a concern to identify causal links.

III. RATIONAL ANOMALY HANDLING

No matter how complete an agent's knowledge base is or how good the agent is engineered, eventually mistakes or anomalies occur. In the face of surprise, an agent should be able to manage to adapt and do something reasonable. We call this capability *rational anomaly-handling (RAH)*. RAH is characterized by the following features [10]:

(i) We have expectations as to how things will be.
(ii) We compare expectation to observation and thereby note indications that an expectation has been violated.
(iii) We assess what we know that might explain this violation.
(iv) We decide what response, if any, to guide into place.
(v) We revise/create expectations as needed.

For the most part, artificial intelligence (AI) seems to be missing this key ingredient. After many decades of very fruitful work in machine learning, automated reasoning, planning, vision, natural language, and so on, we still do not have systems that come anywhere close to human-level performance of a general sort.

IV. GOAL-DRIVEN AUTONOMY

The idea of a goal tends to be conceptualized in two quite distinct ways in AI: as an end-state to be achieved, and as a kind of action to perform. In ordinary language, we conflate these, as in "I want to go to the beach," when we (presumably) mean "I want to be at the beach," which is a state-goal. Yet we also have maintenance goals, such as "keeping the room picked up" which presumably means vigilantly acting on any upcoming needs to do a picking-up action. In addition, there are intentions that are goal-like but perhaps not best described as goals. For instance, we can seek knowledge, in the belief that knowledge is a good thing to have; but no particular piece of knowledge can be identified as *the* goal here. Another is that of identifying what to do, if not particular goal is at hand; here the goal *could* be characterized as *find a goal*, but that begs the question in a way. Possible goals abound, all the time, so selecting one – if not at random – is a highly unspecified task, and could perhaps be driven by some agent-oriented measure of utility or interest.

The model we advocate - called *goal-driven autonomy (GDA)* [4] - casts agents as independent actors that can recognize problems on their own and act accordingly. Furthermore in this goal-reasoning model, goals are dynamic and malleable and as such arise in three cases: (1) goals can be subject to transformation and abandonment (2) they can arise from subgoaling on unsatisfied preconditions or in response to impasses during problem-solving and planning; and (3) they can be generated from scratch during interpretation.

For our purposes here, the most important of the above three cases is the third one. The idea is that given a problem in the world, an autonomous cognitive system must distinguish between perturbations that require a change in plans for the old goal and those that require a new goal altogether. What is missing in the planning and agent communities is a recognition that autonomy is not just planning, acting and perceiving. It also must incorporate a first-class reasoning mechanism that interprets and comprehends the world as plans are executed. It is this comprehension process that not only perceives actions and events in the world, but can recognize threats to current plans, goals, and intentions. We claim that a balanced integration between planning and comprehension leads to agents that are more sensitive to surprise in the environment and more flexible in their responses.

V. RAH AND GDA

RAH and GDA are closely related. RAH can lead to the conclusion that goals need to be altered or invented, for instance to avoid a repeat of a past anomalous situation (such as something identified as having prevented a goal from being achieved). This would then lead to invocation of a GDA process. And GDA in its own right can lead to the uncovering

 ISBN: 978-1-61208-331-5

of anomalies, such as goals not being met (which can trigger the application of RAH).

One way to envision this is in terms of the *A-distance* [7] which assesses alterations in time-series data that exceed a given threshold. This is a crucial kind of hedging-factor. For any given set of expectations will almost certainly fail to be fully identical to observed events. Tiny variations are the norm, and one cannot possibly attend to all of them (nor would it make sense to do so if it were possible). Yet how can suitable thresholds be determined, when context means everything? In some contexts, a small variation in color or noise-level may be insignificant, and in others may flag major problems or opportunities.

We think that learning is a promising approach here: an agent can learn, for a given context in which it may be operating (or planning to operate in), which are the important things to attend to. This can be partly at the explicit symbolic level (e.g., a teacher can tell the agent some items to watch for and some to ignore) and partly subsymbolic (experience can provide ranges of "normalcy" that the agent trains into its routines). While A-distance was developed largely for the latter situation, with continuous real-valued data, recent work has shown that it also is effective for discrete symbolic data [5].

Now, we doubt A-distance alone will be enough to cover all the cases that we envision for MetaMAc. For instance, another agent might simply tell our agent that something is important. That is unlikely to cross an A-distance threshold, since conversations may go on all the time, with words flowing rapidly back and forth. It would presumably require a rather high-level reasoning process to understand language sufficiently well to distinguish in a general principled way between, say, "such matters are unimportant" and "don't ever assume that such matters are unimportant."

The strategic level of MetaMAc then will likely require fairly sophisticated knowledge representation and reasoning (KRR) techniques; requiring reasoning about natural-language processing and the world more generally will at times be essential. Reference [9] gives compelling examples (although it is not focused on high-level strategic reasoning); see also [2] where three KRR criteria are given that flexible system should satisfy.

Thus, we suggest that the strategic level intended for MetaMAc involves an organizer of cognition that integrates activities and seeks to improve their effectiveness. But it also can involve entirely new concerns, such as the fortuitous discovery of a new homeostatic equilibrium in which things "work well" in totally unsuspected ways. This latter would not be an existing goal, nor one arrived at as something to achieve, but rather stumbled on and then perhaps adopted as a maintenance goal. The field of developmental (aka epigenetic) robotics would seem to fall into this category.

VI. Time and Other Realities

All processing takes time; and this applies to MetaMac as well. An issue immediately arises: how can MetaMac keep up with real-world changes (in its associated agent and in the world more generally)? A truly autonomous or autonomic system will not have the luxury of a human-in-the-loop, or of other activity being interrupted so that repairs can be made at leisure [11]. However, a special-purpose real-time logic is available [2], that was designed for such situations. Indeed, one of our aims in this research is to combine active logic with the ideas of MetaMac.

Indeed, the MetaMac idea is in effect a combination of two approaches that we have been exploring and implementing in recent years, namely the RAH and GDA themes. See [4,10] for details.

VII. Conclusion

In this paper, we suggest that an underlying research issue exists of considerable potential for enhanced autonomy: *how to design an agent with an effective and general-purpose "what do I do now" capacity*. This capacity bears on many cognitive processes and seems crucial for high-level reasoning in complex ever-changing environments. Researchers have at times studied aspects of what we describe under the MetaMAc banner in terms of metacognition (e.g., RAH). Other researchers have examined some of these issues in terms of goal reasoning and goal-driven autonomy. It may be the case that we are all speaking of the same process.

Acknowledgment

This material is based upon work supported by ONR Grants # N00014-12-1-0430 and # N00014-12-1-0172 and by ARO Grant # W911NF-12-1-0471. We thank the anonymous reviews for their comments and suggestions.

References

[1] D. Aha, M. Klenk, H. Muñoz-Avila, A. Ram, and D. Shapiro (Eds.) "Goal-directed autonomy: Papers from the AAAI workshop." Menlo Park, CA: AAAI Press, 2010.

[2] M. Anderson and D. Perlis, "Logic, self-awareness and self-improvement." Journal of Logic and Computation 15, pp. 21–40, 2005.

[3] J. Brody, M. T. Cox, and D. Perlis, "The processual self as cognitive unifier," Proceedings of the Annual Meeting of the International Association for Computing and Philosophy. 2013.

 ISBN: 978-1-61208-331-5

[4] M. T. Cox, “Question-based problem recognition and goal-driven autonomy,” D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), Goal Reasoning: Papers from the ACS workshop Tech. Rep. No. CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science. pp. 10-25, 2013.

[5] M. T. Cox, T. Oates, M. Paisner, and D. Perlis, “Detecting change in diverse symbolic worlds,” L. Correia, L. P. Reis, L. M. Gomes, H. Guerra, & P. Cardoso (Eds.), Advances in Artificial Intelligence, 16th Portuguese Conference on Artificial Intelligence, University of the Azores, Portugal: CMATI, pp. 179-190, 2013.

[6] M. T. Cox, and A. Ram, “Introspective multistrategy learning: On the construction of learning strategies.” Artificial Intelligence, vol. 112, pp. 1-55, 1999.

[7] D. Kifer, S. Ben-David, and J. Gehrke, “Detecting change in data streams.” Proceedings of the Thirtieth Very Large Databases Conference, pp. 180-191, 2004.

[8] M. Klenk, M. Molineaux, and D. W. Aha, “Goal-driven autonomy for responding to unexpected events in strategy simulations.” Computational Intelligence, vol. 29, no. 2, pp. 187–206, 2013.

[9] H. Levesque, “On our best behaviour.” IJCAI Research Excellence Award Presentation at the 23rd International Joint Conference on Artificial Intelligence, IJCAI-13, (Beijing), 2013.

[10] D. Perlis, “BICA and beyond: How biology and anomalies together contribute to flexible cognition.” Intl J of Machine Consciousness, v2, n2, pp. 1-11, 2010.

[11] D.Perlis, J. Elgot-Drapkin, and M. Miller, “Stop the world – I want to think”. International Journal of Intelligent Systems, 6, pp. 443-456, 1991. Special issue on temporal reasoning.

 ISBN: 978-1-61208-331-5

Unmanned Aerial Vehicles & Service-Oriented Architecture: LARISSA and Knowledge Based Framework's First Study

Emerson A. Marconato, Douglas Rodrigues,
Arthur A. Chaves, Kalinka R. L. J. C. Branco
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, Brazil
Email: {emerson, douglasr, kalinka}@icmc.usp.br
arthurac@usp.br

Rajiv Ramdhany, Geoff Coulson
School of Computing and Communications
Lancaster University
Lancaster, UK
Email: {r.ramdhany, g.coulson}@lancaster.ac.uk

***Abstract*—Embedded systems are computer systems that are part of a larger system, which generally provide real-time monitoring and control. They execute a predefined set of tasks on behalf of a real-time application, and may have special requirements based on the application domain they support. For instance, these systems are considered safety-critical embedded systems when failure may result in loss of life or high-value assets. UAVs (Unmanned Aerial Vehicles) constitute a typical application of a complex critical embedded system. One concept that can result in radically different solutions in UAVs is the use of Service-Oriented Architecture (SOA) based on standard reference model architecture. The increasing use of SOA in critical applications demands dependable and cost-effective techniques to ensure high security. In this paper we developed different kind of services for avionics with different parameters (security, reliability and performance) to provide use of SOA in a less critical part in the whole systems. Both LARISSA (Layered Architecture Model Interconnection Systems in UAV) and KBF (Knowledge Based Framework for Dynamically Changing Applications) presented in this paper can give more intelligence to UAVs and provide a new way of segregating the UAV mission from the vehicle itself. Some services were developed and a performance evaluation was conducted showing the benefits in choosing some determined services.**

Keywords*—*Critical embedded systems, UAVs, SOA, Web services, performance.

I. INTRODUCTION

Embedded systems are computer systems that are part of a larger system. These systems provide, in most cases, real-time monitoring and control. They are considered safety-critical when possible failure may result in loss of life or high-value assets [1][2][3][4]. Both hardware and software in embedded systems have become increasingly complex. Multicore and multiprocessor systems have become common, which has further increased the complexity of software [5]. Moreover, both can be seen in homes and in business environments where they have been used for the control or management information.

UAV is a typical application of a critical embedded system. The term UAV was adopted by FAA (Federal Aviation Administration) and by the international academic community to describe a system that includes not only the aircraft, but also all the associated elements such as payload, ground control station and communication links [6]. UAVs have been widely used in precision agriculture, national security and environmental monitoring. Several papers have been published in this area, demonstrating the feasibility of using such vehicles as important tools for performing precision agriculture and environmental monitoring [7][8].

There are different types of UAVs that have different capabilities. Some aircraft can fly autonomously, following a pre-programed flight path (based on a grid or a sequence of waypoints) [7], while others can fly receiving commands from ground stations operated by pilots. The aircraft's size can range from micro to large, and the ground control station can be implemented on smartphones, tablets, laptops or networks of workstations (distributed control stations). Thus, the aircraft may vary not only in size, but also in shape and type of propulsion performance. The ground station interface can vary from a joystick to a tangible user interface (for example, a table with tangible augmented reality). The performance of the communication links and the payload type are also very important to fulfil the mission intended for the system. Specialized literature says that UAVs will become popular and will be part of airspace in the next 10 years, performing different missions, from agricultural border inspections to automatic cargo transport [9][10][11].

The UAVs' heterogeneity and constraints and the distinct nature of their interactions are challenges for their successful integration into architecture for a shared exploitation of UASs. The heterogeneity prevalent in UAVs in terms of services for avionics and architecture is particularly relevant to elaboration of multi-application missions. This heterogeneity in UAV services is often manifested as characteristics, such as reliability, security, and performance. Different service implementations typically offer different guarantees in terms of these characteristics and in terms of associated costs. The initial choice of a particular avionics service implementation can therefore become sub-optimal as long new applications/services are deployed, needing a careful selection of services to fulfil particular performance and operational guarantees and, subsequently, to avoid compromising the mission.

In the same way, architectures that enable the organization and more specific definition of the components of these embedded systems (UAVs) ease the development of hardware and software that compose them, allowing these vehicles be more easily inserted and incorporated in a non-segregate airspace. Therefore, the main goal of this preliminary research is propose a new architecture to UASs and investigate the degree of heterogeneity present in UAVs in terms of services, proposing architectural abstractions for the integration of these service variants. In particular, we explored the notion of Service-Oriented Architecture (SOA) in the context of UAVs as safety-critical embedded systems for the composition of services to fulfil the specified application performance and the dependability guarantees.

The rest of this paper is organized as follows. In Section II, we describe the related works in the field of SOA and Reference Model Architecture in embedded and safety-critical embedded systems. Section III shows the concepts of Reference Model Architecture

 ISBN: 978-1-61208-331-5

and the new trends using SOA in this kind of system. Section IV presents KBF (Knowledge Based Framework for Dynamically Changing Applications) and its functioning. Section V shows a case study of default, secure, and reliable services, and then analyses the results. Finally, Section VI presents our conclusions regarding this work, as well some future prospects for this research.

II. Related Work

This section presents a review of Reference Model Architecture and SOA (Service-Oriented Architecture) in embedded systems and critical embedded systems. The Reference Model Architecture found in the literature can be classified in Federated (traditional architectures) and Non-Federated (non-conventional architectures).

NIST (National Institute of Standards and Technologies) provides a reference model for UAVs [12]. In this particular pattern, the reference model was proposed to specify military rules, practices and controls in a comprehensible and intuitive way for a human commander. The proposed architecture approach is different from that required in our project, focusing on a lower level of abstraction, which the layers specify what components found in a UAV system should do. Then the authors introduced a hardware/software embedded architecture especially designed to operate as a UAV's payload and mission controller. The hardware architecture is built as a set of embedded microprocessors connected by a LAN (Local Area Network). Over this hardware infrastructure is implemented a software layer that allows each module to support multiple applications. Every application can create and sign services and these services can be dynamically discovered and consumed as in Internet domain.

The work proposed by Pastor et al. [13] is based on architecture of hardware and software designed to operate the mission controller and the payload in mini/micro UAVs. According to the authors, the innovation is the use of a distributed hardware architecture which is easily scalable by the use of LAN architecture-based software subscription services, communication abstraction layer and execution flow based on mission planning. Still according to the authors, the high level of modularity offered by a LAN provides flexibility for coupling the microprocessor type most appropriate to use the module, given its functional requirements.

Olson et al. [14] proposed another architecture model. This is Phase III of the project named MCAP (Manned/Unmanned Common Architecture Program), used by the U.S. Army in UAVs such as FCS (Future Combat Systems) and C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance). Phase III of MCAP architecture is based on electronics and commercial off the shelf (COTS) and open standards interfaces. The objective of the development of the model was define and develop an architecture capable of supporting a amount of UAVs in the U.S. Army platforms, demonstrating the performance of the resulting system in a laboratory environment. The development of this model relied on the study of three classes of UAS: Unmanned Combat Armed Rotorcraft (UCAR), an unmanned combat helicopter; Class IV Medium Altitude Long Endurance (MALE); and Extended Range/Multi-Purpose (ERMP), with two aircrafts: Fire Scout and Shadow 200.

The project presented by Neto et al. [15] is a modular embedded architecture, consisting of three levels: embedded systems, communication link, and inertial navigation system. The project's purposes are design and build a platform for research and development of UAV with autonomous behaviour. The proposed architecture consists of modular embedded electronics and communication protocols based on the OSI model.

Prisaznuk [16] proposed IMA, an integrated model of avionics that is used as architecture for conventional aircraft. IMA was initially proposed to be used in commercial and military aviation. It is set around the concept of high computational processing power and OS modules that allows independent processing of the application processing software. The modules share hardware resources and are allocated in offices, which have well-defined interfaces with the aircraft.

According to Watkins and Walter [17], it is possible differentiate IMA (Integrated Modular Avionics) architecture from the federated architecture (conventional). The authors state that the IMA architectures provide sharing of processors when processing information, communications and I/O. The resource row is divided for use of multiple avionics functions. The avionics functions served by IMA can be from different companies and their criticality is still guaranteed due to the robust partitioning mechanisms that are inherent in the architecture. In contrast, federated avionics architectures implement independent collections of dedicated computing resources (CPU, communications and I/O) for each avionics function normally contained in Line Replaceable Units (LRUs) or Line Replaceable Modules (LRMs). According to [18], other characteristics of IMA are the use of open standards and provision of a single data bus to interconnect the major aircraft systems.

Advantages of IMA compared to federated architecture are the economy of space, weight and power consumption, due a single unit performs various functions. Another advantage is the consolidation of hardware; it has several applications running on fewer processors [18].

Many complex embedded systems are coupled to a high-level information system. SOA can provide the integration of low-level embedded system services and high-level information system services. This integration is still an incomplete work, despite the many related works found in literature [19]-[25]. In practice, the use of SOA in embedded systems can provide a lot of benefits, such as decoupling configuration from environment, improvement of reusability and maintainability, higher level of abstraction and interoperability, more interactive interface between devices and information systems, and easy use of resource-hungry services provided by more powerful internet servers.

Using SOA and a reference model architecture is possible to get new improvements in critical embedded system. It is possible to take the advantages of the flexibility and can facilitate the modularization of the system components. It is also considered that the adoption and maintenance of standardized interfaces for UAVs can protect clients' investment in the development of new systems.

III. LARISSA: Layered Architecture Model Interconnection Systems in UAV

Architecture is a structure that identifies, defines, and organizes components. The relationship and the principles of design of components, functions and interface established between subsystems can also be defined by architecture. Moreover, a reference model for architecture is an architecture which the entities, relationships and information units involved in the interactions between and within the

subsystems and the components are defined and modelled. In short, it is a model that incorporates the basic purpose and the idea of the system and can be considered as a reference for various purposes. The term architecture model used in this study reflects exactly that last statement: it incorporates the basic goal and ideas of the system.

The increasing use of UAVs should cause them to become increasingly common. In this scenario, the techniques proposed in this work will facilitate the development of automated applications for UAVs, allowing these vehicles be more easily inserted and incorporated into the airspace, contributing to their spread.

In order to propose a broader understanding of the component parts of a UAV system, we propose a layered model, which can be subdivided as needed. Figure 1 illustrates the model named LARISSA (Layered Architecture Model Interconnection Systems in UAV).

In LARISSA model, the components of a UAV may be divided into aerial segment and ground segment. The aerial segment is hierarchically composed of: (i) physical layer, (ii) distributed RTOS (Real Time Operating System) layer, (iii) system abstraction layer, (iv) monitoring and control layer, (v) navigation & services layer, and (vi) mission layer. The ground segment is divided into: (i) physical layer and (ii) ground station layer.

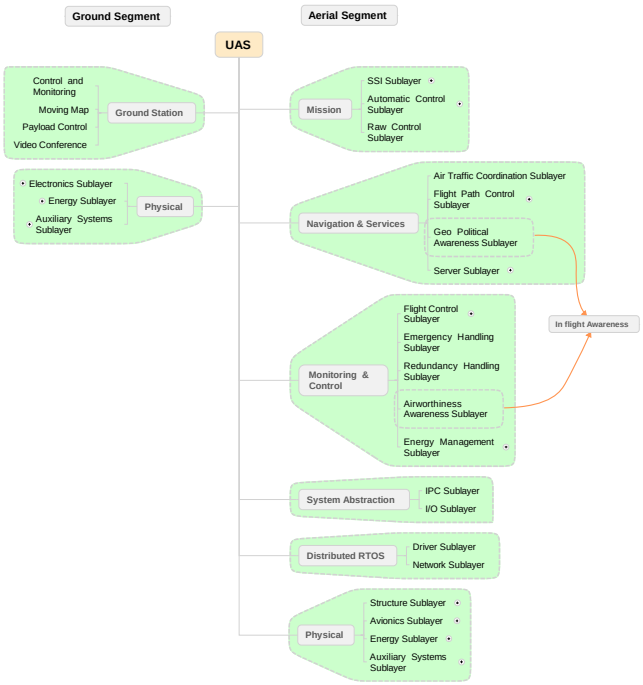

Fig. 1: LARISSA – The proposed reference model architecture.

The separation into layers allows the system to be divided into subsystems that can have different implementations and assists the separation of parts that compose a complex critical embedded system in different levels of criticality. Thus, the advantages offered by the service-oriented architecture can be applied in sections of low criticality, making the development of these sections simpler and more flexible.

These layers can be represented by models, which are intended to serve as guides for development of UAV systems, specifying how it will interconnect the various components, such as sensors, control circuits, GPS, payload, sensors, communication with the ground control station, and others.

In information technology, a layered architecture is used to define the specific responsibilities of each layer and the interconnection among them. Based on an architectural model, the hardware manufacturer or the software designer can develop their products knowing exactly which layer will interact in UAVs, which are the input and output parameters, and what type of connection must be used.

According to Tanenbaum [26], certain principles must be applied to define the layers: a layer must be created where there is need for other level of abstraction; each layer must perform a well-defined function, which should be chosen aiming the definition of standard protocols; layer limits must be chosen to reduce the flow of information transported among the interfaces; the number of layers should be large enough, so that different functions do not need be placed unnecessarily in the same layer, and small enough, so that the architecture does not become difficult to control. Architecture to be considered complete should define what each layer can perform, specifying services and protocols that are used in each one.

Moreover, the papers related to UAVs in the literature show UAVs implemented using traditional approaches [27][10][11]. On the other hand, there are roadmaps published periodically by military organizations (e.g., United States Air Force) illustrating the progress expected for UAVs, and they mentioned that in the future they may adopt an open, standardized, and scalable architecture, allowing the fast addition of modular functionality.

Each layer is composed of sub-layers, which are described in the next subsections. The navigation & services and mission sub-layers will be described in details because they are very important to the development of the framework proposed in Section IV.

A. Aerial Segment's Physical

The aerial segment's physical layer is the aircraft's hardware layer, which is decomposed in the structure, avionics, energy, and auxiliary systems sub-layer. Each sub-layer may be subdivided into more specific sub-layers.

B. Distributed RTOS

The distributed RTOS layer describes a set of API used by the real-time operating system embedded in the aircraft, used as input to or an output of the RTOS. In the driver sub-layer are the hardware drivers APIs, and in the network sub-layer are the network APIs.

C. System Abstraction

The system abstraction layer's function is defining a set of hardware for use in the upper layers. The IPC (Inter-Process Communication) sub-layer is responsible for the abstraction of communication among processes, and the I/O sub-layer controls the operation of input and output devices.

D. Monitoring & Control

The monitoring & control layer is responsible for monitoring the aircraft's actions, as well its control.

It is divided into the flight control sub-layer, emergency handling sub-layer, redundancy handling sub-layer, airworthiness awareness

 ISBN: 978-1-61208-331-5

sub-layer, and energy management sub-layer. The flight control sub-layer responds to basic commands being executed by the aircraft, by the automatic take-off and also by the automatic landing. On the other hand, the emergency handling sub-layer is responsible for events that are not planned, such as battery consumption, making it impossible to accomplish the mission. The redundancy handling sub-layer manages duplicated subsystems in the aircraft that were installed to increase the reliance. The airworthiness awareness sub-layer is responsible for sensors and embedded detectors in aircraft, which purpose is to obtain information like those a human being has the capability to identify, such as smoke on board. The energy management sub-layer is responsible for the monitoring of energy levels consumed by the aircraft.

E. Navigation & Services

The navigation & services layer, illustrated in Figure 2, consists of the air traffic coordination, flight path control, geo political awareness, and server sub-layers. This layer is responsible for the aircraft's navigation, sending signals that perform the required path to accomplish the mission. The air traffic coordination sub-layer responds to traffic in the airspace in which the aircraft is operating. The flight path control sub-layer guides the aircraft's navigation to achieve the waypoints or the grid coordinates defined by the mission. The geo political awareness sub-layer is responsible for the virtual threshold that the aircraft must operate. The server sub-layer contains non-priority services that help navigation and mission accomplishment. These services can be based on WWW (World Wide Web) or DTM (Data Transfer Mechanism).

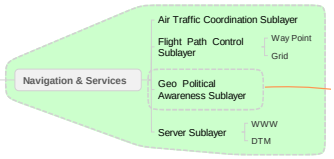

Fig. 2: Navigation & services layer.

F. Mission

The mission layer, illustrated in Figure 3, is divided into SSI (Smart Sensor Interface), Automatic Control, and raw control sub-layers.

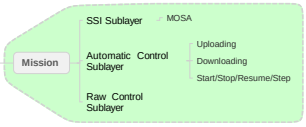

Fig. 3: Mission layer.

The SSI sub-layer is responsible for accessing the MOSA (Mission Oriented Sensor Array) [28] and performing the entire checking, allowing discovering whether the aircraft met all the attributes to accomplish the defined mission. The automatic control sub-layer is responsible for accepting the mission data (uploading), sending the collected data (downloading), and starting, stopping, resuming or performing part of the mission (start/stop/resume/step). The raw control sub-layer is simply responsible for sending data that does not need a proper treatment. With this layer, we ensure that data reaches a pre-set destination.

This layer is the last one from the aerial segment. The ground segment's layers are described in the next subsections.

G. Ground Segment's Physical

The ground segment's physical layer resembles, in some aspects, the air segment's physical layer. The division into sub-layers is presented as follows: electronics, energy, and auxiliary systems.

H. Ground Station

The ground station layer has control and monitoring, moving map, payload control, and video conference sub-layers.

The control and monitoring sub-layer receives information in the form of aircraft telemetry and can also issue commands to guide the aircraft. The moving map sub-layer is responsible for the exchange of maps ability, submitting new maps to the aircraft when the initial mission is changed. The payload control sub-layer sends signals to aircraft in order to control the movement and operation of sensors, cameras and radars. The video conference sub-layer is responsible for exchange sound and image with other control stations.

IV. Knowledge Based Framework for Dynamically Changing Applications (KBF)

To make possible the development of KBF, this paper considers critical embedded systems can be divided into sections of low and high criticality, based on LARISSA.

KBF was proposed in [29] and [30] and is currently under development. It extends the capability of a SOA broker's service discovery, adding knowledge about the application domain. Thus, KBF will use context and monitoring information to select or compose dynamically the best service to perform a specific mission. This selection or composition will be based on a set of usage rules and selection criteria, such as reliability, security, and performance. KBF is illustrated in Figure 4 and can be seen in detail in [9].

KBF uses a knowledge database to store all information and selection criteria defined by the user and by the application. Another key issue is the assembly of reconfigurable matrix, a data structure that correlates the chosen service, its functionality and the selection criteria to mission procedures. This matrix can be: static, semi-static and dynamic, depending on its composition and system operation [9]. Using all available information in the reconfigurable matrix, KBF can either choose or compose the best service to run a mission defined by the user.

V. Case Study: Fast, Reliable, and Secure Web Services in UAVs

The definition and specification of basic services that KBF can use were done, including their features and input/output parameters. These services were implemented using the Java programming language. These services use UAV's basic information (e.g., maximum cruise altitude, cruise speed).

In these experiments, we implemented the services listed in Figure 5. Those services were then replicated, adding reliability through WS-ReliableMessaging (WS-RM) specification [31], which is responsible for guaranteeing that messages are really delivered. Then those

 ISBN: 978-1-61208-331-5

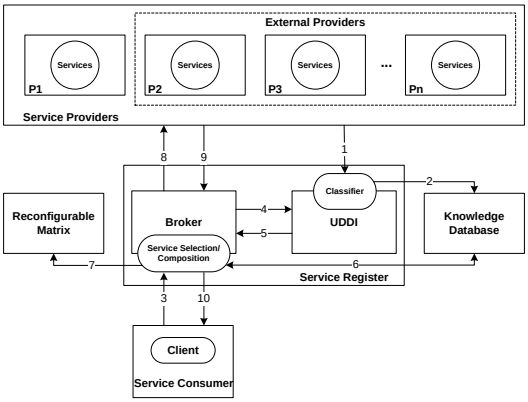

Fig. 4: Knowledge Based Framework for Dynamically Changing Applications (KBF)

services were replicated once again, but this time we added security through WS-Security specification [32], which is responsible for applying cryptography and digital signature to SOAP messages. So there are three different versions of the initial services: one secure, one reliable and a plain version, i.e., without additional parameters. All of these were hosted in and provided by an Apache Tomcat server running on a remote machine.

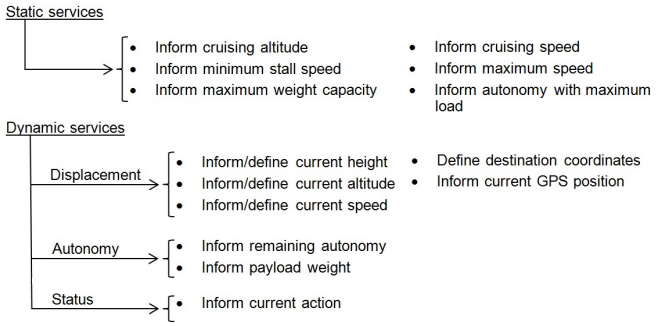

Fig. 5: Basic services implemented in KBF.

On the client side, we implemented an application that makes several calls to those services. The amount of calls varies from one experiment to other. In these experiments, client and server were in different machines at the same local network.

In each experiment, the clients were run several times and the Round Trip Time (RTT) was measured in each repetition. In the end of the experiment, the average RTT was done, as well the standard deviation and confidence interval (95%) in each case for the different versions and then, finally, the results were compared.

In order to compare the performance in terms of RTT of the different versions, we realized an experiment in which a client machine runs a sequence of service requests, and the RTT was measured for different sequence sizes, i.e., for different amounts of services. The results were obtained through 30 repetitions of each experiment.

As shown in Figure 6, for each experiment (different amount of services), the plain version achieved the best performance when compared to the other two versions. Since it has no additional

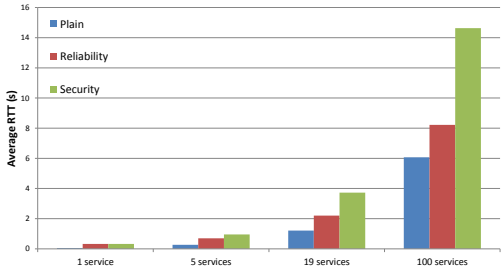

Fig. 6: Average RTT of different amount of services, for different versions.

parameters, it has fewer operations to be executed; therefore it takes less time to be executed.

In the first experiment, with 1 service, both the reliable and the secure versions achieved a similar RTT. Since the secure version uses cryptography and digital signature, it would be usual to think it might have the highest RTT. However, for a single service, the WS-RM exchanges 4 more messages than the other two versions. That is three more times messages in a single services call, and since access to the network is a slow operation, it causes a higher impact on the RTT, making it similar to the impact caused by security operations such as cryptography and digital signature.

In the next experiment, with 5 services, the average RTT increased for all three versions. However, now there is a difference between the secure and reliable version's RTT, because the 4 extra messages of the WS-RM resulted in less than 3 times more messages. Therefore the impact caused by the cryptography and digital signature operations is now greater than the one caused by extra usage of network.

The experiment with 19 services evaluates the behaviour of an application used in a real life situation. As shown in Figure 6, the plain version achieved the best performance, followed by the reliable version in second, and the secure version achieved the highest RTT. Besides, the difference between the reliable and secure versions is greater now than before. Since there are more messages being exchanged, the impact caused by the WS-RM is even smaller.

This pattern can also be observed in the last experiment, with 100 services. The impact of the extra messages sent by the WS-RM specification is even smaller, making the reliable version's RTT more far to the secure version and closer to the plain version.

VI. Conclusions

UAVs are complex systems that perform complex missions. Large UAVs systems are distributed in dozens of different processor systems. The Reference model architecture aims to standardize the various parts that makes up a system. This kind of architecture brings benefits to systems like UAVs, primarily for being safety critical and complex systems. In this sense, LARISSA meets the existing needs.

This paper also introduced the use of SOA in critical embedded systems, providing dynamic behaviour and flexibility to this class of systems. However, the issue of choosing the parts of the system that can be implemented with this technology, without compromising its safety-critical nature, is not a trivial task.

Different types of services, based on avionics, and the effects on performance when using them were also presented. The results

 ISBN: 978-1-61208-331-5

showed that by applying security and reliability to these services, a considerable overhead is generated, and so this might cause problems on applications that are exclusively dependent on real time performance. However, considering UAVs, there are occasions in which the real time performance is not the main requirement. Therefore it is possible to use these resources, but it is designer's responsibility to decide where and when to make services secure and/or reliable, and also to decide the level of security required for the mission being developed.

The architecture and the framework presented, backed up by the tests results, allow the use of SOA in the sections of low criticality of safety-critical embedded systems, specially UAVs, leading to a breakthrough in the development of this class of systems, making it easier and more feasible to create, reuse and maintain safety-critical embedded systems.

REFERENCES

[1] L. Lazić and D. Velašević, "Applying simulation and design of experiments to the embedded software testing process: Research articles," *Software Testing, Verification & Reliability*, vol. 14, no. 4, pp. 257–282, Dec. 2004.

[2] A. Armoush, E. Beckschulze, and S. Kowalewski, "Safety assessment of design patterns for safety-critical embedded systems," in *SEAA '09: Proceedings of the 2009 35th Euromicro Conference on Software Engineering and Advanced Applications*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 523–527.

[3] S. P. Kumar, P. S. Ramaiah, and V. Khanaa, "Architectural patterns to design software safety based safety-critical systems," in *ICCCS '11: Proceedings of the 2011 International Conference on Communication, Computing & Security*. New York, NY, USA: ACM, 2011, pp. 620–623.

[4] Z. Yi, W. Cai, and W. Yue, "Adaptive safety critical middleware for distributed and embedded safety critical system," in *NCM '08: Proceedings of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 162–166.

[5] R. Bergamaschi, G. Martin, W. Wolf, R. Ernst, K. Vissers, and J. Kouloheris, "The future of system-level design: Can we find the right solutions to the right problems at the right time?" in *CODES+ISSS '03: Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. New York, NY, USA: ACM, 2003, pp. 231–231.

[6] GAO, "Unmanned aircraft systems: Federal actions needed to ensure safety and expand their potential uses within the national airspace system," 2008, United States Government Accountability Office. Report to Congressional Requesters.

[7] O. Trindade, L. O. Neris, L. C. P. Barbosa, and K. R. L. J. C. Branco, "A layered approach to design autopilots," in *ICIT '10: Proceedings of the IEEE International Conference on Industrial Technology*, march 2010, pp. 1415–1420.

[8] K. R. L. J. C. Branco, J. M. Pelizzoni, L. O. Neris, O. Trindade, F. S. Osório, and D. F. Wolf, "Tiriba - a new approach of uav based on model driven development and multiprocessors," in *ICRA '11: IEEE International Conference on Robotics and Automation*, may 2011, pp. 1–4.

[9] D. Rodrigues, R. M. Pires, J. C. Estrella, M. Vieira, M. Correa, J. B. Camargo, K. R. L. J. C. Branco, and O. Trindade, "Application of SOA in safety-critical embedded systems," *Communications in Computer and Information Science*, vol. 206, pp. 345–354, 2011.

[10] DoD, "Unmanned systems roadmap 2007-2032," 2007, U.S. Department of Defense. Office of the Secretary of Defense.

[11] DoD, "Unmanned systems integrated roadmap FY2009-2034," 2009, U.S. Department of Defense. Office of the Secretary of Defense.

[12] NIST, "4D/RCS: A reference model architecture for unmanned vehicle systems version 2.0," 2002, NIST. U.S. Department of Commerce.

[13] E. Pastor, J. Lopez, and P. Royo, "UAV payload and mission control hardware/software architecture," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 6, pp. 3–8, 2007.

[14] L. Olson and L. Burns, "A common architecture prototype for army tactical and fcs uavs," DASC '05: The 24th Digital Avionics Systems Conference. Washington,: IEEE, 2005, pp. 8.B.5–1 – 8.B.5–11.

[15] J. Neto, R. Paixao, L. Rodrigues, E. Moreira, J. dos Santos, and P. Rosa, "A surveillance task for a uav in a natural disaster scenario," 2012 IEEE International Symposium on Industrial Electronics (ISIE). Hangzhou: IEEE, 2012, pp. 1516–1522.

[16] P. J. Prisaznuk, "Integrated modular avionics," in *National Aerospace and Electronics Conference (NAECON)*. IEEE, 1992, pp. 39–45.

[17] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," DASC '07: 26th Digital Avionics Systems Conference. Dallas, TX, USA: IEEE, 2007, pp. 2.A.1–1 – 2.A.1–10.

[18] W. R. Inc, "Arinc 653: An avionics standard for safe, partitioned systems," in *IEEE-CS Seminar*, 2008.

[19] N. R. Kakanakov, "Experimental analysis of client/server applications in embedded systems," in *Proceedings of the Annual Scientific and Applied Science Conference Electronics*, vol. 4, 2005, pp. 97–102.

[20] N. R. Kakanakov and G. Spasov, "Adaptation of web service architecture in distributed embedded systems," in *CompSysTech '05: Proceedings of the International Conference on Computer Systems and Technologies*, 2005, pp. 1–6.

[21] K. C. Thramboulidis, G. Doukas, and G. Koumoutsos, "A SOA-based embedded systems development environment for industrial automation," *EURASIP Journal on Embedded Systems - Embedded System Design in Intelligent Industrial Automation*, vol. 2008, pp. 1–15, 2008.

[22] M. H. Lee, C. J. Yoo, and O. B. Jang, "Embedded system software testing based on SOA for mobile service," *International Journal of Advanced Science and Technology*, vol. 1, no. 1, pp. 55–64, 2008.

[23] G. Moritz, S. Prüter, D. Timmermann, and F. Golatowski, "Web services on deeply embedded devices with real-time processing," in *ETFA '08: IEEE International Conference on Emerging Technologies and Factory Automation*, 2008, pp. 432–435.

[24] S. Deugd, R. Carroll, K. E. Kelly, B. Millett, and J. Ricker, "SODA: Service oriented device architecture," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 94–96, 2006.

[25] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains," in *ICNICONSMCL '06: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 43–48.

[26] A. S. Tanenbaum, *Computer Networks*, 5th ed. Prentice Hall, 1997.

[27] K. P. Valavanis, *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*, ser. International Series on Intelligent Systems, Control, and Automation. Springer, 2007.

[28] R. M. Pires, D. Rodrigues, K. R. L. J. C. Branco, and O. Trindade, "Mosa - mission oriented sensor array: A proposal," in *CLEI '11: Proceedings of the XXXVII Conferencia Latinoamericana de Informática*, 2011, pp. 1309–1318.

[29] D. Rodrigues, R. M. Pires, J. C. Estrella, M. Vieira, M. Correa, J. B. Camargo, K. R. L. J. C. Branco, and O. Trindade, "Application of SOA in safety-critical embedded systems," *Communications in Computer and Information Science*, vol. 206, pp. 345–354, 2011.

[30] D. Rodrigues, R. M. Pires, J. C. Estrella, E. A. Marconato, O. Trindade, and K. R. L. J. C. Branco, "Using SOA in critical-embedded systems," in *Proceedings of the 2011 IEEE International Conferences on Internet of Things (iThings), and Cyber, Physical and Social Computing (CPSCom)*, 2011, pp. 733–738.

[31] OASIS, "Web services reliable messaging (WS-ReliableMessaging) version 1.2," 2009, http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.html, Accessed 06 April 2014.

[32] OASIS, "Web services security (WSS) TC," 2006, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss, Accessed 06 April 2014.

 ISBN: 978-1-61208-331-5

A Fault Tolerance Approach Based on Reinforcement Learning in the Context of Autonomic Opportunistic Grids

Alcilene Dalília de Sousa and Luciano Reis Coutinho
Departamento de Informática
Universidade Federal do Maranhão, UFMA
São Luis, Maranhão - Brasil
e-mail: alcileneluzsousa@gmail.com / lrc@deinf.ufma.br

***Abstract* — Fault tolerance is a longstanding problem. Two basic solutions are replication and checkpointing, both with their pros and cons. In this paper, we put forward an approach to balance replication and checkpointing in order to provide fault tolerance in opportunistic grid computing systems. We try to retain the benefits of both techniques, while avoiding their downsides. The approach combines reinforcement learning with the MAPE-K architecture for autonomic computing. To validate our proposal, we have performed experiments based simulation using the Autonomic Grid Simulator Tool (AGST). We report promising results. We show that the proposed approach is able to learn suitable switching thresholds between checkpointing and replication. The suitability is verified by comparing the average completion time and the success rate of applications of our proposal against the values from other approaches in the literature.**

Keywords - ***fault tolerance; grid computing; opportunistic grids; autonomic computing; reinforcement learning.***

I. INTRODUCTION

Achieving a high processing rate by dividing computational tasks among several geographically distributed machines is, in essence, the core idea behind the computational model called *Grid Computing* [9]. In this context, it was introduced the concept of *opportunistic grids* that, potentially, gather thousands of resources, services and applications to provide greater computational power at a lower cost [12]. On the one hand, this type of computational grid promotes the use of non-dedicated resources, such as desktop workstations situated in different administrative domains, by using their idle processing power. On the other hand, there is the challenge of providing services in a dynamic execution environment, where heterogeneous nodes can enter and leave the grid at any time. In this case, it is important to be able of effectively monitoring the grid composition in order to detect and react to these events in a timely manner.

Grid computing encompasses various technical challenges. One of them, especially in the context of opportunistic grids, is how to provide *fault tolerance* in an inherently dynamical environment, an environment in which computing nodes are heterogeneous and can become unavailable at any time. Fault tolerance is the ability of a system to continue to work even in the presence of faults [7]. We say that a fault has occurred when one of system components fails or malfunction, leading to a behavior not in accordance with the system specifications. Concerned with this problem, researchers have been seeking for solutions. Among these, we found some approaches based on the idea of *autonomic computing*. Broadly, the idea of autonomic computing consists in modeling and building computing systems that have the ability of self-management and self-adaptation to unpredictable changes [6][8][10]. Applied to the problem of fault tolerance in opportunistic grids, autonomic computing has given rise to approaches where the grid middleware tries to automatically adjust parameters or to dynamically combine traditional fault tolerance techniques such as *checkpointing* and *replication* [2][15][16][18].

In general, these autonomic approaches provide important gains w.r.t. the traditional fault tolerance techniques. Despite these gains, there are some opportunities for improvement. In this paper, we focus on the following issue regarding the approach presented in [15] [16] (and based on [2]). When should we switch from checkpointing to replication, and vice-versa, given the current workload of an opportunistic grid system? Viana [15] fixes a threshold below which the grid performs replication and above which it performs checkpoint. Our hypothesis is that this threshold can vary, and that this variation is beneficial to the fault tolerance strategy of the grid system. It can lead to fewer delays in the execution time of the application due to fault tolerance concerns.

To test this hypothesis, we put forward an adaptive approach to the problem of balancing checkpointing and replication in the context of opportunistic grids. Our approach is based on the use of *reinforcement learning*. Reinforcement learning is a paradigm of machine learning based on trial-and-error and delayed reward [1][13]. A typical reinforcement learning problem consists in finding a policy (a decision function) that maps environmental states to actions. Such a policy is optimal when it can be used to guide our behavior through the environment in such a way that we obtain maximum cumulative reward over time. In the case of the problem of balancing checkpointing and replication, we want an optimal (or near optimal) policy that help us to decide, directly or indirectly, when it is the right time to switch from checkpointing to replication, and vice-versa. By choosing reinforcement learning we are following the path of several researchers in the areas of Grid, Cloud and Autonomic Computing [3][4][14][17][19].

 ISBN: 978-1-61208-331-5

The paper is organized as follows. In Section II, we further discuss the concepts of *grid* and *autonomic computing*, and briefly present AGST, the *Autonomic Grid Simulation Tool* we have used in our experiments. In Section III, we characterize the need and challenge of *fault tolerance* in grid computing environments, and report on the state of the art. In Section IV, we present the basic ideas of *reinforcement learning* and related algorithms. In Section V, we describe our approach to the problem of fault tolerance in opportunistic grid which merges ideas from autonomic computing and reinforcement learning. Section VI reports some experimental results we have obtained when evaluating the approach. Finally, in Section VII, we draw our conclusion and discuss future work.

II. GRID AND AUTONOMIC COMPUTING

Grid computing is a particular ideal case of distributed and parallel computing [9]. It seeks to extend the potential of computer networks to enable the sharing, selection and aggregation of geographically distributed and possibly heterogeneous computing resources (e.g., processors, data and applications) in a pervasive and transparent way. The idea is that individual users (client applications) can access computing resources as needed with a minimum knowledge of localization, underlying technologies of hardware and software, etc.

A. Opportunistic Grids

Opportunistic grids are grid computing systems that promote the dynamical integration of non dedicated workstations, possibly distributed along several administrative domains (e.g., organizations, academic laboratories, home PCs, etc.), by using their idle computing time to the execution of parallel applications [12]. This way, opportunistic grids are highly heterogeneous and dynamic. They aggregate regular personal computers, with their particular hardware and software configurations, to execute distributed application in large scale. And these machines can enter and leave the grid at different times, using network connections with different capabilities with regard to properties such as bandwidth, error rate and communication latency.

From the point of view of the user, the grid computing system should be viewed as a single integrated resource and should still be easy to use. These are some of the challenges that an opportunistic grid middleware face. The *grid middleware* is a software layer between the operating systems running on the computing nodes and the user applications submitted to the grid. The focus of an opportunistic grid middleware is not the integration of dedicated computer clusters [11], or to provide supercomputing resources, but to promote a better use of existing computing resources and the execution of computationally intensive parallel applications.

B. Autonomic Computing

Autonomic computing, as a research field, aims at developing computational systems able to manage themselves with minimal human intervention [6][8][10]. The term autonomic comes from biology and is inspired from the human nervous system. Like the nervous system, an autonomic computing system must possess some characteristics or properties such as *self-awareness*, *self-configuration*, *self-protection*, *self-optimization* and *self-healing*, among others [6]. In sum, these characteristics relate autonomic computing to the design of complex systems; systems that need constant adjustments to various dynamical circumstances, as is the case of opportunistic grid systems.

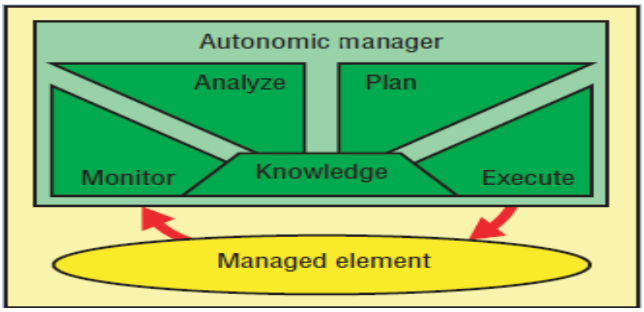

Figure 1. Architecture of an Autonomic Element [8].

In general, an autonomic computing system is conceived as one or more *autonomic elements* composed of an *autonomic manager* associated to a *managed element* (see Figure 1). The managed element represents a resource or device (computer, printers, databases, etc.) composing a computing system. The autonomic manager is an active component that encapsulates a managed element to turn it into an autonomic element. The internal working of an autonomic manager is organized in cycles, each one divided in four distinct phases: *monitoring*, in which data is collected by means of sensors; *analysis*, in which possible needs and problems are detected; *planning*, in which strategies are drawn to make necessary changes and adaptations; and *execution*, in which the planned strategies are effectively implemented. This general architecture is known as the MAPE-K model.

C. Autonomic Grid Simulator Tool

In our research, we have used the *Autonomic Grid Simulator Tool* (AGST)[15], a tool that allows the modeling and simulation of autonomic approaches to self-management problems in the context of Opportunistic Grids [5]. AGST is based on the MAPE-K model, providing support for all phases of the autonomic manager execution cycle. By using AGST, one can simulate autonomic approaches based on two types of dynamic adaptation: *parametric* and *compositional*. Parametric adaptation consists in the continual modification of variables that determine the behavior of algorithms used by the grid middleware. And compositional adaptation is the exchange of algorithms or components of the grid middleware, enabling the adoption of new strategies to handle new situations and to react to changes in the grid environment.

 ISBN: 978-1-61208-331-5

III. FAULT TOLERANCE

In opportunistic grid systems, and distributed systems in general, it is important to have mechanisms that allow the system to continue functioning despite the presence of faults [7]. Faults occur every time the system behavior does not comply with its intended operation due to some failure or malfunctioning of one or more of its components. What happens after a fault determines the degree of fault tolerance of the system. At one extreme, we have full fault tolerant systems in which fault does not decrease their quality of service. At other, there are zero fault tolerant systems in which any fault causes total system breakdown. Between these extremes, we found systems that present a graceful degradation characterized by a reduction in the quality of their operations proportional to the severity of the fault. Given that failures are inevitable in dynamic and heterogeneous systems, and that the cost of zero fault tolerance can be very high, fault tolerance becomes a characteristic of paramount importance.

A. Basic Techniques

A basic source of faults in opportunistic grid systems is the loss of running tasks due to problems on a grid node. These can be caused by many factors, among them the computer be turned off by its owner. In general, this is the kind of fault tolerance we deal in our work, and that will be considered in the remainder of the paper.

Ultimately, the less sophisticated way to deal with the fact that nodes may become unavailable is to detect the node failure and restart, in another available node, the tasks that were running on it. To avoid this restart, the researchers have devised two basic techniques: *replication* and *checkpointing* [7]. Replication consists in executing several replicas of the same task in different nodes at the same time. On the one hand, with several replicas, the change of having to restart a task due to a node failure is minimized. On the other hand, the grid middleware has to manage and synchronize several running replicas that consume computational resources and leave less space for scheduling new incoming tasks. Looking for a better use of resources, checkpointing is a technique that promotes fault tolerance by periodically saving the state of the running tasks so that they can be resumed on a different machine in the case of a node failure. However, the downside of this technique is the time overhead imposed upon the tasks that need to be constantly interrupted by the grid middleware to record their running states.

Replication and checkpointing have pros and cons depending on different conditions of the grid environment. If there are plenty of computing nodes in comparison to number of tasks to be run, then replication is a better option than checkpointing. But, to the extent that the number of available computing nodes decreases, replication becomes less and less attractive until we reach a point where checkpointing becomes a better option than replication. With these pros and cons in mind, in the last few years, some researchers have proposed adaptive fault tolerance mechanisms that try to autonomically balance the use of replication and checkpointing depending on the current grid condition [2][15][18].

B. State of the Art

Wu et al. [18] propose a mechanism based on the number of times a task is resumed due to node failures. Initially, for each new task, the grid middleware performs checkpointing. If a node failure occurs, the task is restarted from the last saved state in the same computing node. It is considered that it was only a transient fault and that the restarting on the same node is sufficient to solve the problem. If a second fault occurs, it is considered that probably the node in which the task is running is not stable and, therefore, the task is restarted from the last checkpoint on another node. If the task fails a third time, it is considered that the grid environment has a high fault rate and, therefore, the middleware starts multiple replicas of the task to be executed simultaneously.

Chtepen et al. [2] present heuristics for the adaptive use of checkpointing, replication and a combination of them. The goal is to improve resource utilization and reduce the execution time of tasks. In the case of checkpointing, one heuristics consists in increasing or decreasing the interval between checkpoints for each task according to the mean time between failures (MTBF) of the computing nodes. Regarding replication, another heuristics is to limit the use of replication according to the system workload (grid occupancy). A third heuristics is to dynamically switch from checkpointing to replication, and vice-versa, based on workload (if occupancy is high use checkpointing, otherwise use replication).

Based on the work by Chtepen et al. [2] and the MAPE-K model, Viana et al. [15][16] propose an autonomic fault tolerance mechanism for opportunistic grids. The basic idea is to make each computing node a managed element controlled by an autonomic manager. Thus, the autonomic manager continually adjusts the parameters of the fault tolerance technique currently in use for each node of the grid. It also makes a structural reconfiguration, replacing the fault tolerance technique in use by another one, when the system workload reaches a given fixed threshold.

Despite being adaptive (by combining replication and checkpointing taking into account the current state of the grid environment), these state of art mechanisms still depend upon certain parameters that need to be adjusted empirically by the system administrator. One example is the fixed threshold to switch between checkpointing and replication found in [15]. Another limitation perceived is that [2] and [15] rely only on a measure of grid occupancy to switch between checkpointing and replication. As can be seen in the work [18] other factors such as rate mean time between failures (MTBF) can also have a decisive influence in this decision.

Motivated by these shortcomings, we put forward an extension of the work by Viana et al. [15]. Our proposal,

 ISBN: 978-1-61208-331-5

presented in Section V, is inspired by some studies that use *Reinforcement Learning* for resource allocation in computing grids, such as [3][4][14][17][19].

IV. REINFORCEMENT LEARNING

Reinforcement learning is a *Machine Learning* paradigm [1][13] that addresses the issue of how an *agent* (i.e., an autonomous entity that perceive and act in an *environment*) can interactively learn the right *policy* to achieve a given purpose (see Figure 2).

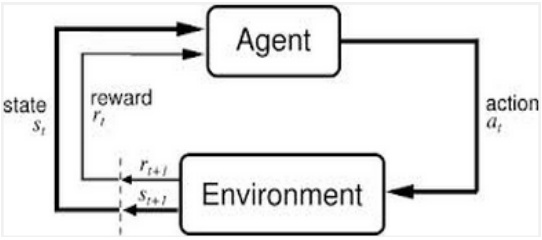

Figure 2. The agent-environment interaction [13].

A. Markov Decision Process

Formally, the problem faced by the agent in a reinforcement learning scenario is rendered as a *Markov Decision Process* (MDP)[13]. A MDP is characterized by a finite set of environmental *states S*; a finite set of possible *actions A*; a *state transition function* $T: S \times A \rightarrow \Pr(S)$ that gives, for each state and action pair, a probability distribution over the set of states (where $T(s, a)(s')$ is the probability of the environment transit to state s' when the agent perform action a in state s); and a (expected) *reward function* $R: S \times A \rightarrow \mathbb{R}$ that maps each state and action pair to a real value representing the (expected) immediate reward after performing action a in state s. In this setting, the functions T and R abstract the dynamics of the environment and are not necessarily known to the agent.

To solve a MDP is to come up with an optimal *policy* $\pi: S \rightarrow A$, i.e., a decision function that maps each possible state to an action in such a way to produce, over time, the greatest possible cumulative reward to the agent.

B. Temporal Difference Learning

In general, reinforcement learning algorithms are based on estimating value functions that characterize optimal policies. One of these functions maps state action pairs (s, a) to real values $Q(s, a)$ that are estimations of the cumulative reward that the agent is expected to receive in the long run if it performs the action a in the state s.

Two popular algorithms for learning Q value functions are Q-learning and SARSA [13]. Both are *Temporal Difference Learning* (TD Learning) algorithms. This means that they work by using the difference between the current and previous estimates to incrementally update Q values. Specifically, the update rule in SARSA is $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$; and in Q-learning is $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$; where α is a step size parameter, γ is the discount factor for future rewards, $r = R(s,a)$ and s' the observed next state when the action a was performed in the current state s.

The basic difference between Q-learning and SARSA lies in the action used to update the Q values. In Q-learning the update is done with the optimal action obtained from a greedy choice. Regarding SARSA, it is used the next action obtained when the agent follows the policy derived from the current Q values. In practice, this difference is reflected in the optimality and in the safeness of the learned policy. Q-learning tends to find policies with expected cumulative rewards higher than SARSA. However the polices found by SARSA are safer than those by Q-learning, in the sense of obtaining less negative rewards when the agent deviates from the policy to explore new possibilities.

V. PROPOSED APPROACH

As discussed in Section III, recent autonomic mechanisms for fault tolerance in grid systems have used heuristic rules that require empirical adjustments of some parameters. Discovering these parameters is not always an easy task. Thus, we propose a fault tolerance approach that extends the work by Viana et al. [15][16] by using Reinforcement Learning to automatically adjust the threshold used to switch between checkpointing and replication. Instead of relying only in the occupancy level of the grid, the idea is to make this switching also dependent on the amount and reliability of the computing nodes composing the grid system.

A. Adaptive Switching Threshold

The autonomic fault tolerance mechanism in [15], based on [2], deals with two levels of adaptation: parametric and structural adaptation. Regarding parametric adaptation, some parameters such as number of replicas or frequency of checkpointing are dynamically adjusted depending on the grid statistics (e.g., MTBF and grid workload). With respect to structural adaptation, what occurs is the switching between checkpointing and replication based on the current grid occupancy.

We retain these basic ideas from [15]. However, we add to the parametric adaptation a further item: the *switching threshold between checkpointing and replication* (measured in terms of grid occupancy percentage). To do this parametric adaption, we develop an approach in which the autonomic manager learns by reinforcement how to increment or decrement the switching threshold in order to minimize the execution time of successfully completed applications. In this way, the grid middleware initially adopts a switching policy reflecting the threshold proposed in [15]. Over time, to the extent that applications complete, the middleware try to increment or decrement the threshold guided by the amount of delayed or restarted applications. At the end, the switching threshold is modified to a value below or above the default value reflecting the particular

 ISBN: 978-1-61208-331-5

characteristic of the grid environment (e.g., MTBF, grid workload and number of computing nodes).

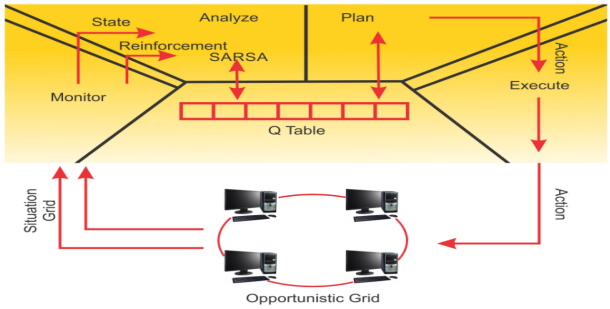

Figure 3. Proposed approach.

In Figure 3, we depict the proposed approach. It is an adaptation of the MAPE-K model shown in Figure 1. The managed elements are the nodes of an opportunistic grid. The autonomic manager, in addition to dealing with the parametric and structural adaptations laid down in [15], is supposed to record and update the policy for changing the switching threshold. This policy is represented as a Q-table, a table that holds a Q value function. During the analysis phase, the Q-table is updated given the current state and reinforcement coming from the grid environment. The update is performed by using the SARSA algorithm (here we have opted for the safeness at the expense of optimality). In the planning phase, the policy coded in the Q-table is used to update the switching threshold and to decide if a structural adaptation (change to checkpointing or replication) is needed or not.

B. Markov Decision Process

In sum, the problem of adapting the switching threshold, formalized as a MDP, consists in: states $s \in \{$ *threshold values in terms of grid occupancy percentage* $\}$; actions $a \in \{$ *increment, decrement, maintain* $\}$; state transition function $T(s,a)$ is deterministic and previously known since increment and decrement assume their mathematical meaning of addition and subtraction, and maintain means leave the threshold value unaltered; immediate reward function $R(s,a)$ is unknown, but delayed negative reward values are calculated from the amount of delayed and restarted tasks (i.e., for each autonomic cycle, the number of delayed and restarted tasks is counted and aggregated as a negative reward; it is used a weighted sum in which a restart is 10 times worse than a delayed task).

VI. EVALUATION OF THE APPROACH

To evaluate the proposed approach, we have conducted several simulation experiments using AGST (Section II-c). To put into perspective the results obtained we compare our approach with the traditional techniques of checkpointing, replication, and the autonomic approach reported in [15].

A. Scenarios

Here, we consider two basic scenarios: 1 – many resources, many faults, 100 applications; and 2 – few resources, many faults, 200 applications.

In the first scenario, we created a simulation model with 1400 computing nodes; in the second, 700 nodes. In both, the nodes were interconnected by a network of 100 Mbps. The medium processing power was equivalent to a Pentium IV 1.6 GHz (1,858 MIPS, based on the TSCP 5 benchmark); to simulate heterogeneity, this medium varies according to a uniform distribution U (938; 2,779) MIPS, where the processing power of the faster machine is approximately three times greater than the processing power of the slower machine. Regarding the faults, AGST was configured to generate synthetic failures with an exponential distribution with MTBF equal to 500 seconds. The duration of failure (downtime) was determined by an exponential distribution with variable mean, whose minimum and maximum values were respectively 300 and 600 seconds (faults with fast recovery, typical of opportunistic grids environments where frequent failures are due to restarting of machines by the users, or electrical current fluctuations, instead of long terms failures such as hardware failures). At last, concerning the grid workload, the applications consist in *bag-of-tasks* applications with three tasks each, resulting in a total of 300 tasks in the first scenario and 600 tasks in the second. These applications were generated with a variation in size (in terms of millions of instructions) according to a uniform distribution U (53,510; 321,062) MI (considering the medium processing power of 1,858 MIPS, each application would take approximately from 8 to 48 hours to complete). All these settings are similar to the settings found in [15] to easy the comparison.

B. Simulations

By combining the four fault tolerance strategies (checkpointing, replication, autonomic [15] and our approach denoted in the sequel as RLearning) and the two scenarios we reach at eight different simulations. In the simulation involving checkpointing, the technique was configured to perform the checkpoint of the tasks on a fixed interval of 30 minutes. With regard to replication, it was configured to statically create three replicas for each task. The autonomic approach was configured as described in [15]. Specifically, the threshold adopted for switching between checkpoint and replication is 30% (i.e., when the grid workload is < 30% use replication, otherwise use checkpointing). Finally, the RLearning approach follows the same configurations of the autonomic approach, with the difference that the switching threshold is variable.

All the eight different simulations were repeated 40 times, resulting in a total of 320 experiments. The metrics used to compare the fault tolerance strategies were the average completion time (in hours) and the success rate of the applications (percentage of application that concluded execution without restarting).

 ISBN: 978-1-61208-331-5

C. Results

The simulation results are shown in Figures 4 and 5.

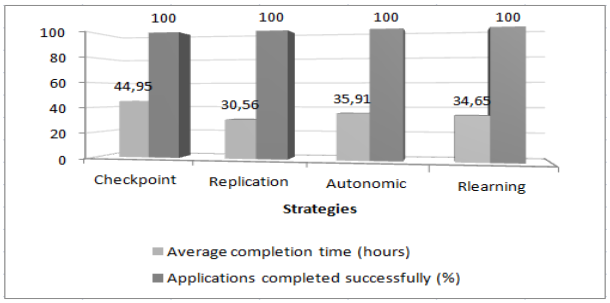

Figure 4. Scenario 1 – 1,400 nodes, 300 tasks and many faults.

Analyzing Figure 4, it is possible to notice that when there plenty of resources the best strategy is to use replication only. It leads to the smallest average completion time (30.56) with 100% of successfully completed applications. The worst is checkpointing only. In between these extremes we have the autonomic and the RLearning approaches. They both approximate the results of replication only by using replication when there are many idle nodes. As applications are submitted and replicas are created, the grid occupancy begins to increase, forcing a switch to checkpointing when the workload reaches the predefined threshold value. This explains why the autonomic and RLearning approaches are beaten by replication in a scenario with many resources and few applications.

When we focus on the autonomic and the RLearning approaches, we see that RLearning was a better approximation to replication than the autonomic approach was. We attribute this difference to the fact that the RLearning dynamically changes the switching threshold between checkpointing and replication. Broadly, the RLearning approach tries other threshold values; if these values do not produce negative rewards, then they became adopted by the system. Thus, the tendency is the threshold to converge to a higher value that prevents the system to incur in an earlier than needed use of checkpointing.

Analyzing Figure 5, we observe how the fault tolerance strategies fare when the amount of resources decreased and the number of application increase. In this case, the checkpointing only remains as the worst approach. However, replication only is not the best approach if we deem the loss of running tasks as an undesirable event. In comparison to checkpointing and the autonomic approach, replication has lost 5% of the running applications (this loss occurs when all replicas are killed due to node failures).

Looking specifically to the autonomic versus the RLearning, we realize that the RLearning approach pursue the average completion time of the replication only strategy (which is the lowest of all approaches), while trying to avoid application loss. In our experiments this loss was less than 0,03%, a small value compared to the 5% obtained by replication. In this way, we judge that the RLearning approach arrived at a good tradeoff between average completion time and application successful completion, w.r.t. the other approaches.

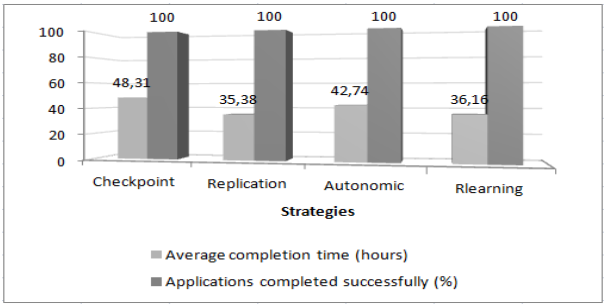

Figure 5. Scenario 2 – 700 nodes, 600 tasks and many faults.

Finally, we call the reader's attention to the fact that we have performed other simulation experiments than these that were reported. We have varied number of resources, applications and fault rates. In general, the results have shown that we obtain better levels of adaptation to the greed characteristics by using the RLearning over the autonomic approach.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have dealt with the problem of providing fault tolerance in opportunistic grid environment, by balancing the use of checkpointing and replication. Building upon the state of art, we have proposed the use of the MAPE-K model together with reinforcement learning as a viable approach to decide the exact point when checkpointing should be used instead of replication, and vice-versa. Our reinforcement learning approach was evaluated by means of simulation models developed by using AGST. The obtained results have corroborated our initial hypothesis that the switching threshold between checkpointing and replication should not be a fixed value, but may dependent on the amount of resources per applications and the reliability of the computing nodes composing the grid system.

Fault tolerance is a challenging problem. Currently, we are exploring the aspects of convergence versus continual policy modification lying at the heart of our approach. For this we are performing further experimental evaluation by means of simulation. As future work we plan to extend the approach to deal with other parameters discussed in [15]. For example, we can try to learn the number of replicas, or the interval between checkpoints. Finally, at long run, we also plan to experiment the approach in a real grid middleware. In this regard, we are thinking about the InteGrade middleware [12].

ACKNOWLEDGMENT

Thanks to the Graduate Program in Computer Science from the Federal University of Maranhão and the Foundation for Research Support of Maranhão (FAPEMA) for the financial support, Proc. BM-01440/13.

REFERENCES

[1] E. Alpaydin, Introduction to Machine Learning (2nd Ed). Cambridge, MA: MIT Press, 2010.

[2] M. Chtepen, F. H. A. Claeys, B. Dhoedt, F. Turck, P. Demeester, and P. A. Vanrolleghem, "Adaptive Task Checkpointing and Replication: Toward Efficient fault-tolerant grids," IEEE transactions on parallel and distributed systems, vol. 20, no. 2, Feb. 2009.

[3] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow," The Seventh International Conference on Autonomic and Autonomous Systems (ICAS 2011) IARIA, May 2011.

[4] A. Galstyan, K. Czajkowski, and K. Lerman, "Resource allocation in the grid using reinforcement learning," Third International Joint Conference on Autonomous Agents and Multiagent Systems – Vol. 3 (AAMAS 2004), IEEE Computer Society, Jul. 2004, pp. 1314-1315, doi:10.1109/AAMAS.2004.232

[5] B. T. Gomes, and F. J. Silva e Silva, "AGST - Autonomic Grid Simulation Tool. A Simulator of Autonomic Functions Based on the MAPE-K Model," First International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2011), Jul. 2011, pp. 354–359.

[6] S. Hariri, B. Khargharia, H. Chen, J. Yang and Y. Zhang, "The Autonomic Computing Paradigm," Cluster Computing, vol. 9, pp. 5–17, 2006.

[7] P. Jalote, Fault Tolerance in Distribuited Systems. Englewoods Cliffs, NJ: Prentice Hall, 1994.

[8] J. O. Kephart, and D. M. Chess, "The vision of autonomic computing," IEEE Computer, vol. 36, n. 1, pp. 41–50, Jan. 2003.

[9] F. Magoules, J. Pan, K. Tan, and A. Kumar, Introduction to Grid Computing. CRC Press, 2009.

[10] M. Parashar, Z. Li, H. Liu, V. Matossian, and C. Schmidt, "Enabling Autonomic Grid Applications: Requirements, Models and Infrastructure," In: Self-star Properties in Complex Information Systems (SELF-STAR 2004), Springer LNCS, vol. 3460, pp. 273-290, 2005.

[11] P. R. Prins, "Teaching parallel computing using beowulf clusters: a laboratory approach," J. Comput. Sci. Coll., vol. 20, n. 2, pp. 55–61, Dec. 2004.

[12] F. J. da Silva e Silva, F. Kon, A. Goldman, M. Finger, R. Y. Camargo, F. F. Castor, and F. M. Costa, "Application Execution Management on the InteGrade Opportunistic Grid Middleware," Journal of Parallel and Distributed Computing, vol. 70, n. 5, pp. 573–583, May 2010.

[13] R. S. Sutton, and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 1998.

[14] G. Tesauro, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies," IEEE Internet Computing, vol. 11, n. 1, pp. 22-30, 2007.

[15] A. E. Viana, An Autonomic Approach to Fault Tolerance in Running Applications on Desktop Grids. Master's thesis, Universidade Federal do Maranhão, São Luís, MA, Brasil, 2011.

[16] A. E. Viana, B. Gomes, J. Gonçalves, L. R. Coutinho, and F. J. Silva e Silva, "Design and Evalutation of Autonomic Fault Tolerance Strategies Using the AGST Autonomic Grid Simulator," Latin American Conference On High Performance Computing (CLCAR 2011), Sep. 2011.

[17] Z. Wang, X. Qiu, and T. Wang, "A Hybrid Reinforcement Learning Algorithm for Policy-based Autonomic Management," 9th International Conference on Service Systems and Service Management (ICSSSM 2012) IEEE, Jul. 2012, pp. 533–536, doi:10.1109/ICSSSM.2012.6252294

[18] Y. Wu, Y. Yuan, G. Yang, and W. Zheng, "An Adaptive Task-Level Fault-Tolerant Approach to Grid," Journal of Supercomputing, vol. 51, n. 2, pp. 97–114, Feb. 2010.

[19] Z. Zhai, "Grid Resource Selection Based on Reinforcement Learning," Int. Conference on Computer Application and System Modeling – Vol. 12 (ICCASM 2010) IEEE, Oct. 2010, pp. 644 – 647, doi:10.1109/ICCASM.2010.5622441

 ISBN: 978-1-61208-331-5

Using Performance Modelling for Autonomic Resource Allocation Strategies Analysis

Mehdi Sliem
MOVEP Laboratory, USTHB
Algiers, Algeria,
msliem@usthb.dz

Nabila Salmi
MOVEP Laboratory, USTHB
Algiers, Algeria,
LISTIC, Université de Savoie
Annecy le Vieux, France
nsalmi@usthb.dz

Malika Ioualalen
MOVEP Laboratory, USTHB
Algiers, Algeria,
mioualalen@usthb.dz

***Abstract*—Distributed resource allocation in data centers has gained a lot of attention from the research community in the last few years, especially in fields like cloud computing and multitier systems. It is usually expected that these systems deliver some performance guarantees to users' Service Level Agreements (SLAs). Therefore, data center servers may need to be dynamically redeployed to optimize some performance metrics so that to meet the promised SLAs. Moreover, the total profit of a system depends on its ability to reduce a data center's energy cost through the resources utilization optimization. The main challenge of resource allocation is then to find the minimum amount of resources that an application needs to meet the desired Quality of Service (QoS). In this direction, autonomic computing appears to be one of the most popular concepts to achieve these goals by means of self optimization. These properties provide a system with a dynamic optimization of its own resources use, and enable it to autonomously adapt itself to its environmental changes. However, such autonomic resource allocation strategies may result in a loss of performance or even service degradation under some conditions. Furthermore, it is interesting to predict the behaviour and the efficiency of those strategies, before applying a new resource allocation, to forecast the most appropriate configuration and ensure the effectiveness of the autonomic manager. Thus, we propose in this paper a general insight of performance modelling of resource allocation strategies using the modelling of an autonomic resource allocation server as an example. The modelling is based on stochastic Petri net models (SPN). We consider in our modelling dynamic allocation strategies, based on workload intensity and user mixes. Finally, we illustrate the effectiveness of our modelling through a set of experimental results.**

Keywords–Autonomic computing; data center; performance modelling; resource allocation

I. INTRODUCTION

Todays data centers are becoming increasingly large and complex, hosting a variety of business-critical applications with a set of QoS requirements, such as those for web hosting or e-commerce sites. The increasing demand for computing resources in a shared infrastructure creates the challenge of dynamic on-demand resource provisioning and allocation in response to variable workloads [1].

Data centers need then to have more flexible execution environments, allowing resources sharing between its different applications in order to meet performances requirements of those applications. In a cloud computing application for instance, the main objective is to maximize profits by an efficient use of resources, such as meeting the clients SLAs and reducing the energy cost of the data centre, by an efficient use of resources. Furthermore, modern Internet applications are implemented on multi-tier architectures, increasing the application's complexity. Each tier provides a defined service to the next tier and uses services from the previous tier. The resource allocation problem for multi-tiers applications is then harder than that for single tier applications: tiers may not be homogeneous and a performance bottleneck in one tier may decrease the overall profit.

The key-challenge of resources allocation is then, to provide enough resources to an application to meet its performance goals while avoiding an over-provisioning that could increase the energy cost and reduce the efficiency for other concurrent hosting (i.e., less resources for next applications). Due to these economic benefits, the resource provisioning optimization has been the subject of much investigations [2].

Some approaches focused on static allocation strategies that consider a fixed set of applications and resources, but these approaches have shown their weak efficiency because of the changing workload mixes. Other approaches are based on peak demand but suffer too from a lack of efficiency and a non cost-effectiveness due to their poor resource utilization during off-peak periods. In contrast, autonomic resource management may lead to efficient resource utilization and fast response in the presence of changing workloads.

In cloud applications, autonomic resource allocation provides application environments with self-configuration and self-optimization capabilities according to their environmental changes. The system can then be enforced through scale-up/down (i.e., adding/removing resources to individual Virtual Machines (VM)), scale-out/in (i.e., adding/removing VMs to an application environment), and migration (i.e., moving VMs over the physical infrastructure). This will directly impact both applications performances and the providers operation cost.

To achieve those autonomic computing features, IBM suggested a reference model for autonomic control loops, which is sometimes called the MAPE-K (Monitor, Analyse, Plan, Execute, Knowledge) loop [3]. The architecture dissects the loop into four parts that share knowledge:

- The *monitor* function provides mechanisms that collect, aggregate, filter and report details (such as metrics and topologies) collected from a managed resource.
- The *analyze* function correlates and models complex situations (for example, time-series forecasting and queuing models). The provided mechanisms allow the autonomic manager to learn about the Information Technology (IT) environment and help in predicting future situations.

 ISBN: 978-1-61208-331-5

- The *plan* function provides mechanisms that construct actions needed to achieve goals and objectives. The plan mechanism uses policy information to guide its work.
- The *execute* function provides launches the execution of a plan and controls it with considerations for dynamic updates.

Even if self-optimization may appear as an attractive way to enforce resource allocation process, reaching the goal of system improvement varies from an autonomic system to another, depending on the autonomic manager architecture and its implemented features. The targeted function to improve and the resource allocation process are also determinant criteria in building an efficient autonomic system. Furthermore, autonomic features are costly: they make systems more and more complex and need considerably more resources than other systems. So, attempting to improve a system with autonomic features may lead to some undesired configuration, with the introduction of new bugs or the loss of some vital settings, or else the degradation of the resulted configuration performances.

Hence, it is important to ensure that the chosen allocation strategy offer the desired performances under different circumstances while designing the system. This requires to predict and measure performances of an autonomic manager and its possible impact on the current system, before applying a solution or a reconfiguration. The main goal is to know how long the decision making process will take and how much the system's performances will be improved. The autonomic resource allocation has also to be compared with static resource allocation according to the application case. In this direction, formal methods are strong tools for system performance prediction based on modelling. Mathematical models, such as Petri nets [4] are well suitable for modelling distributed systems and fit the autonomic computing process with the system operation.

We attempt, in this paper, to define a general modelling of autonomic resource allocation using Stochastic Petri Nets (SPN). This modelling is presented through a simple typical example of an autonomic resource allocation system. We consider for our modelling a dynamic allocation strategy, based on workload mixes analysis, applied by an autonomic server. Some static allocation strategies are defined, each one assigning a fixed amount of resources to user requests. The system will work initially with a predefined strategy, while the autonomic manager analyzes continuously workload mixes. The autonomic manager will, then, reconfigure the system to move to a more appropriate strategy, whenever there are pending user requests while other requests hold several resources. So, according to the analysis of the monitored data, the autonomic manager redistributes fairly the existent resources.

The paper is organized as follows: Section II discusses related work. Then, Section III presents our general modelling and the experimental results. Finally, Section IV concludes the paper and gives future work.

II. Related Work

Significant attention has been given to the topic of distributed resource allocation in the last few years. The main studied issue is the cost, efficiency and the generated profits of the used methods, especially when client Service Level Agreements (SLAs) must be satisfied. Several work have then been proposed to use autonomic computing, while using predictive models. We provide, in the following, the most relevant prior work.

As Clouds are complex, large-scale, and heterogeneous distributed systems, resource allocation is one of the main topics of interest studied in the last few years in the context of cloud computing. Thus, Ghanbari et al.'s, results [2] provide valuable insights on the performance of alternative resource allocation strategies and job scheduling disciplines for a cloud computing infrastructure. The service level agreement is based on a response time distribution, which is more relevant than the mean response time with respect to performance requirements of interactive applications. The authors developed an efficient and effective algorithm to determine the allocation strategy that results in a smallest number of required servers. They have also developed a novel scheduling discipline, called probability dependent priority, which is superior to First Come First Served (FCFS) and head-of-the-line priority in terms of requiring the smallest number of servers. The authors consider in their work the case of two job classes.

In the same direction, Buyya et al. [5] identifies open issues in autonomic resource provisioning and presents innovative management techniques for supporting Software as a Service (SaaS) applications hosted on Clouds. The authors present a conceptual architecture and early results highlighting the benefits of Clouds autonomic management. They presented the first steps towards an autonomic Cloud platform able to handle many of the above problems. Such a platform will be able to dynamically supply applications with Cloud resources in such a way that Quality of Service user expectations are met with an amount of resources that optimizes the energy consumption required to run the application.

In [6], an SLA-based resource allocation problem for multi-tier applications in the field of cloud computing is considered by Goudarzi and Pedram. An upper bound is provided on the total profit and an algorithm based on force-directed search is proposed to solve the problem. Processing, memory requirements and communication resources are considered as three dimensions in which optimization is performed. In [1], the purpose was to demonstrate the advantage of "adaptive" models, relative to "static" models in optimization. Hu et al. investigated model based optimization of a private cloud where applications are clustered across a known homogeneous set of physical machines. They modified resource sharing of applications, to minimize SLA violations. The focus was only on response time, considering that multiple service level objectives will not change the approach, but just the complexity of solving the optimization problem. The main contribution of this work was using dynamically tracking models (for each application) within the global optimization loop. These models update themselves at runtime in order to adapt to environment perturbations, not captured in initial model specification.

Workload variation and its resource consumption is also an important point to study for the resource management, in this direction, Litoiu [7] investigates performance analysis techniques to be used by the autonomic manager. The workload complexity was studied, and algorithms were proposed for computing performance metrics bounds for distributed transactional systems under asymptotic and non-asymptotic

 ISBN: 978-1-61208-331-5

conditions, with saturated and non-saturated resources respectively. The proposed technique makes use of linear and non-linear programming models and their performance evaluation. Workloads are characterized by their intensity representing the total number of users in the system, and workload mixes, which depict the number of users in each service class.

Finally, some authors investigate the use of mathematical and performance modelling or optimization approaches to improve the resource allocation. Bennani and Menasce [8] addressed the resource allocation problem in autonomic data centers. The presented solution is based on the use of analytic queuing network models combined with combinatorial search techniques. The authors have shown how analytic performance models can be used in an efficient manner, to design controllers that dynamically switch servers from one application environment to another as needed.

In [9], Xu et al. propose a two-level resource management system to dynamically allocate resources to individual virtual containers. It uses local controllers at the virtual-container level and a global controller at the resource-pool level. An important advantage of this two-level control architecture is that it allows independent controller designs for separately optimizing applications performances and the resources use. Autonomic resource allocation is realized through the interaction of the local and global controllers. A novelty of the local controller designs is their use of fuzzy logic-based approaches to efficiently and robustly deal with the complexity and uncertainties of dynamically changing workloads and resource usage. The global controller determines the resource allocation based on a proposed profit model, with the goal of maximizing the total profit of the data center.

Regarding these proposals, we notice that most of them use formal modelling for an autonomic online optimization, but only few work focused on the efficiency of those autonomic components and performance modelling and prediction of a whole autonomic system in the context of resource allocation.

We introduce, then, in this paper, a general performance modelling and analysis approach for the autonomic resource allocation problem. The main idea is to model the complete autonomic system behaviour including resource management and allocation and the autonomic loop. A system configuration is seen as a distribution of system resources for user requests and their availability. We illustrate then our modelling methodology through a simple typical example of an autonomic server and a series of experimental results.

III. An autonomic resource allocation Platform

To explain our modelling approach, we choose a typical example of an autonomic resource allocation system, based on a simple monitoring process. We first, in this section, define the architecture of our example, then we give its general functionalities which have to be considered in our modelling. After that,we explain our modelling methodology through the presentation of our model based on Stochastic Petri Nets (SPN) [10].

A. System architecture

Our system consists of an autonomic server with a set of resources and an allocation strategy for those resources based on a monitoring process. The monitoring process aims to determine the amount of resources to allocate to the next user requests.

The main managed element in an autonomic resource allocation system is the resource. A resource may be any element used or invoked during the processing of a user request: it may be a server, a processor cycle, a memory space, a used device, a network bandwidth, an available component, and so on. An allocation strategy defines the amount of resources to allocate to a new requests: it could be a static allocation approach considering a fixed set of resources or a dynamic one adapting the resource management strategy according to the system's state and the user SLAs.

The autonomic loop aim at self-optimize system performances through self-configurations, by switching between allocation strategies relatively to the changing workload. We take for our example a simple self-optimization technique based on the current workload mixes, a continuous monitoring of processing services is done by the autonomic manager, the autonomic loop is, then, triggered periodically to analyze the monitored data, the analysis and plan phases determine the most present service's class in the system and choose the most appropriate strategy for this current class for the next requests. Finally, the act phase switches from the current strategy to the new one and reset the autonomic loop periodicity.

B. System features to model

To show how we operate to model an autonomic system, we take generic concepts of an autonomic resource allocation system. However, to have a reliable trustworthy model with accurate results, some key concepts have to appear in our modelling :

a) System's resources and allocation strategies: We need, in our modelling, to represent the system resources, their states and their distribution over the time and their allocation mode. A resource may be a server, a processor cycle, a memory space, a network bandwidth, and so on. A system configuration is then seen as a state where resources are allocated to different user requests according to a predefined strategy.

For instance, the system may reach or approach a saturation state, becoming unefficient to process new requests; it may also be in a an unoptimized state not satisfying a required SLA, even if the available resources are sufficient. It may also give the expected processing performance while producing a high consumption cost. All these drawbacks or failures depend on the allocation strategy and its efficiency to optimally distribute the system's resources.

b) Classes of service: As presented in [7], the system performance and the saturation state do not depend only on the workload intensity (number of users in the system), but also on the workload mixes that represent the users number of each service class. Classifying user requests into different classes allows us to separate them according to a set of specificities that may affect differently the system behaviour and its performances. In fact, a request which needs a database access in a multi-tier system do not have the same impact on the system's performance as a request invoking only an application process in the same system. In our work, we consider different service classes classified according to the needed number of identical resources (i.e., several CPUs or

 ISBN: 978-1-61208-331-5

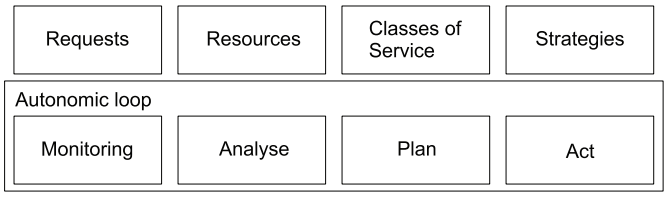

Figure 1: Key concepts to appear in the model

more memory space): each class requires a given number of resources. This consideration comes from the fact that a job holding a set of resources ends more quickly than if it had only one resource. Contrariwise, allocating more than needed resources to a client will waste additional resources for the same job performances while increasing costs and reducing concurrency processing for other jobs. Hence, resource provisioning and the resulting system configuration vary according to the workload mixes in the system.

c) Self-optimization strategy: As explained before, an allocation strategy have to be defined according to the workload mixes. Modelling static and autonomic self-optimization approaches will allow to compare the efficiency and inconveniences of each resource management manner for each system configuration. Hence, for an autonomic approach, it is important to represent the system and its autonomic loop in the same model, this allows to measure autonomic features impact on the system while measuring the system performance itself. Grouping the system functioning and the autonomic loop in the same model is also more appropriate to make tests under different cases: different system configurations can then be used to test the autonomic loop influence; the system's state and its evolution over time being visible directly in the model.

C. Modelling our autonomic architecture

1) Used formalism: Stochastic Petri Nets: Stochastic Petri Nets (SPN) [4] are a stochastic temporal extension of Petri Nets, widely used for performance analysis of complex systems. Our choice of this formalism is first motivated by the fact that we need a state based model to be able to evaluate performance indices related to system configurations (number of requests in some part of the system, mean usage time of some resource, etc.). Petri Nets are state based models, which are well known for being able to model complex systems with concurrency and conflicts, and the stochastic extension allows to do a performance analysis based on a Markov chain-like state space graph. This is in contrast to other performance formalisms like Queuing networks or process algebras models where conflicts cannot be modelled. Moreover, as our methodology is incremental, we need to compose sub-models to connect multi-tier sub-models, or servers sub-models. In this direction, Petri Nets are compositional models and interaction between Petri nets representing different parts of a system may be easily defined as transitions or places, which are merged when interacting sub-models are composed. So, we define an SPN in the following.

Definition 1 (Stochastic Petri Net)**.** *A Stochastic Petri Net is a couple $\mathcal{N} = (\mathcal{N}'; \theta)$ where:*

- *$\mathcal{N}' = (P, T, Pre, Post, Inh, Pri)$ is a Petri Net where P is a set of places, T a set of timed transitions with a stochastic firing delay, Pre, $Post$, Inh are respectively the precondition, postcondition and inhibition functions relating transitions to places, and Pri the transition priority function.*
- *$\theta : T \times Bag(P) \rightarrow \mathbb{R}^+$ where $\theta(t, M)$ is the firing rate of t in M, i.e, the parameter of its exponential law, where $\theta(t)$ represents:*
  - *The weight of t if $Pri(t) > 0$ (t is immediate).*
  - *The firing rate of t if $Pri(t) = 0$ (t is timed): the enabling duration before the firing of $t(c, M)$ follows an exponential probability distribution with mean $\theta(t)$.*

2) System's modelling: Regarding to the system features given before, we follow the principles below to model the autonomic resource allocation system:

- We first model the requests arrival, a request being any kind of a client service invocation, applied to different data center applications.
- We then represent system resources, where we abstract the kind of resources as for requests.
- Resources are allocated to requests according to an allocation strategy. As in our system, different strategies are considered, this requires to model each strategy. However, only one strategy have to be applied at one time.
- After the allocation of the required resources to a request, the service processing in progress is modelled, as well as the different service classes to represent resources consumption and the obtained performances
- Finally, as our example is based on an autonomic system, we need to model each phase of the autonomic loop: the continuous monitoring of processing services, the autonomic loop triggering, the analysis and plan phases, which chooses the most appropriate strategy for the next requests, and the act phase which switches from the current strategy to the new one.

The modelling methodology explained above is depicted in Figure 1, giving the skeleton of a general resource allocation autonomic system model. This figure shows the key concepts that should be considered when modelling the system. Each part of the system is then replaced with the appropriate sub-model according to the real system to analyze. The merging of the obtained sub-models gives the whole model of the autonomic system. We give next the proposed sub-models for an autonomic server.

Figure 2: The request submitting sub-model

 ISBN: 978-1-61208-331-5

Figure 3: The resource allocation strategies sub-model

a) The request submitting sub-model: The request is modelled with a place named $Requests$, marked with a number of tokens corresponding to a possible number of user requests. The system processes a request (represented by a token) and answers the user, then the token is put back into the $Requests$ place to model a new request arrival and keep the testing possibility of the whole system. The transition $NewReq$ is used to model the arrival and queueing of the request in the system (see figure 2). This transition is characterized with a firing rate modelling the arrival request rate.

b) The resource allocation strategies sub-model: Each allocation strategy is modelled by a place. We consider in our model, three strategies: the first one for allocating one resource to new requests, modelled by a place named $Strat-1$; the second one for allocating two resources modelled by a place named $Strat-2$; and the last one for allocating three resources, represented by a place named $Strat-3$. This model is shown in Figure 3.

c) Servers sub-model: The model of servers is composed of two parts:

- The server requests queue part: It is modelled by a first place, named $Queue$, connected to the $Requests$ place. The requests are sorted into different classes to represent different execution processing times, giving a number of places equal to the number of service classes: we associate a place for each request class to model resource allocation to requests of the corresponding class.
- The servers parts: each server of the system is modelled by a place, named $Server$, containing the server's resources (see Figure 4). The request processing is modelled by three transitions and three places for each service class: The transitions are named $Begin-Cli-1$, $Begin-Cli-2$, $Begin-Cli-3$, representing the processing beginning for each current allocation strategy; The places, named $Cl1i-1-exe$, $Cli-2-exe$, $Cli-3-exe$ model the different request execution states. Three other transitions, named $End-Cli-1$, $Cli-1$, $End-Cli-2$, $End-Cli-3$, are used to model the request processing end (see Figure 4). The $Begin-Cli-s$ transitions are linked to the $Server$, $Queue$, $C1i-exe$ and $Strat-i$ places. The $End-Cli-s$ transition consumes resource tokens from the $Cli-s-exe$ place, and puts them back in the $Server$ place, and by the way, puts back the request to the $Requests$ place. This server sub-model represents a request processing according to a predefined execution time, given by the firing rates associated to the $Begin-Cli-s$ transition. These rates are fixed before analyzing the model, according to the system under test. As many user classes are defined in our model, this sub-model is duplicated for each user class with a slight modification of the required resources number to consume to get the best performances. Different processing rates are associated to their corresponding transitions, to show a better processing time when using more resources. Hence, only one beginning transition can be fired at one time in the same service class, according to the current strategy.

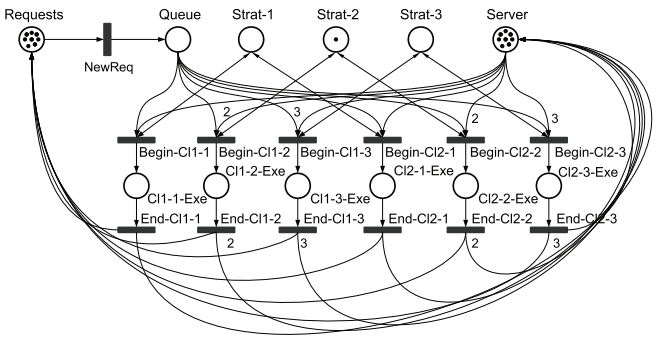

Figure 4: The server sub-model

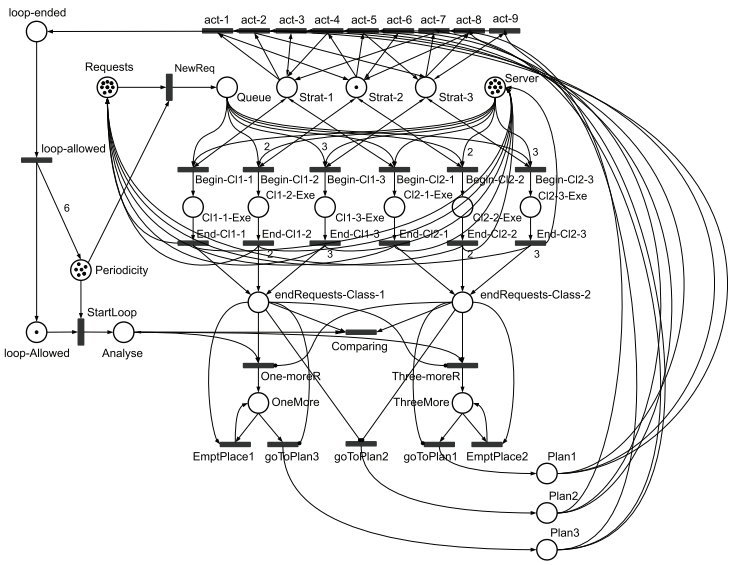

Figure 5: Global system model

d) The autonomic loop sub-model: The MAPE-K loop in our modelling is composed of five parts:

- The first part models the monitoring phase of the autonomic loop. As in our example, the self-optimization is based on the amount of requests in each service class, we model the monitoring by a place named, $endRequests-Class-i$, for each service class, each of these places is connected to $End-Cli-s$ places belonging to the corresponding service class (see Figure 5).
- Unlike the monitoring phase, which is performed continuously, other phases of the autonomic loop are, in our example, triggered periodically. In Figure 5, a place, named $periodicity$, models the periodicity of the autonomic loop by its number of tokens. A token

is taken periodically by the $newRequest$ transition. The place named $loopallowed$ contains the token that will represent the progression of the MAPE-K loop execution through the model and ensure that is executed once at a time.

- The analyze phase sub-model aims to empty all monitoring places, to reset them for the next period while computing which service class is the most frequent in the system in the last period. Hence, a place, named $analyse$, models the beginning of the phase. A transition, named $comparing$, takes a token from each monitoring place until some of them become empty, the $one - moreR$ and $three - moreR$ transitions are then used to continue the same process for the remaining monitoring places containing tokens. Moreover, inhibitor arcs are used to guide the analyze phase towards the appropriate transitions, when some monitoring places are empty. Once only one place is containing tokens, $OneMore$, $EmptPlace1$ or $ThreeMore$, $emptPlace2$ places are used to reset the corresponding place to an empty state. Finally, transitions $goToPlan1$ and $goToPlan2$ guide the autonomic loop to the associated reconfiguration plan. The transition $goToPlan3$ is fired when both service classes get the same number of requests in the last period (see figure 5).

- The plan phase sub-model is consisting of a set of places whose number is equal to the number of service classes. Each place represents the most appearing class in the last period ($Plan1$, $Plan3$). Moreover, other places have to be added to the plan, to consider all equality cases between the class requests: in our example, the $Plan2$ place models the equality case between the two service classes of our system (see Figure 5).

- Finally, the last part of the MAPE-K loop to model is the act phase. We model it by a set of transitions representing all combinations between the current applied strategy and the current computed plan. The new strategy, indeed, have to be chosen based on these two parameters, for instance, if the current strategy allocates three tokens (resources) to user requests while these requests belong to the first service class, the current strategy is then considered as non-cost-efficient, as it allows two more resources for the same service performance. A unique transition may be then fired, switching the strategy to the more appropriate place $strat-1$. The same process is repeated for each combination by a particular transition $act-i$. A place $loop-ended$ is reached after the firing of one of the act transitions; the transition $loop-allowed$ reset, then the loop periodicity and puts back the loop progression token to it initial place ($loopallowed$).

Figure 5 shows the general model of the autonomic server. The obtained model can be tested using different number of classes and periodicities for each test, to get more accurate results or to identify the best requests classification. In the next section, we analyze the obtained SPN and try to predict the modelled server behaviour under different configurations.

TABLE I: TRANSITION RATES OF CLASSES OF SERVICE

Config	Transition	Rate value
1	Begin-Cl1-strat	1.00
1	Begin-Cl2-1	0.33
1	Begin-Cl2-2	0.66
1	Begin-Cl2-3	1.00

D. Experimental results

The final model presented in Section III was analyzed using the GreatSPN package [10], on an Ubuntu linux 12.4 LTS workstation with 4 GB of RAM.

For performance analysis, we used various configurations of the system, obtained by varying the initial markings of the requests number, the available resources and the loop periodicity when using dynamic strategy. The model used in our experiments contains two service classes: the first one needs one resource from the server's place, while the second one uses three resources. We performed our tests under each of the three static allocation strategies and a dynamic strategy using the autonomic loop.

Table I shows the transitions rates, depending on the number of consumed resources. Only the second service class transitions are affected by the available resources. Transitions not appearing in this table have rate 1 (i.e., faster than all other transitions, rates being given in the same unit).

To compute performances, we vary the number of requests and available resources under each configuration. The GreatSPN tool [10] computes the state space and its steady-state probabilities. We study the evolution of several metrics from obtained steady-state probabilities. To evaluate the efficiency of a configuration, we interested in the following metrics:

- The response time for a user request, being of the first or second class.
- The throughput of processed requests.
- The relationship between response time and resource consumption.

We first analyze the SPN for a fixed number of requests (10 in our example), with varying the resources number of the server (1 to 40). Figure 6 shows that the mean response time of the static strategies is unchanged under increasing the number of available resources, while dynamic resource allocation slightly improves it taking advantage from the powerful autonomic system. However, the response time of the autonomic strategy remains worse than static ones that allocate more resources.

Figure 6 depicts the throughput of processed requests. Static strategy allocating one resource gives the best throughput, but is exceeded by other static strategies allocating more resources from a certain threshold of available resources. The autonomic resource allocation gives the worse throughput, which is partly due to the autonomic loop processing searching the best strategy. The results can be improved using a separated server for the autonomic manager. The efficiency of a given strategy in our case depends, though, more on its ability to reduce the final system's cost.

We were also interested in evaluating the total resource consumption of the system. The results shown in Figure 6 show

 ISBN: 978-1-61208-331-5

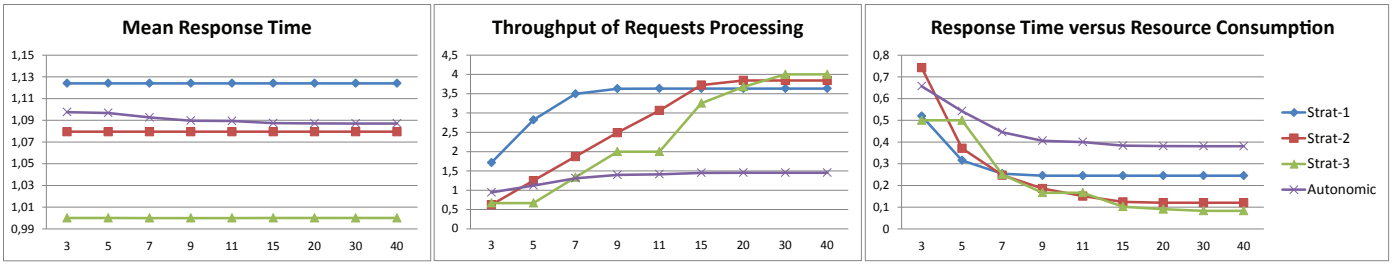

Figure 6: Experimental results depending on available resources

that the dynamic resource allocation gets the best ratio between the response time and the resource consumption metric, but it is also due to a less number of processed requests, as it is shown in Figure 6.

We can conclude that, with these different analysis results, an autonomic allocation strategy gives a better relationship between the response time and the resource consumption, and so less final user cost. It allows, though, to process less requests, which may make the system less effective. The most efficient strategy to use for a particular system, depends then, on the different costs related to each metric and the required expected quality of service.

IV. Conclusion and Future Work

This paper addresses a general modelling of resource allocation strategies for an autonomic server using SPN. The objective is to show, through the chosen example, how we can gradually build a formal model for a resource allocation autonomic system, to be able to analyze it and think about its performances and efficiency. For this purpose, we have studied the most important concepts to consider when modelling a particular system to obtain a reliable model, then construct the global model of the system example, basing on the building of sub-models representing parts of the system, then the merging of constructed sub-models.

To finalize the study, we compared different models of static and autonomic strategies with the aim of forecasting the more appropriate allocation strategy for the given system.

Regarding the obtained results of our modelling methodology, more research work is still required in several directions, among which: considering workload mixes in the request arrival modelling, modelling more specific applications of resource allocation systems, with specific requirements, such as a cloud computing system. Finally, more specific autonomic resource allocation strategies using different techniques have to be compared.

References

[1] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, "Resource provisioning for cloud computing," *Conference of the Center for Advanced Studies on Collaborative Research*, pp. 101–111, 2009.

[2] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai, "Feedback-based optimization of a private cloud," *IBM Canada*, vol. 28, pp. 10–111, January 2012.

[3] IBM, "An architectural blueprint for autonomic computing (fourth edition)," 2006.

[4] S. Natkin, "Stochastic petri nets and their application for computer systems evaluation," Ph.D. dissertation, CNAM, Paris, France, juin 1980.

[5] R. Buyya, R. N.Calheiros, and X. Li, "Autonomic cloud computing: Open challenges and architectural elements," *Third International Conference on Emerging Applications of Information Technology (EAIT)*, pp. 3–10, 2012.

[6] H. Goudarzi and M. Pedram, "Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems," *IEEE 4th International Conference on Cloud Computing*, July 2011.

[7] M. Litoiu, "A performance analysis method for autonomic computing systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 2, 2007.

[8] M. N. Bennani and D. A. Menasce, "Resource allocation for autonomic data centers using analytic performance models," *2005 IEEE Intl'. Conf. on Autonomic Computing, Seattle, Washington*, pp. 229–240, June 2005.

[9] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Comput'08*, vol. 11, pp. 213–227, 2008.

[10] S. Baarir, M. Beccuti, D. Cerotti, M. D. Pierro, S. Donatelli, and G. Franceschinis, "The greatspn tool: Recent enhancements," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, pp. 4–9, March 2009.

 ISBN: 978-1-61208-331-5

Operation of Accumulator-Bank Serving Agent System Using Machine Learning

Ágnes Werner-Stark, Tibor Dulai and Katalin M. Hangos
Department of Electrical Engineering and Information Systems
University of Pannonia
Veszprém, Hungary
Email: werner.agnes@virt.uni-pannon.hu, dulai.tibor@virt.uni-pannon.hu

***Abstract*—Advancements in on-demand power management of renewable energy can be achieved by multi-agent systems. This paper proposes an innovative approach where a population of autonomous agents are able to cooperate in managing an accumulator-bank in order to effectively deliver energy in places where it is required. The distributed and adaptive multi-agent approach is able to decrease the interferences by avoiding the negative interactions and conflicts, using the cooperation among agents. Our method uses the learning ability of agents to minimize the number of communications among agents and the central unit. This adaptive behavior lets the agents minimize the time to find the optimal routes during the search. A simulation environment has also been developed for visualizing the movements of the agents and the conflict situations. The operation and the efficiency of the algorithm have been investigated using simple case studies.**

Keywords–renewable energy; agent; genetic algorithm (GA); cooperation

I. INTRODUCTION

It is widely recognized (see e.g. [3]), that ”researchers must find a sustainable way of providing the power our modern lifestyles demand.” Along this line, more and more research and development projects are aimed at modernizing energy sources e.g., [9], [4]. Hadjipascalis *et al.* [10] present an overview of the current and future energy storage technologies used for electric power applications.

The problems of applying renewable energy sources are widely and extensively investigated because of the need for sustainability of energy systems. Solar and wind power are the two main sources of renewable energy, both of them suffer from the disadvantage that they are not always readily available on demand. At the same time, the available ways of energy storing are not economic enough and are of limited capacity. One of the possible ways of storing the energy is using accumulators that can be placed in an *accumulator-bank*. An automated service of such an accumulator-bank is desirable that can be implemented by using *autonomous agents* (i.e., robots) that can cooperate with each other to provide optimal on-demand service.

Literature on learning abilities of agents: Usually, in a multi-agent system the agents have specific pre-defined abilities to perform a certain task. One of the challenges of a multi-agent system is to develop agents with the ability to learn their behaviour from each others. In recent years, more and more researchers focus on the learning ability of agents that can improve the efficiency of a multi-agent system. Saggar *et al.* [19] developed a learning algorithm for agents to optimize walks in both speed and stability in order to improve a robot’s visual object recognition. Nguyen-Thinh presented a learning algorithm for agents based on interactions with humans in conflict situations [14]. Taylor et al. [21] present an algorithm that combines transfer learning, learning from demonstration and reinforcement learning to achieve rapid learning and high performance in complex domains. Using experiments in a simulated robot in soccer domain, they show that human demonstrations transferred into a baseline policy for an agent and refined reinforcement learning significantly improve both learning time and policy performance.

Learning by using GA: Genetic algorithms (GA) are popular tools for implementing heuristic learning policies. In the context of robot movements, GA is applied for route planning using the variants of the well-known Vehicle Routing Problem (VRP) with the help of other heuristic methods. These methods are called hybrid GAs [5],[6],[12],[16],[17],[20], where the improvement can be achieved by imitating biological evolution for solutions of VRP [2],[8],[15],[18]. It is important to emphasize that hybrid GA methods are used for improving the result starting from an initial - usually not optimal - solution. Hagen et al. [11] present the implementation of a GA based path planning on RoboCup’s small-size league robots. Because path planning on mobile robots is a continuous process, the path planning runs until the robot arrives at its destination. Hereby, the path is updated according to the environmental changes, such as moving obstacles.

Cooperation: Another way of improving the reactivity of an agent system is to develop the cooperation ability of its agents. A multi-agent approach was presented in [7], that uses cooperation among the agents, task decomposition and task allocation, and decentralized planning. The paper [1] proposes a solution approach of managing roadway network congestion over time based on cooperative multi-agent-based principled negotiation between agents. In our recent study [22] we proposed a cooperative optimal route planning algorithm in the accumulator-bank servicing model by using a specially constructed model that will be extended by a learning method in this paper.

Learning and cooperation in renewable energy systems: Advancements in on-demand power management of renewable energy can also be achieved by multi-agent systems. Many researchers have used this technology recently. In [23], the effectiveness of the coordination model was analysed by investigating the effect of the environmental conditions that affect the traveling time. Their approach is based on a multi-agent system for a road transportation network using supply chain management. Hrncir et al. [13] present the problem of finding parts of routes, which can be shared by several travelers with different points of departure and destinations. This is a complex multi-agent problem, for which a special method

 ISBN: 978-1-61208-331-5

can be developed. They proposed a three-phase algorithm that utilizes performance of single-agent planners to find individual plans in a simplified domain first, then merges them using a best-response planner. This planner ensures that the resulting solutions are individually rational, and then maps the resulting plan onto the full temporal planning domain to schedule actual journeys.

The aim: Based on our cooperative optimal route planning algorithm [22], we aim at developing a novel approach that allows autonomous agents to carry out learning in conflict situations through communication with each others but without the interaction with any human during the operation. Our approach uses the learning ability of agents to minimize the number of communications among agents that is necessary to the quick service in an accumulator-bank. The learning is realized by using GA tailored to this special problem.

II. BASIC NOTIONS AND TOOLS

A. The model of the accumulator bank and the basic route planning algorithm

In our recent study [22] we proposed a cooperative optimal route planning algorithm in the accumulator-bank servicing model by using a specially constructed model that guarantees the avoidance of collisions. This approach is called the basic route planning algorithm that will be briefly described here.

In order to have a simple model of the accumulator-bank, we separate the storage place into cells of equal size so that a transport agent can fit in one cell. These cells are arranged in a matrix that will be the most important helping tool for the transport of the agents: moving from cell to cell to get from one place to another, and this logical unit will also be used to avoid collision with other agents.

The basic route planning algorithm minimizes the cost that is the number of covered cells from the start to the destination. We added certain cost of every 90 degrees turns made between cells, too. (Figure 1 illustrates the cost calculation of a path with turning.)

Figure 2 shows a possible, simple cell matrix (this is the map for the accumulator-bank) with the costs in the cells. The green cell is the starting point, the blue cell is the end point and the red cells mark obstacles (wall/rack).

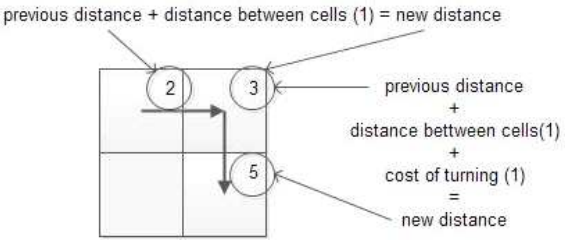

Figure 1: Cost calculation

B. Communication in the basic route planning algorithm

In the basic route planning algorithm [22], each agent navigates not only avoiding collisions with each other but do this in an optimal way. At the same time we assumed that the agents receive and send the information directly from/to a central processing unit. This is not realistic/economical in practical situations, but a wireless technology should be considered instead for communication that allows direct communication among the agents, too. (We selected the TCP protocol because this can be applied on WIFI and Bluetooth technologies, so the adaptation will be easier for future devices.)

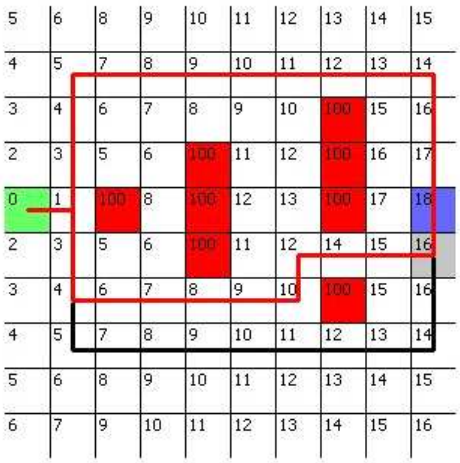

Figure 2: Matrix-based representation of a storage place of a simple example [22]

The communication system has to deliver to the agents the data necessary for the route planning:

1) the store layout (the map), that shows, which cells it can go through and which are prohibited; and
2) the planned route of the other agents (the cell-reservations), on the basis of which it can determine, which cell is free, and when or how long it needs to wait for which other agent. We can send these information in wireless way to the agents.

C. Synchronization

Because every agent should communicate with the central unit, it is practical to build up a connection when the agent connects the system, this connection is reserved continuously after that. Every agent needs a personal identifier. When an agent starts its operation, it connects to a predefined server, from where the agent asks a serial number and a connection identifier. On the server side a separate thread waits for the agent on a specific IP-address and port. After connection this thread manages the communication between the agent and the central unit.

In this process, synchronization plays an important role. If two agents join the name-server at the same time, one of the connection requests is forced to wait until serving the other, but we assume that this waiting time is negligible compared to the operation times.

Figure 3 illustrates the exchange of the necessary information to build up the connection and to plan the route. The new agent - placed in the system - is connected to the name-server, after that it queries its destined connector reach (IP-address

 ISBN: 978-1-61208-331-5

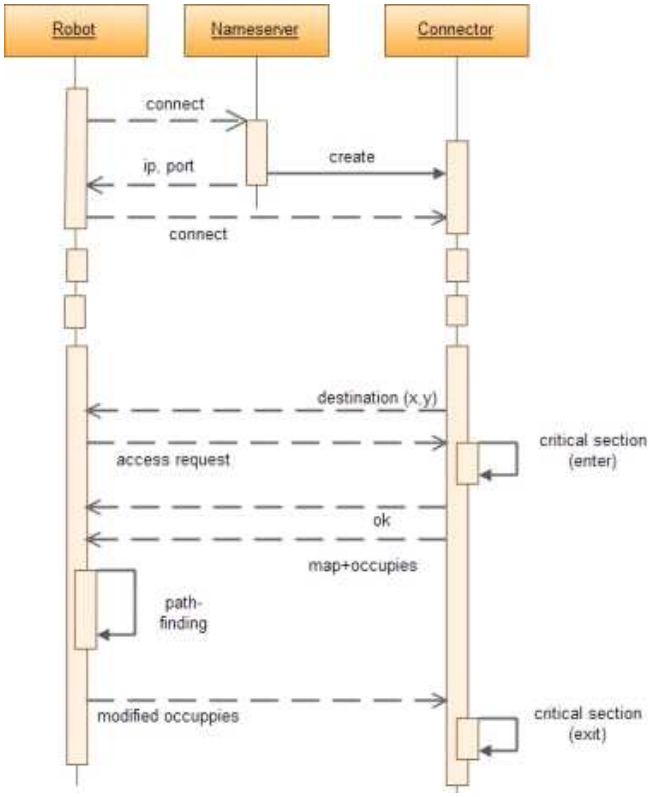

Figure 3: Building up of the connection and the exchange of information that is necessary for planning

and port) then it closes this connection. The agent builds up another connection with its connector then it waits for its task. The connector gives a task to the agent submitting its goal (with x and y coordinate pair). After that the agent asks for an access to the map for reading. Then, the first synchronized function is executed on the connector side, which induces the agent to wait until the map is used by another agent. As soon as the resource gets free, the connector grants the request and sends the necessary data: the map and the seizing. Then the agent can plan its route while taking all other agents' planned movements into consideration. As soon as the agent has performed planning it sends back the modified seizing. As a result the connector logs out of the critical section and the other agents can reach the map again. It is important that the agent can only reach the map for a given short time period.

There may be a situation in which two agents both read the seizing and thereafter they both load their calculated results back but these are in conflict (because neither could plan the other agent's route at the retrieving of the seizing). That is why we apply mutual exclusion, which forces agents to wait, but each agent has to plan their route only once. Finally, the agent sends back its route when it reaches the destination, so it no longer seizes the common resources.

D. Inter-agent communication

Until now, a simple policy of ordering the route planning of agents has been followed: the agent that gets access earlier can send back its seizing first, and from that point it cannot be changed by the later arriving agents. So if the first agent seized itself a route, the other agents had to respect it. However, this policy may not be an optimal one. For example, it may happen that if an earlier planning agent were yielding precedence to a later arriving agent, than the agents could reach their destinations quicker. Figure 4 shows an example of such a situation. The first agent planned earlier (the blue arrow marks its motion) so this agent can pass through passage first. The second agent (the green arrow marks it) passes through this passage too, however it has to wait until the first agent leaves the passage. We suppose that they start at the same time from the starting point of the arrow representing them. The second agent is nearer to the entrance of the passage so the agent waits more for the first agent than the first agent would wait for the second agent if they had reserved their route in reverse order. Negotiating is optimal when the agents can directly exchange data with each other: passing data through the central server takes up twice as much time than sending the data directly.

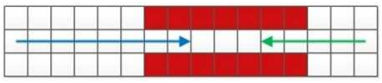

Figure 4: An example of the priority giving

E. Unexpected events

If an agent cannot continue the way for some reason, but the communication unit is operable, it has to report its break down to the central unit. The central unit then deletes every seizing of the broken down agent, thereafter it informs every agent whose path passes through that cell on which the agent is broken down and waiting. The affected agents then redesign their route and bring it again to the central unit.

Of course, a braking down failure can occur in such a way that causes the complete stop of the agent and it cannot signal its state. We will not deal with this problem in this paper.

F. Advance planning

An agent can move on one route, but we need to pay attention to certain situations meanwhile, at the same time. For example:

1) There may be situations in which it is simply not enough to avoid another agent because for example the agent takes up a bottleneck passage and the other passage is too far. At that time it is more appropriate to wait for the passing of another agent than to choose a bypass route.
2) There can be some narrow passages in the storage for the sake of better utilization of space, therefore we also need to deal with them. In these passages there can be one agent at a time, this can cause a traffic-jam. If two agents approach the passage at its opposite ends then the route search algorithm can sense only the character of the problem before the collisions.

The possible collisions can be detected in advance, not locally. This requires communication among agents but if every agent communicates with each other then it is a lot of time that can

 ISBN: 978-1-61208-331-5

slow down the operation. The learning ability of agents may help in such situations.

III. ADVANCED ALGORITHM WITH MACHINE LEARNING

The proposed algorithm that uses machine learning implemented as a GA problem is described in this section that enables the agents to learn in order to minimize the number of communications.

A. Learning ability of agents

The basic route planning algorithm provided means on how two agents can yield precedence to each other [22]. With the cooperative communication ability of the agents available in the advanced algorithm, they can determine which agent is worthwhile to contact. If every agent established contact with every other agent at the planning, it would increase the planning time $n-1$ fold in case of n agents. The learning ability of the agents is used to minimize the need for communication, i.e., to limit the number of negotiations among agents. For this it is necessary to determine, which agents have a potential conflict against which other ones that should be contacted.

For this purpose we store the following *data* about a *collision*:

- *coordinate*: where the collision occurred
- *agentID*: identifier of the collided agent
- *distance_covered*: the time of the traveled route to collision (an expected value)
- *waiting*: how much the agent waited in the cell when the collision occurred
- *shut_down*: it is true, if the thread was shut down because of the collision; it is false, if we have to wait longer to avoid collision
- *on_route*: Is the coordinate on one of the optimal route? (true/false)
- *manh_distance*: manhattan distance from the target
- *estimation*: a specific value of the collision, which is calculated by weighting parameters (equation (1))

These variables representing the collision are stored in a data structure. The route planner builds up a list for these in the course of running.

B. The GA problem

Using these data, we use one of the methods of machine learning, this is the GA to determine from the list of collisions which agents should be contacted. For this we need to prioritize the collisions. This is stored in the *estimation* field, and calculated as:

$$estimation = distance\_covered * A + waiting * B + \\ + shut\_down * C + on\_route * D + \\ + manh\_distance * E \quad (1)$$

The numbers A, B, C, D, E are the so-called weights to which we give initial values, and their future values are determined by the GA. With their help we are able to estimate, which agents are willing to give us priority with high possibility in case of a collision.

The *list of collisions* contains data about all collision of an agent. An element of this list consists of two numbers: the serial number of the agent that is collided and the *estimation* field of that agent. From this list, we select N agents with the highest "estimation" values (this is called the *agent list*) and we give back the list of these to the agent to which the list belongs to. The value of N is typically about 5: smaller value may cause that we do not succeed, i.e., we don't get priority from any agent, in case of bigger value we may spend too much time on communication. The order of the agent list has the most important role: if we get priority from an agent of the list, it has effect on the complete route search. In this case the total collision list voids and it is not worth to begin discussions with other agents.

GA aims at minimizing the number of communications among cooperating agents and yet to achieve the best possible result, i.e., to determine optimal routes for the agents. This is served by the estimation of the collisions, the amendment was done by the weights $(A-E)$. A GA will be responsible for the determination of these weights. Following a route plan the corresponding values are calculated and are added to the agent to store, and these values are used to calculate the estimation of the collisions during the next route planning.

In two cases, the GA can be left out:

1) if we got priority from the agent that is the first in the agent list, or
2) if we didn't get priority by any of the agents meaning that the order is irrelevant.

C. GA parameters

We carried out several tests of the algorithm that we can determine the main parameters of GA, which affect to the results. We performed tests with different map size and different numbers of agents in our program simulation environment. Some important parameters of the used GA are

- maximum number of populations: 100. This is necessary to ensure that we find the optimal solution.
- mutation rate: 0.07
- size of population: 20 entities
- selection: roulette wheel selection
- recombination: we used one point crossover for generating the first third of the population, two points crossover for the second third of the population and uniform crossover for the third third of the population
- coding: binary, 5 bits per weight. So the values of the weights are placed in the $[-15, 16]$ interval.

D. Fitness function

The fitness calculation is based on noting, which was the *first element of the agent list that has succeeded* in the previous route planning. Based on this, the elements of the agent list are classified as:

 ISBN: 978-1-61208-331-5

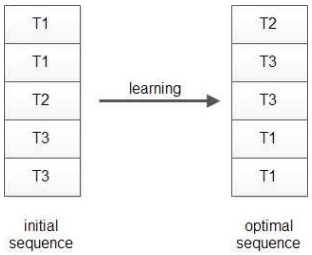

Figure 5: A possible instance of the original and optimal sequence

- priority was requested but we didn't receive (T_1 instance)
- priority was requested and we received (T_2 instance)
- priority wasn't requested because we already got it at a former element of the list (T_3 instance)

The goal of the GA is that the *estimation* field value of the T_2 instance becomes the highest, so next time the algorithm uses it earlier, this way we can save time.

Therefore, the fitness function is determined based on how much the agent list counted again by the new weights $(A, \ldots, E)$, which were stored by some individuals of the GA, approximating the optimal sequence. Optimal sequence is considered to be the one where the T_2 instance is in the first place, under it the T_3 instances (about these we do not know if the outcome had been right or wrong), and at the bottom are the T_1 instances (because these are proven wrong), as Figure 5 shows.

To be able to calculate the fitness of the new agent list - given by the GA - the agents of the existing agent list and their types ($T_1 - T_3$) should be noted. We retrieve the agents of the new list from this list, after that we execute the following algorithm (where type is type of the given agent, serial number is its position in the list).

$result := 0$
if $type = T_1$ **then**
  $result := result + serial\_number * 2$
end if
if $type = T_2$ **then**
  $result := result + (5 - serial\_number) * 5$
end if
if $type = T_3$ **then**
  $result := result + (5 - serial\_number) * 1$
end if
return result

Thus, better results are obtained when

- disadvantageous cases (T_1 type) are placed higher in the list,
- advantageous cases (T_2 type) are placed at the top of the list,
- neutral cases (T_3 type), preferably should be in the lower positions of the list. (Fig. 5)

The multipliers of positions (2, 5, 1) give the weights of importance.

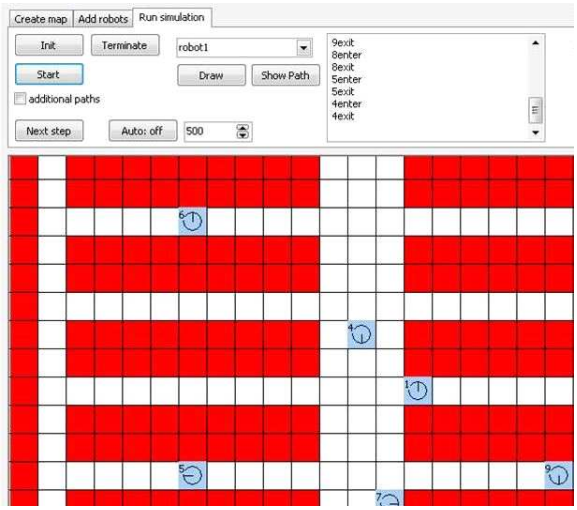

Figure 6: Screen shot part from the simulation environment

TABLE I: THE EFFECT OF THE AGENT NUMBER ON THE RUNNING TIME (BASIC ALGORITHM)

Number of agents	Time of route planning (ms)	Time of planning/ agent(ms)
5	14,4	3,08
10	28	2,8
15	46,8	3,12
20	62,6	3,13

IV. CASE STUDIES

Simple case studies were used to test the efficiency of the proposed learning by GA. It is important to emphasize that the proposed learning by GA method is used for *improving* the result (usually not an optimal solution).

The simulation environment has been developed in Delphi programming language (see Fig. 6 for a screen shot part), by which we could test the agents' movements, and we could compare the operation with and without learning by GA.

A. Efficiency test

In order to test and compare the route planning algorithm we recorded the full running time of the algorithms and examined how this value changed with the increasing complexity of the planning problem.

1) *Effect of the number of agents using the basic route planning algorithm:* In case of the first test the agents were arranged randomly in a $25x25$ cell map-file. Five program runs were performed with each agent number value, and the running times were averaged. Table I shows the simulation results without learning, i.e., by using the basic algorithm [22].
2) *Effect of the number of agents using the advanced algorithm with GA:* In order to test the effect of learning, the agents were arranged randomly in a $25x25$ cell map-file, too. Five program runs were performed with each agent number value, and the

 ISBN: 978-1-61208-331-5

TABLE II: THE EFFECT OF THE AGENT NUMBER ON THE RUNNING TIME (LEARNING BY GA)

Number of agents	Time of route planning (ms)	Time of planning/ agent(ms)
5	13,9	2,78
10	26,6	2,66
15	44,25	2,95
20	59,6	2,98

running times were averaged. Table II shows the simulation results.

It can be seen from the results that the system integrates the new agents well, the agent per-planning time is below 3 ms independently of the number of agents. This important result shows that the GA scales up well with the size and complexity of the problem, thus offering an efficient service of the accumulator-bank.

V. CONCLUSION AND FUTURE WORK

A novel GA-based learning method is proposed in this paper for optimal cooperative route planning of autonomous agents moving in an accumulator-bank. The agents calculate their best possible route in a distributed way giving precedence to other agents to avoid conflict situations, and communicate with each other and the central unit. The agents are equipped with a learning ability in order to minimize the number of communications with the other agents. The adaptive behaviour lets the agents minimize the time to find the optimal routes during the search.

The effect of learning on the performance of the system has been investigated using simple case studies, and substantial improvement has been observed. At the same time, it was observed that the GA used for the learning scales up well with the size and complexity of the problem.

In the future, we will test our method in different situations and we plan to build a simulation environment, in which the agent-robots' motion as well as the unexpected events can be tested.

ACKNOWLEDGMENT

We acknowledge the financial support of the Hungarian State and the European Union under the TAMOP-4.2.2.A-11/1/KONV-2012-0072. This publication/research has been supported by the European Union and Hungary and co-financed by the European Social Fund through the project TAMOP-4.2.2.C-11/1/KONV-2012-0004 - National Research Center for Development and Market Introduction of Advanced Information and Communication Technologies.

REFERENCES

[1] J. L. Adler and V. J. Blue, "A cooperative multi-agent transportation management and route guidance system", Transportation Research Part C: Emerging Technologies, 2002, pp. 433-454.

[2] E. Alba and B. Dorronsoro, "Solving the vehicle routing problem by using cellular genetic algorithms", Evolutionary Computation in Combinatorial Optimization (EvoCOP), Lecture Notes in Computer Science, Vol. 3004, Springer-Verlag, Berlin, 2004, pp. 11-20.

[3] M. Armand, and J. M. Tarascon, "Building better batteries", Nature, Vol. 451, 7 February 2008, pp. 452-457.

[4] J. Baker, "New technology and possible advances in energy storage", Energy Policy, Vol. 36, 2008, pp. 4368-4373.

[5] J. Berger, M. Sassi and M. Salois, "A hybrid genetic algorithm for the vehicle routing problem with time windows and itinerary constraints", In: Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, USA, 1999, pp. 44-51.

[6] O. Bräysy, "Genetic Algorithms for the Vehicle Routing Problem with Time Windows", Special Issue on Bioinformatics and Genetic Algorithms, 2001, pp. 33-38.

[7] K. Fischer, J. P. Müller and M. Pischel, "Cooperative transportation scheduling: An application domain for DAI", Applied Artificial Intelligence: An International Journal, 1996, pp. 1-34.

[8] D. E. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", Addison Wesley, USA, 1989.

[9] A. Göllei, P. Görbe, and A. Magyar, "Modeling and optimization of electrical vehicle baterries in complex clean energy systems", Journal of Cleaner Production, 34, 2012, pp. 138-145.

[10] I. Hadjipaschalis, A. Poullikkas, and V. Efthimiou, "Overview of current and future energy storage technologies for electric power applications", Renewable and Sustainable Energy Reviews, 13, 2009, pp. 1513-1522.

[11] B. Hagen and S. Ralf, "Implementation of Path Planning using Genetic Algorithms on Mobile Robots", Evolutionary Computation, CEC 2006. IEEE Congress on, 2006, pp. 1831-1836.

[12] W. Ho, G. T. S. Ho, P. Ji and H. C. W Lau, "A hybrid genetic algorithm for the multi-depot vehicle routing problem", Engineering Applications of Artificial Intelligence 21, 2008, pp. 548-557, 2008.

[13] J. Hrncir and M. Rovatsos, "Applying Strategic Multiagent Planning to Real-World Travel Sharing Problems" 7th International Workshop on Agents in Traffic and Transportation, AAMAS, 2012 http://arxiv.org/abs/1301.0216

[14] L. Nguyen-Thinh and N. Pinkwart, "Strategy-based Learning through Communication with Humans", Jezic et.al. (Eds.): KES-AMSTA 2012, LNAI 7327, 2012, pp. 54-64.

[15] H. Nazif and L. S. Lee, "Optimized Crossover Genetic Algorithm for Vehicle Routing Problem with Time Windows", American Journal of Applied Sciences 7, 2010, pp. 95-101.

[16] Y. B. Park, "A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines", International Journal of Productions Economics, 2001, pp. 175-188.

[17] J. Y. Potvin, and S. Bengio, "The Vehicle Routing Problem with Time Windows Part II: Genetic Search", INFORMS Journal on Computing 8, 1996, pp. 165-172.

[18] S. Ronald, and S. Kirkby, "Compound optimization solving transport and routing problems with a multi-chromosome genetic algorithm" In The 1998 IEEE International Conference on Evolutionary Computation, ICEC'98, 1998, pp. 365-370.

[19] M. Saggar, T. D'Silva, N. Kohl, and P. Stone, "Autonomous learning of stable quadruped locomotation", In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) ToboCup2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, 2007, pp. 98-109.

[20] S. Sahli, and R. J. Petch, "A GA based heuristic for the vehicle routing problem with multiple trips", J. Math. Model. Algorithms 6, 2007, pp. 591-613.

[21] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability", Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, 2011, pp. 617-624.

[22] Á. Werner-Stark, T. Dulai, and M. K. Hangos, "Cooperative optimal route planning of accumulator-bank servicing robots", Proceedings of ICSEA 2013: The Eighth International Conference on Software Engineering Advances, 2013, pp. 408-413.

[23] M. Zolfpour-Arokhlo, A. Selamat, and S. Z. M. Hashim, "Route planning model of multi-agent system for a supply chain management". Expert Syst. Appl. 40, 5, 2013, pp. 1505-1518.

 ISBN: 978-1-61208-331-5

Resource Aware Workload Management for Autonomic Database Management Systems

Wendy Powley, Patrick Martin, Natalie Gruska
School of Computing
Queen's University
Kingston, Ontario, Canada
{wendy, martin, gruska}@cs.queensu.ca

Paul Bird, David Kalmuk
IBM
Markham Lab
Markham, Ontario, Canada
{pbird, dckalmuk}@ca.ibm.com

Abstract **-- Workloads running in a multi-purpose database environment often compete for system resources causing resource contention, which leads to poor performance. Autonomic database systems will be required to recognize that the system resources are not being utilized optimally and take action to correct the situation. Workload management techniques can be used to choose an appropriate mix of concurrent work to reduce resource contention. We describe a resource aware scheduling approach that predicts the amount of CPU, I/O and sort heap memory that will be required by a query and schedules each query to run only when doing so is unlikely to overwhelm the resources. We present experimental evidence that indicates that overall system performance can be improved using this technique.**

Keywords- workload management; database management systems; autonomic computing; scheduling

I. INTRODUCTION

Database management systems (DBMSs) are an integral part of virtually every computing system and with modern day demands on such systems to handle diverse data types, mixed workloads, and ever-changing demand, it is more important than ever to ensure that these complex systems are self-managing and self-optimizing. It is no longer feasible to manually reconfigure a system to handle a new workload type or a change in workload intensity. The system itself must recognize changing conditions and adapt accordingly. Maintaining a balance of work in the system is crucial to ensure that all the demands and goals are met.

The characteristics of a database workload determine how the resources are used. Online Analytical Processing (OLAP) workloads, for instance, may access a large quantity of data, perform complex calculations and sort large quantities of data thus taxing the CPU, the I/O subsystem and the sort memory. Transactional workloads, on the other hand, may simply scan a table for a particular result and use very little CPU or sort memory. Two or more workloads with similar characteristics running concurrently on the DBMS can result in workload interference, often due to resource contention.

Workload interference may lead to performance degradation in the DBMS system. Consider a workload that is currently executing 300 transactions per second and is using 98% of the CPU. If another workload begins executing on the system that is also CPU-heavy, the CPU will become overloaded. The work will continue to be processed, but at a slower speed as the CPU must be shared. The performance of the initial workload will degrade, perhaps violating goals that have been defined for this workload. If the competing workload was sort intensive and CPU-light, the two workloads may have executed in harmony without detrimental effects to the initial workload.

Workload control is the process whereby the DBMS exerts control over which work is allowed to run in the system. This may be done by admission control (deciding whether or not a query will be admitted to the system based on some criteria), scheduling (deciding the order that the admitted queued queries will be allowed to run) or execution control (termination, suspension, or throttling of currently executing queries) [9]. An autonomic database system incorporates workload control to ensure that the system runs in an optimal state where resources are used effectively and efficiently while allowing all work to meet its service level objectives.

Over the past several years, we have developed a number of workload management techniques [5] [6] [7] [8] and defined a framework that combines the various techniques into a unified system for autonomic workload management [10]. The work described in this paper is a subset of our framework and involves a scheduling approach to workload management. In previous work [4], we proposed a method of scheduling queries based on estimates of the amount of sort heap memory required by each query. The present work extends this work to add additional resources, namely CPU and I/O, and bases the scheduling decisions on the predicted usage of all three resources. The goal of our work is to schedule database queries such that the order of execution ensures that system resources are utilized as fully as possible while not overloading any one resource.

The remainder of the paper is structured as follows. Section II outlines related work. Section III describes the architecture for our prototype scheduling system and outlines the approach. Section IV presents experimental validation of the work. In Section V, we present the conclusions and future directions.

II. RELATED WORK

The current work focuses on scheduling as a form of workload control for database systems. Many algorithms

 ISBN: 978-1-61208-331-5

such as first-come-first-served, shortest job first and priority scheduling are used in operating system job scheduling [16]. We make use of the first fit algorithm in our scheduling of database queries in the current research.

Modeling approaches to predict performance metrics for database queries are gaining in popularity [15]. These performance metrics are necessary for making scheduling decisions. Ahmad et al [14] take this one step further to model the interactions between queries and, using these models, select a mix queries to run concurrently that minimizes contention in the system. This work takes advantages of the unique characteristics of report generation workloads and enforces a fixed multi-programming level (MPL). In contrast, our approach allows the MPL to vary during workload execution and allows for a general workload mix.

Like the work of Ahmad et al, scheduling approaches to control DBMS workloads often control the multi-programming levels; that is, workload control is achieved by controlling the number of queries running concurrently in the system. The work by Schroeder et al. [12] uses queuing theoretic models and a feedback control loop to predict the relationship between throughput, response time and multi-programming levels to optimize the MPL. Although Schroeder et al. evaluate this approach using query priorities in which high priority queries should be chosen to run first, it is also relevant in terms of scheduling for resource control. If the queue is larger, then a query with resource requirements suitable to the currently available resources is more likely to be found. Mehta et al. [13] focus on scheduling business intelligence (BI) batch workloads and attempt to optimize overall response time for the workload. Queries are admitted based on their priority and memory requirements.

Our approach uses models based on information from the optimizer to predict the CPU, I/O and Sort Heap memory required by individual queries. These resources are considered "high impact" resources in a DBMS. We use these predicted measures along with scheduling algorithms to choose which queries will be allowed to run concurrently in the system so as to make efficient use of the system resources and avoid resource overload. Our work is distinctive in that we are considering multiple resources in scheduling decisions.

III. ARCHITECTURE AND APPROACH

Our system can be considered a "load control system", that is, one which controls the current workload executing on the DBMS. The architecture of the load control system is shown in Figure 1. Clients submit queries to the DBMS which are intercepted by the scheduler which consults the DBMS to collect pertinent information regarding potential resource usage. Using this information, a prediction is made by the Resource Requirements Estimator for CPU and I/O usage and the memory requirements for the sort heap. The query is then queued for admission. The Requirements Model contains the policies that rule how the scheduling decisions are made. The Scheduler constantly checks the queue and, if a query can be admitted into the system based on its requirements and the current state of the system, then the query is allowed to proceed. We outline the various components in more detail in the following sections.

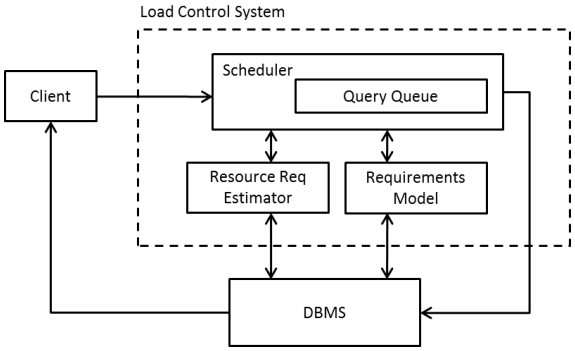

Figure 1. Prototype Architecture

A. *Requirements Estimator*

The requirements estimator predicts the amount of CPU, I/O and sort memory that will be used by a query. The system uses these estimates along with an estimate of the current resource usage to determine whether or not the query can be admitted to the system at a specific point in time.

Estimates are derived from statistics provided by the DB2 Explain tool [1] which generates an access plan for a given query complete with statistics pertaining to the cost of execution of the plan. Relevant statistics for our work include the cumulative CPU cost (measured in the number of instructions required to execute the query), the cumulative I/O cost (the total number of seeks and page transfers executed by the query), the total cost (a weighted sum of I/O and CPU costs for the query expressed in a measurement that IBM calls "*timerons"*) and sort-related costs such as the number of rows to be sorted and the approximate length of each row. Details of how the estimates for each resource are derived are described below.

CPU

The CPU is at maximum capacity when it is nearing 100% utilization. DB2 Explain provides an estimate of the *cumulative* CPU cost (in number of instructions) of executing a particular query plan. The cumulative estimate is the total amount of CPU that will be used during query execution. For our estimates, it is more useful to know the average amount of CPU that will be used over the lifetime of the query. To estimate the *average* CPU cost during the query execution, we divide the cumulative CPU cost by the overall cost of the query provided by DB2 Explain. To estimate the average CPU cost for each individual query, we ran each of our experimental queries (of which there were 17) alone (without competing workload) 100 times while noting both the estimated and the actual CPU cost (average percentage CPU used during query execution). We have found that there is a relatively high correlation ($r = 0.7$, $n = 1700$, $p = 0.05$) between the estimated value and the actual

 ISBN: 978-1-61208-331-5

measured average CPU. We used linear regression to find a formula to predict the percent utilization of CPU for a query given the estimates from DB2 Explain tool. The equation used was:

%CPU = (0.0001 * cumulative_cpu/overall_cost) + 8.39 (1)

I/O

To determine when the I/O subsystem was nearing capacity, we measured the maximum throughput for our I/O channel using a large database table scan with a small buffer pool. The maximum observed throughput was approximately 190MB/s. A reasonable (r = 0.65, n=1700, p = 0.05) correlation exists between the cumulative number of I/Os predicted by the DB2 Explain facility and the average measured throughput of each query run alone. To calculate this correlation, each of the 17 queries was run alone (without competing workload) 100 times while measuring the average measured throughput. Linear regression was used to derive a formula to estimate the average throughput for a query as follows:

Throughput(MB/s) = (0.00004 * cumulative_io) + 19.2 (2)

Sort Heap

The amount of sort heap memory allocated in a DBMS is important to performance because extending the sort heap leads to spills to disk requiring additional I/O and increased response times.

The estimation of the sort heap required by a query, like for CPU and I/O estimation, uses the information contained in the query execution plan. The plan consists of nodes in a tree structure with each non-leaf node representing an operator. Two DB2 operators require sort heap space; sort and hash join. The amount of sort heap required by each of these operators is determined by the query execution plan which provides the number of rows to be sorted as well as the approximate width of each row in bytes. We experimentally determined that there is approximately 75 bytes per row of overhead. Therefore, the estimate for a single sort operator is

RequiredSortMemory = #Rows * (RowWidth + 75) (3)

The DB2 *sortheap* parameter limits the amount of sort heap space that can be assigned to a single sort or hash join. Therefore, the minimum of the value of the sortheap parameter or RequiredSortMemory is used as the estimate for the sort requirements.

Given that not all nodes in a plan are active at the same time, we cannot simply sum the sort requirements for all the nodes in a query [4]. We determine which nodes can be active at the same time by the types of nodes (blocking, not blocking) and the relationships between them (ancestor, descendant). The sort heap estimation process for a complete query plan can be separated into two steps: calculating sort heap sets and using the sort heap sets to calculate sort heap requirements. Both sorts and hash joins are blocking operations. Hence, any node that requires sort heap is a blocking node. This means that when node N becomes active, the sort heap demand is constant for a period of time, until N starts to produce output. Specifically, the amount of sort heap required while N is blocking is the amount that N requires plus that which its active descendants require. This total amount of sort heap is referred to as the sort heap set of N. Conceptually, a sort heap set for node N is calculated by starting at N and traversing towards the leaves of the query execution tree, summing the sort heap requirements of the traversed nodes, until blocking nodes are encountered.

The amount of sort heap required varies throughout its execution. In other work, we evaluate different estimations including the average usage, the dominant usage and the maximum usage [2]. In the current work, we use the average estimate, that is, the average amount of sort heap that a query will use during its execution time.

B. Requirements Model

The Requirements Model represents the current resource status of the system, that is, how much of a particular resource is available in total as well as the amount that is in use by currently running work. A model is used for each of the CPU, the I/O subsystem and the sort memory.

For the CPU, we assume that the maximum amount of CPU utilization is 100 percent. Our goal is to keep the resource busy, but not overload it. Our model states that a query "fits" in terms of CPU if the CPU estimate of the current query plus the total sum of the CPU estimates for all currently executing queries is less than or equal to 90 percent and the actual measured CPU usage is less than 100 percent.

The I/O model is based on our measured maximum throughput which was 190MB/s. To avoid overloading the I/O system, we use 185MB/s as our maximum desired throughput.

Our resource estimator provides us with the worst case I/O estimate; that is, all data that is requested will need to be read from disk. In a DBMS, however, recently requested data will often be found in the bufferpool, a main memory cache managed by the DBMS. The data in the bufferpool may be reused by other queries requesting the same data, thus reducing the amount of necessary I/O. To account for data sharing, we measure and incorporate the buffer pool hit rate (expressed as a value between 0 and 100), which is the measure of how often a page access is satisfied without a physical I/O. A hit rate of 50 (percent) means that a requested page is found in the buffer pool 50% of the time. We then calculate the maximum throughput that will be allowed into the system as:

Total_Throughput = 185 + current_hit_rate * 185/100 (4)

The theory is that if the buffer pool hit rate is high, we can allow more work into the system without overloading the disk. If it is low, it means that more physical I/O is occurring, therefore, less work should be allowed into the system.

 ISBN: 978-1-61208-331-5

Our Requirements Model admits a query based on I/O if its estimated I/O plus the sum of the I/O usage of currently running queries is less than Total_Throughput (as calculated above).

A query is admitted into the system in terms of sort heap requirements if the sort heap requirements estimate for the current query plus the sum of the sort heap requirement estimates for currently running queries does not exceed the value specified by the DB2 parameter, *sheapthres_shr*, which limits the total amount of sort heap used by all running queries.

C. Scheduler

The scheduler makes decisions as to which query to run next based on the rules defined by the Requirements Model. We have considered several different scheduling algorithms in previous work [4]. In the current work, we use the First Fit Scheduling algorithm. Queries are queued in the order in which they arrive for execution. The scheduler traverses the queue (from earliest arrival to most recent arrival) and considers the requirements of each query. In order to fit into the system, all conditions must be met for each of the three resources. That is, the query must fit in terms of predicted CP, I/O and sort heap usage in order to be admitted to the system. The first query found that meets all the requirements is admitted to the system for execution.

IV. EXPERIMENTAL EVALUATION

A. Experimental Environment

Our database workload consisted of 17 OLAP queries based on the TPC-H benchmark [3]. The ordering of the queries was randomly assigned (on a per client basis) prior to the run, but was kept constant throughout all subsequent runs. Our OLAP workload was sort-intensive and the queries varied in their use of CPU and I/O.

In order to ensure that the CPU and the I/O subsystems were heavily utilized at some points, we simulated a CPU-intensive workload by running a simple program that consumed approximately 30 percent of the CPU when run alone. We simulated an I/O intensive workload by performing multiple repeat scans on a table not used by our OLAP workload. We used a very small (and separate) bufferpool for the I/O intensive workload to ensure that the I/O subsystem was being used extensively.

Twelve clients each sent the 17 OLAP queries to the system for processing. The workload was varied every two minutes in the following pattern:

1. OLAP workload alone
2. CPU intensive workload + OLAP workload
3. I/O intensive workload + OLAP workload
4. CPU intensive workload + I/O intensive workload + OLAP workload

Each run was repeated 8 times and average values reported. Between each run, the database system was restarted to clear all monitor elements and a sample workload was run to warm up the bufferpool and to bring the database system to a steady state.

DB2 V9.7 was used to house the 3GB database for the OLAP workload. The bufferpool was configured to 1GB. The parameters sortheap (the maximum sort heap allocated to any single query) and sheapthres_shr (the limit on the total amount of sort heap used by all running queries) were set to 500 and 2500 4K pages respectively. The DBMS was run on a dedicated Windows 8 Server machine configured with 8 GB of RAM and a quad core CPU. The clients and the scheduling system were run on a remote machine.

We compared our proposed scheduling approach to a) a system running with no control where queries were run on a first come, first serve basis and, b) to a system where we fixed the maximum multi-programming level (MPL) to four, that is, the maximum number of queries that were allowed to run concurrently was four. This number was determined experimentally to be an optimal setting for steady performance in our configuration [2]. We expected that the scheduling approach would yield better performance than the system running with no control and that it would perform at least as well as when the optimal multiprogramming level was used. We compare our approach with a limited MPL as setting the MPL is a common approach to reducing the amount of resource contention in a database system.

B. Results

The results are summarized in Tables 1, 2 and 3. Table 1 shows general metrics including the total run time for the 204 queries (12 clients each running 17 queries) in minutes (including wait time), the average wait time per query (minutes), the maximum wait time (minutes), the average execution time (minutes) and the maximum multi-programming level (MPL). Table 2 shows CPU Usage and I/O metrics such as the average disk queue length, the maximum disk queue length, the average throughput in MBs per second, and the buffer pool hit rate (percentage). Table 3 presents the sort metrics including the number of post threshold sort operations, the sort overflows, and the number of hash join overflows and small hash join overflows. Sort and hash join overflow operations are an indication of sort heap contention. Overflows occur when not enough memory can be granted to perform a sort in memory. In this case, temporary results are often written to (and re-read from) disk resulting in increased I/O.

TABLE I. GENERAL METRICS

	Total RunTime (mins)	***Average Wait Time (mins)***	***Max Wait Time (mins)***	***Average Execution Time (mins)***	***Max MPL***
No Control	135	0.07	1.1	7.1	12
MPL 4	133	4.7	12.9	7.8	4
First Fit Schedule	127	4.9	15.7	6.8	8

 ISBN: 978-1-61208-331-5

TABLE II. I/O AND CPU METRICS

	Average Disk Queue Length	*Max Disk Queue Length*	*Buffer Pool Hit Rate (%)*	*Average Through-put (MB)*	*Average CPU Usage (%)*
No Control	10.4	43	83	86	94
MPL 4	9.8	42	84	78	94
First Fit Schedule	5.2	30	86	79	91

TABLE III. SORT METRICS

	Post Threshold Sort Operations	*Sort Overflows*	*Hash Join Overflows*	*Small Hash Join Overflows*
No Control	81	116	62	24
MPL 4	77	112	58	21
First Fit Schedule	21	22	20	7

The results show that the overall execution time was reduced by approximately 6% using the scheduling approach over the baseline (no control) or MPL 4 approaches. Although the average wait time per query was higher for the scheduling approach, the average execution time per query was lower, indicating a more efficient use of resources. The load on the I/O subsystem was reduced as indicated by a reduction in the average (and maximum) disk queue length and a lower average throughput. The average CPU usage decreased slightly. Sort operations were improved with significantly fewer post threshold sort operations, sort overflows, hash join overflows and small hash join overflows performed in the scheduling approach than either the baseline or the MPL 4 cases.

V. CONCLUSIONS AND FUTURE DIRECTIONS

We have presented and validated a scheduling approach to DBMS workload control that we plan to incorporate into our framework for autonomic DBMS workload control. The described approach schedules queries based on their predicted resource usage. Based on our experimentation, the approach yields reasonable results and appears to be promising approach for adapting to workload changes.

The current work will be integrated as the scheduling component of a proposed framework for DBMS workload management [10]. This framework provides coordinated control of different workload management techniques such as admission control, execution control, and scheduling. Each component is controlled by a feedback controller which monitors system performance and adjusts the amount of control exerted by the mechanism accordingly. For example, the execution control component consists of a controller that a) determines the type of execution control to use (throttling or query canceling) and b) sets the degree of control (for example, in the case of throttling, the controller would set the amount of throttling based on feedback regarding the system performance). The controller for the scheduler will measure actual system resource usage and will feed this information back to the system to update the requirements estimators, and to set the threshold policies in the requirements models accordingly. Building the autonomic controller for the scheduler and integrating it into our overall workload control framework will be the next step in our work.

We have presented the results of only the "first fit" scheduling algorithm in this paper. Experiments have been conducted with the smallest job first and the blocking query scheduling algorithms that were used in our previous work [4]. Results using these algorithms are similar to those reported here with the main difference being that the average and maximum wait times are vastly increased for longer queries using a smallest job first algorithm.

Currently a query is only allowed to run if it fits in terms of CPU, I/O and sort memory. There are many other variations of this approach which may prove to be useful. The most promising approach currently under investigation is scheduling by "critical resource". That is, the resources are monitored and if the usage of one or more resources enters a pre-determined "critical state", the scheduling algorithm considers only the critical resource(s) when making scheduling decisions. We plan to base this work on work done by Zeldes and Feitelson [11], who present an algorithm for system resource management that focuses on bottleneck resources and allocates them to the most deserving clients.

REFERENCES

[1] IBM DB2 Universal Database. DB2 V9.5 Information Center. Available: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 [retrieved: Feb 2014].

[2] N. Gruska, Resource-aware Query Scheduling in Database Management Systems. MSC thesis, Queen's University, Kingston, Ontario, July 2011.

[3] Transaction Processing Performance Council, TPC-H Benchmark Specification. Available: http://www.tpc.org/tpch/ [retrieved: Feb 2014].

[4] N. Gruska, W. Powley, P. Martin, P. Bird, and K. McDonald, "Sort-Aware Query Scheduling in Database Management Systems," Proc of 2012 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON 2012), November 2012, pp. 2-10.

[5] P. Martin et al., "The Use of Economic Models to Capture Importance Policy for Autonomic Database Management Systems," Proc. of the 8th Intl. Conf. on Autonomic Computing (ICAC'11) workshops (Autonomic Computing in Economics), June, 2011, pp. 3-10, doi: 10.1145/1998561.1998564

[6] M. Zhang et al., "Utility Function-based Workload Management for DBMSs," Proc of the 7th International Conference on Autonomic and Autonomous Systems (ICAS 2011), May, 2011, pp. 116-121.

 ISBN: 978-1-61208-331-5

[7] W. Powley, P. Martin, M. Zhang, P. Bird, and K. McDonald, Autonomic Workload Execution Control Using Throttling," Proc of the 4th International Workshop on Self-Managing Database Systems (SMDB 2010) in Conjunction with the 26th International Conference on Data Engineering (ICDE 2010), March, 2010, pp. 75-80.

[8] B. Niu, P. Martin, and W. Powley, "Towards Autonomic Workload Management in DBMSs, " Journal of Database Management, 20(3), July - Sept 2009, pp. 1-17.

[9] S. Krompass et al., "Managing Long-Running Queries" In Proc.of EDBT'09, March 2009, pp. 132-143, doi: 10.1145/1516360.1516377.

[10] M. Zhang, P. Martin, W. Powley, P. Bird, and D. Kalmuk, "A Framework for Autonomic Workload Management in DBMSs," Information Technology (special issue on Engineering Adaptive Information Systems), in press.

[11] Y. Zeldes, and D. Feitelson, "On-line Fair Allocations Based on Bottlenecks and Global Priorities," Proc of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13), April 2013, pp. 229-240, doi: 10.1145/2479871.2479904.

[12] B. Schroeder, M. Harchol-Balter, A. Iyengar, E. Nahum, and A. Wierman. "How to Determine a Good Multi-Programming Level for External Scheduling," Proc of the 22nd International Conference on Data Engineering, April 2006, pp. 60-66, doi: 10.1109/ICDE.2006.78

[13] A. Mehta, C. Gupta, and U. Dayal, "BI Batch Manager: A System for Managing Batch Workloads on Enterprise Data-warehouses," Proc of the 11th International Conference on Extending Database Technology, March 2008, pp. 640-651, doi: 10.1145/1353343.1353420

[14] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala, "Interaction-aware Scheduling of Report-Generation Workloads," The VLDB Journal, 20:589-615, 2011, pp 589-615.

[15] A. Ganapathi, et al, "Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning," Proc International Conference on Data Engineering (ICDE), March 2009, pp. 592-603, doi: 10.1109/ICDE.2009.130.

[16] A. Silberschatz, P.B. Galvin, G. Gagne, Operating System Concepts, Wiley, 9th Edition, 2012.

 ISBN: 978-1-61208-331-5

Policy for Distributed Self-Organizing Infrastructure Management in Cloud Datacenters

Daniela Loreti, Anna Ciampolini
Department of Computer Science and Engineering, Universitá di Bologna
Bologna, Italy
Email: {daniela.loreti, anna.ciampolini}@unibo.it

***Abstract*—Modern data centers for cloud computing are facing the challenge of an ever growing complexity due to the increasing number of users and their augmenting resource requests. A lot of efforts are now concentrated on providing the cloud infrastructure with autonomic behavior, so that it can take decisions about virtual machine (VM) management across the datacenter's nodes without human intervention. While the major part of these solutions is intrinsically centralized and suffers of scalability and reliability problems, we investigate the possibility to provide the cloud with a decentralized self-organizing behavior. We present a new migration policy suitable for a distributed environment, where hosts can exchange status information with each other according to a predefined protocol. The goal of the policy is twofold: energy saving and load balancing. We tested the policy performance by means of an ad hoc built simulator. As we expected, our distributed implementation cannot perform as good as a centralized management, but it can contribute to augment the degree of scalability of a cloud infrastructure.**

Keywords-Distributed Infrastructure Management; Cloud Computing; Self-Organization; Autonomic Computing

I. INTRODUCTION

The Cloud Computing paradigm experienced a significant diffusion during last few years thanks to its capability of relieving companies of the burden of managing their IT infrastructures. At the same time, the demand for scalable yet efficient and energy-saving cloud architectures makes the Green Computing area stronger, driven by the pressing need for greater computational power and for restraining economical and environmental expenditures.

The challenge of efficiently managing a collection of physical servers avoiding bottlenecks and power waste, is not completely solved by Cloud Computing paradigm, but only partially moved from customers's IT infrastructure to provider's big data centers. Since cloud resources are often managed and offered to customers through a collection of virtual machines (VMs), a lot of efforts concerning the Cloud Computing paradigm are concentrating on finding the best virtual machine (VM) allocation to obtain efficiency without compromising performances.

Since an idle server is demonstrated to consume around 70% of its peak power [1], packing the VMss into the lowest possible number of servers and switching off the idle ones, can lead to a higher rate of power efficiency, but can also cause performance degradation in customers's experience and Service Level Agreements (SLAs) violations.

On the other hand, allocating VMs in a way that the total cloud load is balanced across different nodes will result in a higher service reliability and less SLAs violations, but forces the cloud provider to maintain all the physical machines switched on and, consequently, causes unbearable power consumption and excessive costs.

In addition, we must take into account that such a system is continuously evolving: demand of application services, computational load and storage may quickly increase or decrease during execution. Due to these contrasting targets, the VM management in a Cloud Computing datacenter is intrinsically very complex and can be hardly solved by a human system administrator. For this reason, it is desirable to provide the infrastructure with the ability to operate and react to dynamic changes without human intervention.

The major part of the efforts in this field relays on centralized solutions, in which a particular server in the cloud infrastructure is in charge of collecting information on the whole set of physical hosts, taking decisions about VMs allocation or migration, and operating to apply these changes on the infrastructure [2], [3]. The advantages of these centralized solutions are well known: a single node with complete knowledge of the infrastructure can take better decisions and apply them through a restricted number of migrations and communications. However, scalability and reliability problems of centralized solutions are known as well. Furthermore, as the number of physical servers and VMs grows, solving the allocation problem and finding the optimal solution can be time expensive, so some other approximation algorithm is typically used to reach a sub-optimal solution in a fair computation time [4].

In this work, we investigate the possibility of bringing allocation and migration decisions to a decentralized level allowing the cloud's physical nodes to exchange information about their current VM allocation and self-organize to reach a common reallocation plan. To this purpose, we designed a novel distributed policy, Mobile Worst Fit (MWF), able to both save power (by switching off the underloaded hosts) and keep the load balanced across the remaining nodes as to prevent SLA violations. The policy adopts a decentralized approach: we imagine the datacenter as partitioned into a collection of overlapping neighborhoods, in each of which the local reallocation strategy is applied. Taking advantage from the overlapping, the VM redistribution plan propagates from a local to a global perspective. We analyze the effects of this approach by comparing it with the centralized application of a best fit policy. In particular, we relay on the definition of the Distributed Autonomic Migration (DAM) protocol [5], used

 ISBN: 978-1-61208-331-5

by cloud's physical hosts to communicate and get a common decision as regards the reallocation of VMs, according to a predefined global goal (e.g., power-saving, load balancing, etc.).

We tested our approach by means of DAM-Sim, a software to simulate the behavior of different policies applied in a traditional centralized way or through DAM protocol on a decentralized infrastructure.

The article is organized as follows: in Section II, we show the architectural structure of our system, giving an overview of the DAM protocol and focusing on the adopted MWF policy; in Section III, we show the experimental results obtained by means of the DAM-Sim simulator; Section IV shows the state-of-the-art of Cloud Computing infrastructure management and Section V illustrates our conclusions and future works.

II. Architectural Framework

We present a distributed solution for Cloud Computing infrastructure management, with a special focus on VM migration.

As shown in Fig. 1, the framework is composed of three main layers:

- the infrastructure layer, specifying a software representation of the cloud's entities (e.g., hosts, VMs, etc);
- the coordination layer, implementing the DAM protocol, which defines how physical hosts can exchange their status and coordinate their work;
- the policy layer, containing the rules that every node must follow to decide where to possibly move VMs.

The separation between coordination and policy layer allow us to use the same interaction model with different policies. We describe each layer in the following sections.

A. Infrastructure Layer

The infrastructure layer defines which information must be collected about each host's status. To this purpose two basic structures are maintained: the *HostDescriptor* and the *VmDescriptor*.

The *HostDescriptor* can be seen as a bin with a certain capacity able to host a number of VMs, each one with a specific request for computational resources. We only take into account the amount of computational power in terms of MIPS offered by each host and requested by a VM. An empty *HostDescriptor* represents an idle server that can therefore be put in a *sleep* mode or switched-off to save power.

The *HostDescriptor* contains not only a collection (the *current map*) of *VmDescriptor*s really allocated on it, but also

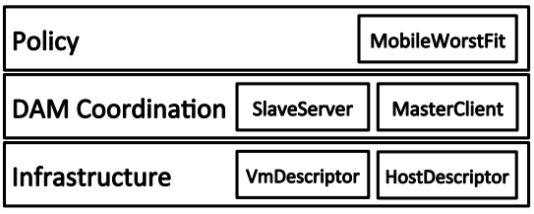

Fig. 1: The three tiers architecture of the Sim-DAM simulator.

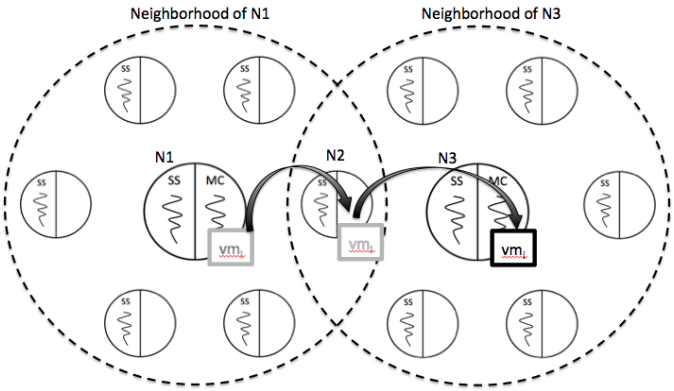

Fig. 2: Schema of two overlapping neighborhoods.

a temporary collection (the *future map*) initialized as a copy of the real one and exchanged between hosts according to the defined protocol. During interactions only the temporary copy is updated and, when the system reaches a common reallocation decision, the *future map* is used to apply the migrations.

In a distributed environment, where each node can be aware only of the state of a local neighborhood of nodes, the number of worthless migrations can be very high. Thus, this double-map mechanism is used to limit the number of migrations (as we describe in Section II-B), by performing them only when all the hosts reach a common distributed decision.

Each VM is also equipped with a migration history keeping track of all the hosts where it was previously allocated. For the sake of simplicity, we assume that a VM cannot change its CPU request during the simulation period.

B. Coordination Layer

The coordination layer implements the DAM protocol which defines the sequence of messages that hosts must exchange in order to get a common migration decision and realize the defined policy.

DAM protocol coordination details are explained in-depth in [5]. The protocol is based on the assumption that the cloud is divided into a predefined fixed collection of overlapping subsets of hosts: we call each subset a *neighborhood*.

We assume that each physical host executes a daemon process called *SlaveServer* (SS in Fig. 2), which owns a copy of the node's status stored into an *HostDescriptor* and can send it to other nodes asking for that.

Each node can monitor its computational load and the amount of resources used by the hosted VMs; according to the chosen policy, it can detect either it is in a critical condition or not. A node can, for example, detect to be overloaded, risking to incur in SLA's violations, or underloaded, causing possibile power waste. If one of these critical conditions happens, the node starts another process, the *MasterClient*, to actually make a protocol interaction begin. We call *rising condition* the one that turns on a node's *MasterClient*.

Since there is a certain rate of overlapping between neighborhoods, the effects of migrations within a neighborhood can cause new rising conditions in adjacent ones.

 ISBN: 978-1-61208-331-5

To better explain the DAM protocol, Fig. 2 shows an example of two overlapping neighborhoods. Each node has a *SlaveServer* (SS in Fig. 2) always running to answer questions from other node's *MasterClient* (MC in Fig. 2) , and optionally can also have a *MasterClient* process started to handle a critical situation. A virtual machine vm allocated to an underloaded node N1 can be moved out of it on N2 and, as a consequence of the execution of the protocol in the adjacent neighborhood of N3, it can be moved again from N2 to N3. It is worth to notice that node N2, as each node of the datacenter, has its own fixed neighborhood, but it starts to interact with it (by means of a *MasterClient*) only if a *rising condition* is observed.

Note that N1's *MasterClient* must have N2 in its neighborhood to interact with it, but the *SlaveServer* of N2 can answer to requests by any *MasterClient* and, if a critical situation is detected (so that N2 *MasterClient* is started) its neighborhood does not necessarily include N1.

As regards this environment, we must remark that the migration policy should be properly implemented in order to prevent never-ending cycles in the migration process.

We must ensure that the neighbors's states the *MasterClient* obtains, are consistent from the beginning to the end of the interaction. For this reason, a two-phase protocol is adopted:

1) DAM Phase 1: The *MasterClient* sends a message to all the *SlaveServers* neighbors to collect their *HostDescriptors*. This message also works as a *lock* message: when the *SlaveServer* receives it, locks his state, so that no interactions with other *MasterClients* can take place. If a *MasterClient* sends a request to a locked *SlaveServer*, simply waits for the *SlaveServer* to be *unlocked* and to send its state.

2) DAM Phase 2: The *MasterClient* compares all the received neighbor's *HostDescriptors* with a previous copy he stored. If the *future map* allocation is changed, performs phase 2A, otherwise increments a counter and, when it exceeds a certain maximum, performs phase 2B:

- *Phase 2A:* the *MasterClient* computes a VM reallocation plan for the whole neighborhood, according to the defined policy, and sends back to each *SlaveServer* neighbor the modified *HostDescriptor*. The state is accepted passively by the slaves, without contradictory. The migration decisions only change the *future map* of VM allocation. No host switch-on/off or VM migration is performed in this phase. After all new states are sent, the *SlaveServers* are *unlocked* and the *MasterClient* begins another round of the protocol interaction by restarting phase 1.
- *Phase 2B:* when the number of round with unchanged neighbor's allocation exceeds a defined maximum, the *MasterClient* sends an *update-current-status* request to all *SlaveServers* and terminates. This last message notifies the *SlaveServers* that information inside the *HostDescriptor* should be applied to the real system state. The *SlaveServer* again executes it passively and unlocks his state.

Phase 2A and 2B alternatives come from the need for reducing the number of migration physically performed. Looking at example in Fig. 2, if hosts only exchange and update the current collection of VMs, every *MasterClient* can only order a real migration at each round, so that vm_i on N1 would be migrated on N2 at first, and later on N3. Using the temporary *future map* (initially copied from the real one) and performing all the reallocations on this abstract copy, real migration are executed only when the N3's *MasterClient* exceeds a maximum number of rounds and vm_i can directly go from N1 to N3.

Input: h, t, FTH_DOWN, FTH_UP.

```
1:  ave = calculateNeighAverage();
2:  MTH_DOWN = ave − t;
3:  MTH_UP = ave + t;
4:  u = h.getLoad();
5:  if u < FTH_DOWN or u < MTH_DOWN then
6:     vmList = h.getFutureVmMap();
7:  else if u > FTH_UP or u > MTH_UP then
8:     vmList = selectVms();
9:  end if
10: if vmList.size ≠ 0 then
11:    migrateAll(vmList);
12: end if
```

Fig. 3: MWF policy algorithm.

C. Policy Layer

The Policy Layer is responsible for the decentralized migration decision process. This paper presents MWF, a novel policy aiming to switch off the underloaded hosts to save power, while maintaining the load of the other nodes balanced. MWF exploits two fixed thresholds (FTH_UP and FTH_DOWN) and two dynamic thresholds (MTH_UP and MTH_DOWN) used to detect rising conditions. The fixed thresholds identify risky situations: if the host is less loaded than FTH_DOWN an energy waste is detected, while, if the host is more loaded than FTH_UP, SLA violations may occur. The dynamic thresholds (MTH_UP and MTH_DOWN) represents the upper and lower values that cannot be exceeded in order to maintain the neighborhood balanced.

According to the DAM coordination protocol, at each iteration the *MasterClient* collects the VM allocation map of the neighbors and executes a MWF optimization as detailed in Fig. 3: the *MasterClient* calculates the average of resource utilization in his neighborhood (*calculateNeighAverage()* in line 1 of Fig. 3) and uses it to compute the two dynamic thresholds (MTH_DOWN and MTH_UP) by adding and subtracting a tolerance interval t (lines 2-3 of Fig.3). Then the *MasterClient* checks its *HostDescriptor* h and collects the current computational load u by invoking a specific $getLoad()$ method on the *HostDescriptor* (line 4 of Fig. 3).

The computational load u of the host is compared to fixed and dynamic thresholds: if it is less than the lower thresholds, the *MasterClient* attempts to put the host in *sleep* mode by migrating all the VMs allocated; otherwise, if the host load exceeds the upper thresholds, only a small number of VMs are selected for migration. As we can see in line 5-6 of Fig.3, if the computational load u is less than the fixed (FTH_DOWN)

 ISBN: 978-1-61208-331-5

Input: h, MTH_UP, FTH_UP. **Output:** vmsToMove.

```
1:  u = h.getLoad();
2:  vmList = h.getFutureVmMap();
3:  vmList.sortDecreasingLoad();
4:  minU = ∞; bestVm = null;
5:  thr = min{FTH_UP, MTH_UP};
6:  vmsToMove = emptyList();
7:  while u > thr do
8:    for each vm in vmList do
9:      var = vm.getLoad() − u + thr;
10:     if var ≥ 0 then
11:       if var < minU then
12:         minU = var;
13:         bestVm = vm;
14:       end if
15:     else
16:       if minU == ∞ then
17:         bestVm = vm;
18:       end if
19:       break;
20:     end if
21:   end for
22:   u = u − bestVm.getLoad();
23:   vmsToMove.add(bestVm);
24:   vmList.remove(bestVm);
25: end while
```

Fig. 4: The selectVms() procedure.

or the dynamic (MTH_DOWN) lower thresholds, all the VMs of the host are collected for migration into an array $vmList$. $h.getFutureVmMap()$ in line 6 is the method to collect the temporary allocation. Indeed in this phase, the policy only works on a copy of the real VM allocation map, because according to DAM protocol, all the migrations will be performed only when the whole datacenter reach a common decision. If the load u is detected to be higher than the fixed (FTH_UP) or dynamic (MTH_UP) upper thresholds, then the $selectVm()$ operation is invoked to pick (from the host h temporary state) only the less loaded VMs whose migration will result in the host load to go back under both MTH_UP and FTH_UP. $selectVm()$ is a modified version of Minimization of Migrations algorithm from Beloglazov et al. [6] and is detailed in Fig. 4. Differently from [6], we select the threshold thr as the minimum between FTH_UP and MTH_UP.

The list of chosen VMs $vmList$ is finally migrated to neighbors by means of a modified worst-fit policy ($migrateAll(vmList)$ in line 11 of 3). As shown in Fig. 5, the $migrateAll$ procedure takes as input the list of vm to move ($vmList$), the host h where they are currently allocated, the list $offNeighList$ of switched-off hosts in h's neighborhood, the $underNeighList$ of h's neighbors with load level lower than FTH_DOWN, and $otherNeighList$ of all the other neighbors of h. The procedure considers the VMs by decreasing CPU request and, according to the principles of worst-fit algorithm, tries to migrate it to the neighbor n with the highest value of free capacity (lines 2-13 of Fig. 5). If no neighbor in $otherNeighList$ can receive the vm, the $underNeighList$ is considered with a best-fit

Input: vmList, h, offNeighList, underNeighList, otherNeighList.

```
1:  vmList.sortDecreasingLoad();
2:  for each vm in vmList do
3:    vmU = vm.getLoad();
4:    maxAvail = 0; bestHost = null;
5:    for each n in otherNeighList do
6:      if n ∉ vm.getMigrationHistory() then
7:        avail = FTH_UP − n.getLoad() + vmU;
8:        if avail > maxAvail then
9:          maxAvail = avail;
10:         bestHost = n;
11:       end if
12:     end if
13:   end for
14:   if bestHost == null then
15:     minU = ∞
16:     for each n in underNeighList do
17:       if n ∉ vm.getMigrationHistory() then
18:         avail = FTH_UP − n.getLoad() + vmU;
19:         if avail >= 0 and avail < minU then
20:           minU = avail;
21:           bestHost = n;
22:         end if
23:       end if
24:     end for
25:   end if
26:   if bestHost == null and !empty(offNeighList)
      and u > FTH_UP then
27:     bestHost = offNeighList.get(0);
28:   else
29:     migrationMap = null; {all-or-none behavior}
30:     break;
31:   end if
32:   migrationMap.add(vm, bestHost);
33: end for
34: commitOnFutureMap(migrationMap);
```

Fig. 5: The migrateAll() procedure.

approach (lines 14-25 of Fig. 5), thus allocating vm on the most loaded host of the list. This ensure that neighbors with CPU utilization near to FTH_DOWN are preferred, while less loaded ones remain unchanged and will be hopefully switched-off by other protocol's interactions. Finally, if neither hosts in $underNeighList$ can receive vm (e.g, because the list is empty), but h is more loaded than FTH_UP, then h is in a risky situation because SLA's violations can occur. Thus a switched-off neighbor is woken up (line 27 of Fig. 5). $migrateAll(vmList)$ operates in a "all-or-none" way, such that the migrations are committed on the future maps (line 34 of Fig. 5) only if it is possible to reallocate all the VMs in the list (i.e., without making other hosts to exceed FTH_UP), otherwise no action is performed (line 29 of Fig. 5).

As shown in Fig. 6, suppose that a protocol execution by the *MasterClient* of h_b decides to migrate a virtual machine vm_i currently allocated on h_c to h_b. When the *SlaveServer* of h_b is unlocked, the policy execution on h_a's *MasterClient* can decide to put vm_i into h_a. Now if h_c has a *MasterClient*

 ISBN: 978-1-61208-331-5

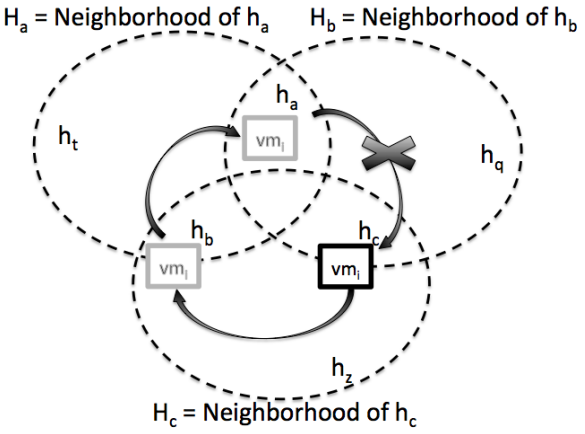

Fig. 6: Example of three overlapping neighborhoods.

running, and decides to migrate vm_i back to h_c, then h_c can take the same decision as before and a loop in vm_i migration starts. If this happens, the distributed system will never converge to a common decision. In order to face this problem, the MWF policy exploits the migration history inside each *VmDescriptor* to avoid loops in reallocation: a VM can be migrated only on a host that it never visited before. Once the distributed autonomic infrastructure reached a common decision, the migration history of each VM is deleted.

III. Experimental results

To understand the effectiveness of the proposed model we developed DAM-Sim [5]: a Java simulator able to apply a specific policy on a collection of neighborhoods through DAM protocol and compare the performance with a centralized policy implementation.

We tested our approach on a set of 100 physical nodes hosting around 3000 VMs (i.e., an average value of 30 VMs on each host), repeating every experiment with an increasing average load on each physical server. We fixed the FTH_DOWN threshold at 25% of computational load and the FTH_UP at 95%, while the tolerance interval t for load balancing is fixed at 8%. We always start from the worst situation for power-saving purposes, i.e., all the servers are switched on and have the same computational load within the fixed thresholds. To make the DAM protocol start we need some lack of balance in the datacenter, so we forced 20 hosts to be more loaded and 20 hosts to be less loaded than the datacenter average value.

In Fig. 7, we compare the MWF performance with nN=5 and 10 nodes in each neighborhood, with the application of a centralized best fit policy (GLO in Fig. 7a). We also show the performance of a best fit policy applied in a distributed way by means of DAM protocol (BF in Fig. 7d). Details about BF implementation can be found in [5].

Fig. 7a and 7d show the number of servers switched on at the end of the MWF and BF executions. As we expected, the DAM protocol cannot perform better than a global algorithm. Indeed, the global best fit policy can always switch off a higher rate of servers resulting in the lower trend. Furthermore, as regards the power saving objective, we can see that BF perform better than MWF for all the selected neighborhood dimensions. This comes from the different objectives of the two policies: MWF tries to switch-off the initially underloaded servers to save power, while keeping the load of the working servers balanced; BF brings into question all the neighborhood allocation at each *MasterClient* interaction, considering only power-saving objectives.

Fig. 7b and 7e show the number of migrations executed. Since the number of VMs can vary a bit from a scenario to another and the number of switched off servers influences the result, in the graph we show the following rate:

$$nMig\frac{onServers}{nVM} \tag{1}$$

where $nMig$ is the number of migrations performed, $onServers$ is the number of working servers at the end of the simulation and nVM is the number of VMs in the initial scenario.

Since no information about the current allocation of a VM is taken into account during the policy computation in a global environment, the number of migrations can be very high. Indeed is high the resulting trend of migration for the global policy, while DAM always outperforms it. In particular, MWF performs better than BF for every selected neighborhood dimension. Nevertheless, for high value of computational load the performance of MWF in terms of number of switched off server are comparable to those of the global best fit policy, while the migration rate is significantly lower.

Fig. 7c and 7f show the number of messages exchanged between hosts during the computation. As we expected, it significantly increases as the number of servers in each neighborhood grows. Even if the number of messages for low values of neighborhood dimension is comparable to the one of the global solution, when it grows, the number of messages exchanged significantly increases.

At the moment, the simulator is not able to give trustworthy results about execution time for distributed environments, because the CPU executing the simulator code can only sequentialize intrinsically concurrent processes of the protocol. For this reason, no test about execution time is reported.

In Fig. 8, we can see the distribution of number of servers along load intervals. In the initial scenario (INITIAL in Fig. 8) all the servers have 50% load except for 20 underloaded and 20 overloaded nodes. We show the distribution after a global best fit optimization (GLO in Fig. 8) and the application of MWF and BF by means of DAM protocol with 5 as neighborhood dimension.

The application of a global best fit switches-off a large number of servers to save power, but packs too much VMs on the remaining hosts. This results in the red distribution in Fig. 8, where almost all the switched on servers are loaded at 95s% creating an high risk of SLAs violations. The best fit (BF) algorithm applied by means of DAM protocol suffers of the same problem: a large number of servers is switched-off, but a part is forced to have 95% load. MWF is more

 ISBN: 978-1-61208-331-5

effective from the load balancing perspective: it can switch-off less servers than BF, but is able to decrease the load of the overloaded nodes leaving all the working servers balanced.

As we expected, Fig. 8 reveals that the median of the MWF distribution is augmented respect to the initial configuration. This is due to the fact that a certain number of servers is switched-off, thus the global load of the remaining servers results increased.

IV. Related works

Our work mainly concern low level infrastructural support, in which the management of virtualized resources is always a compromise between system performance and energy-saving. Indeed, in a cloud infrastructure there are usually well-defined SLAs to be compliant to and perhaps the simplest solution is to use all the machines in the cloud. Nevertheless, if all the hosts of the datacenter are switched on, the energy waste increases leading to probably too high costs for the cloud provider.

Around cloud environments, with their contrasting targets of energy-saving versus performance and SLAs compliance, a lot of work was done in order to provide some kind of autonomy from human system administration and reduce complexity. Some of these works involves automatic control theory realizing an intrinsic centralized environment, in which the rate of utilization of each host is sent to a collector node able to determine which physical machines must be switched off or turned on [2], [3], [7]. Some other solutions concern centralized energy-aware optimization algorithms [4], [8], [9], in particular extensions of the Bin Packing Problem [10], [11] to solve both VMs allocation and migration problems [6]. These approaches focus on finding the best solution and minimizing the complexity of the algorithm, without concerning the particular implementation, but assuming a solver aware of the whole system state (in terms of load on each physical host and VM allocation). Thus they particularly lend to a centralized implementation.

Finally, other approaches involve intelligent, optionally bio-inspired [12], [13], agent-based system, which can give to the datacenter a certain rate of independence from human administration, showing an intelligent self-organizing emergent behavior [14], [15], and also provide the benefits of a more distributed system structure [16]. As in [14] which is based on Gossip protocol [17], we adopt a self-organizing approach, where coordination of nodes in small overlapping neighborhoods leads to a global reallocation of VMs, but differently from [14] we created a more elaborate model of communication between physical hosts of the datacenter. In particular, while in [14] each migration decision is taken after a peer-to-peer interaction comparing the states of the only two hosts involved, in our approach the migration decisions are more accurate because they comes from an evaluation of the whole neighborhood state.

V. Conclusions

We presented a VM migration policy suitable for a distributed management in a cloud datacenter. To do so, we relayed on a decentralized solution for cloud virtual infrastructure management (DAM), in which the hosts of the datacenter are able to self-organize and reach a global VM reallocation plan, according to a given policy.

We tested the policy behavior by means of a software simulator. MWF shows good performances for various computational loads in terms of both number of migrations requested and number of switched-off servers. MWF is also able to achieve an appreciable load balancing among the working servers, while still some work remain to do to decrease the number of messages exchanged. Therefore in the near future, we plan to optimize the DAM protocol in order to reduce the amount of messages in each interaction. As we expected, the distributed MWF policy cannot outperform a centralized global best-fit policy (especially in terms of number of switched-off hosts and exchanged messages), but the decentralized nature of our approach can intrinsically contribute to augment the scalability of the cloud management infrastructure.

In the near future, we will use DAM-Sim to test different and more elaborate reallocation policies, taking into account not only computational resources, but also memory and bandwidth requirements. We will introduce variations of VM load requests at simulation time to better mirror real datacenter environments. Furthermore, in this work, we avoid loops in VM migrations by preventing the allocation on nodes that already hosted the same vm before. We plan to relax this restrictive constraint by means of a Most Recently Used queue of hosts.

Finally, we plan to test our implementation on a real cloud infrastructure and compare the time to get a common distributed decision with the centralized implementation of the same reallocation policy. Furthermore, on a real cloud infrastructure we expect to face low level architectural constraints in overlapping neighborhoods definition, which will request deeper investigations.

References

[1] X. Fan, W.-D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in The 34th ACM International Symposium on Computer Architecture. ACM New York, 2007, pp. 13–23.

[2] Jung, “Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures,” in International Conference on Distributed Computing Systems, IEEE, Ed., June 2010, pp. 62–73.

[3] H. C. Lim, S. Babu, and J. S. Chase, “Automated control in cloud computing challeges and opportunities,” in ACDC ’09, Proceedings of the 1st workshop on Automated control for datacenters and clouds. ACM New York, 2009, pp. 13–18.

[4] A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers,” Concurrency and Computation: Practice and Experience, vol. 24, no. 13, pp. 1397–1420, September 2012.

[5] D. Loreti and A. Ciampolini, “Green-dam: a power-aware self-organizing approach for cloud infrastructure management,” Università di Bologna, Tech. Rep., 2013 - http://www.lia.deis.unibo.it/Staff/DanielaLoreti/HomePage_files/Green-DAM.pdf.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, May 2012.

 ISBN: 978-1-61208-331-5

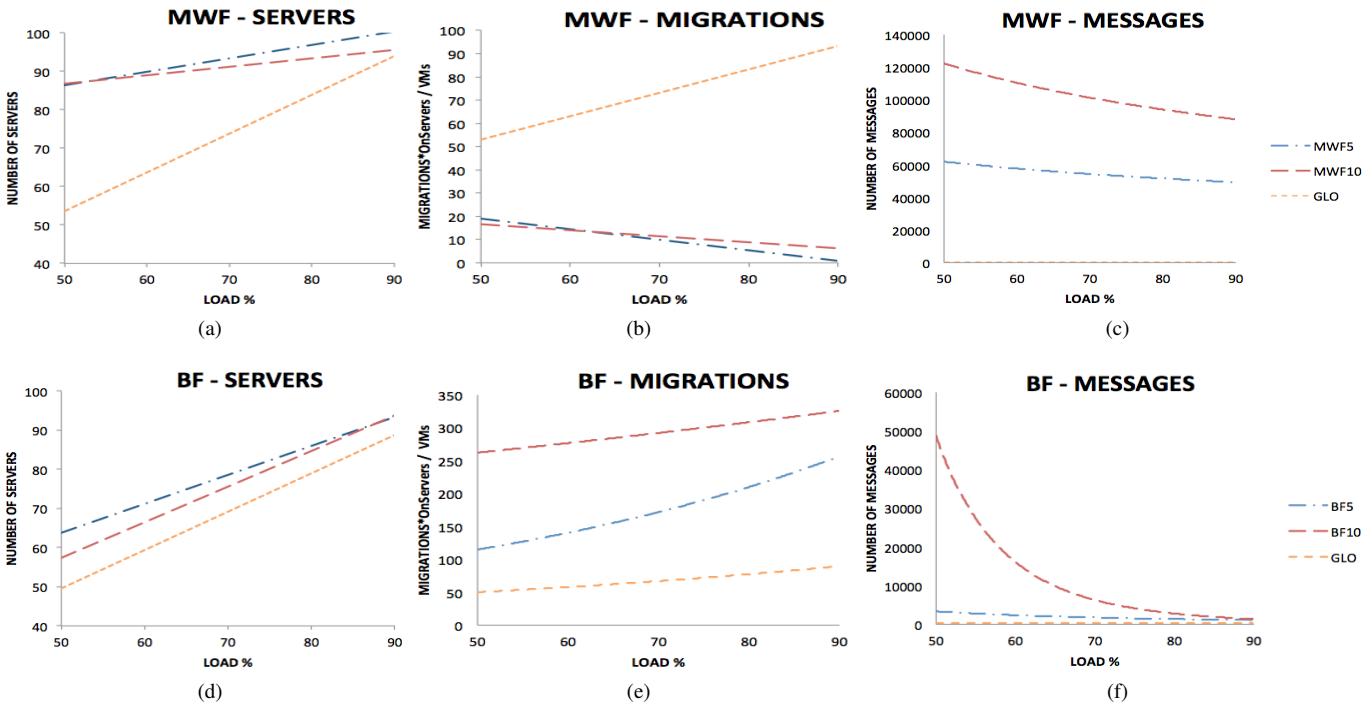

Fig. 7: MWF end BF performance comparison.

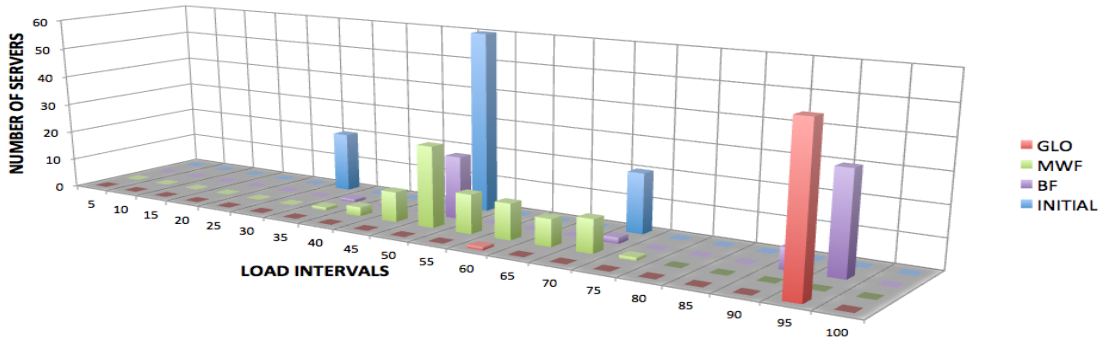

Fig. 8: Distribution of servers on load intervals.

[7] E. Kalyvianaki, "Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters," in ICAC '09 Proceedings of the 6th international conference on Autonomic computing, ACM, Ed., 2009, pp. 117–126.

[8] R. Jansen, "Energy efficient virtual machine allocation in the cloud," in Green Computing Conference and Workshops (IGCC), 2011 International. IEEE, July 2011, pp. 1–8.

[9] A. J. Younge, "Efficient resource management for cloud computing environments," in Green Computing Conference, 2010 International. IEEE, August 2010, pp. 357–364.

[10] J. Levine and F. Ducatelle, "Ant colony optimisation and local search for bin packing and cutting stock problems," Journal of the Operational Research Society, pp. 1–16, 2003.

[11] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," in 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, Ed., December 2010, pp. 127–134.

[12] R. Giordanelli, C. Mastroianni, and M. Meo, "Bio-inspired p2p systems: The case of multidimensional overlay," ACM Transactions on Autonomous and Adaptive Systems (TAAS), vol. 7, no. 4, p. Article No. 35, December 2012.

[13] S. Balasubramaniam, K. Barrett, W. Donnelly, and S. V. D. Meer, "Bio-inspired policy based management (biopbm) for autonomic communications systems," in 7th IEEE International workshop on Policies for Distributed Systems and Networks, IEEE, Ed., June 2006, pp. 3–12.

[14] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," Technical Report UBLCS-2011-01, 2011.

[15] A. Vichos, "Agent-based management of virtual machines for cloud infrastructure," Ph.D. dissertation, School of Informatics, University of Edinburgh, 2011.

[16] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "A distributed approach to dynamic vm management," in Network and Service Management (CNSM), 2013 9th International Conference on, October 2013, pp. 166–170.

[17] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dinamic networks," ACM Transaction on Computer Systems, vol. 23, no. 3, pp. 219–252, August 2005.

 ISBN: 978-1-61208-331-5

Experiments with NetLogo for Distributed Channel Assignment in Dense WLAN Networks

Vangelis Gazis, Konstantinos Sasloglou, Andreas Merentitis, Kostas Mathioudakis
AGT Group (R&D) GmbH
Darmstadt, Germany
e-mail: {vgazis,ksasloglou,amerentitis,kmathioudakis}@agtinternational.com

***Abstract*—Dense IEEE 802.11 networks require thorough radio network planning to minimize the detrimental effects of channel interference and achieve good performance. When multiple IEEE 802.11 networks operating under different administrations reside in the same geographical area (as defined by the bounds of radio signal transmission and reception), the problem of efficient radio resource allocation arises. Particularly, in highly populated urban environments, efficient radio resource allocation can be quite a challenge, due to the finite number of interference-free channels available in the IEEE 802.11 family of standards. To address this shortcoming, we have investigated how the radio resource management capabilities provided by the IEEE 802.11k standard can support a distributed approach in channel assignment. In this paper, we present our family of distributed channel assignment algorithms and elaborate on their pros and cons. Furthermore, we present our simulation environment for simulating distributed channel assignment scenarios based on the NetLogo agent modeling environment.**

***Index Terms*—Channel assignment; Channel Allocation; WLAN; Distributed Systems; Simulation; NetLogo**

I. INTRODUCTION

The establishment of wireless communication as a technological commodity in modern society and the insatiable demand for Internet connectivity paved the way to the tremendous popularity and commercial success of the IEEE 802.11 family of standards. With IEEE 802.11b/g as their flagship and—currently—most widespread standard, IEEE 802.11 Wireless LAN (WLAN) type systems feature an ever increasing footprint worldwide. Standardisation work is feverish and the IEEE 802.11 working groups regularly publishes improvements to existing standards and introduces additional features to address emerging use cases.

To address the complexity associated with the management of large scale deployments in the Future Internet (FI), the vision of self-managing systems has been proposed and adopted. Under the umbrella self-management concept, self-awareness, self-configuration, self-protection, self-healing and self-optimization stand as prominent properties. The fundamental premise of the self-management paradigm is to constrain the operational expenses resulting by the current network management practices in the Internet. To this end, three thematic areas have converged [1] under the umbrella of the Future Internet architecture: autonomic computing, cognitive networking and self-organization.

A. Radio interference

In a wireless infrastructure, network management and, in particular radio network planning, becomes a challenging task due to the volatile and unpredictable nature of the wireless medium and the mobility patterns of terminal devices. The increasing footprint and diversification of IEEE 802.11 type components in ICT equipment suggests that a large portion of FI devices will employ one or more types of wireless access technology. Due to the lack of coordination that characterizes applications in the mass consumer market segments, static and centralized solutions to wireless network planning prove inefficient in this chaotic environment [2].

Minimizing radio interference is essential to realizing good system performance in IEEE 802.11 networks due to the finite number of interference-free channels available [3]. However, the volatile nature of the wireless medium, combined with varying patterns of traffic demand, render any radio network planning exercise a challenging task. The latter is in iteself a sophisticated task, which, besides expert knowledge, may also require a solid understanding of the propagation profile at each radio site [4]. In cases where IEEE 802.11 networks operating under different administrations reside in the same geographical area (as defined by the bounds of radio signal transmission and reception, efficient radio resource allocation becomes quickly problematic. This is due to available technology solutions for radio resource management of IEEE 802.11 installations supporting only a centralized model of administration (e.g., Cisco Unified Wireless Access [5], HP MultiService Mobility Controller [6]). In addition, they lack support for peer-to-peer communication between IEEE 802.11 access points in regard to radio resource management procedures. Thereupon, as IEEE 802.11 systems continue to increase their footprint in the residential and enterprise market segments, efficient radio resource allocation in a dense urban environment rapidly devolves to a chaotic situation [2].

B. Channel assignment

Channel assignment in IEEE 802.11 systems may result in a conflict where more than one adjacent (in terms of radio coverage) wireless access points use the same channel, thus causing a substantial drop in performance. In addition, as Fig. 2 illustrates for the case of IEEE 802.11b/g/n, adjacent access points may use different channels but still experience a substantial spectrum overlap [3], thus still suffering from

 ISBN: 978-1-61208-331-5

interference and degraded performance. Performance can also be degraded by the so-called hidden terminal problem whereby the transmission of interfering access points are beyond each other's reception range but within the reception range of client devices, thus limiting throughput (e.g., client device D_1 and access points AP_1 and AP_2 in Fig. 1).

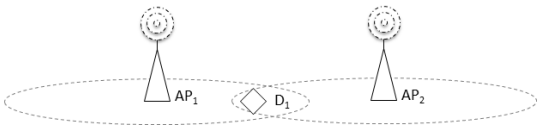

Fig. 1. The hidden terminal case in an IEEE 802.11 access point network.

The main set of IEEE 802.11 standards defines the basic capacities of the Medium Access Control (MAC) layer that enable wireless data exchange and mechanisms [3]. Radio resource management concerns (e.g., measurements of the radio channel) are addressed by the IEEE 802.11k standard [7]. The latter defines the exchange of radio resource measurement information to facilitate network management tasks. To this end, it provides measurements related to the physical and the (wireless) link layer of the protocol stack [7]. In regard to channel assignment, IEEE 802.11k defines channel load measurements as well as noise and time histograms.

C. Research motivation

By making detailed measurements of the radio environment avaialble, several key optimizations in terms of performance become feasible. Particularly deployments in harsh radio environments (e.g., aeroplanes, factories, municipalities, etc.) are greatly facilitated. Utilizing such measurements in a distributed manner whereby each access point operates autonomously in deciding which channel to assign presents a particularly promising proposal. The challenges entailed in distributed approaches include the engineering of convergence and stability properties to the distributed algorithm that is collectively effected by the access points. In prior work we studied the convergence properties between different distributed algorithms for channel assignment in dense IEEE 802.11k systems. In continuation of that work, we explore the parameter space of these algorithms using the NetLogo environment for multi-agent simulation.

We maintain focus on dense deployments of independently managed IEEE 802.11k systems in an urban environment where central coordination of radio resource management tasks is not a realistic assumption. Independent administrations may be insufficiently skilled, unable or even unwilling to coordinate their radio resource management strategies, assuming they have any. Hence, dynamic distributed resource management strategies without operator involvement are the most realistic approach to optimize the use of common radio resources in this chaotic environment. In developing our discussion further, we focus exclusively on IEEE 802.11 systems operating in infrastructure mode.

The rest of the paper is organized as follows: Section II presents the network model of our studies and outlines related work. Section III introduces the NetLogo agent modeling enrivonment and presents its salient features. Section IV presents our approach of modeling in NetLogo dense IEEE 802.11 networks operating under an autonomic setting. Section V presents the family of distributed channel assignment algorithms we have integrated in our NetLogo model. Section VI presents the experimentation settings and shares early results. Finally, Section VII concludes the paper and sets directions for future work.

II. System model

A. Related work

In [8], the authors take into account the traffic load at the MAC layer and propose a heuristic method to minimize the maximum effective channel utilization at the access point with the highest load. In [9] each access point monitors the number of active stations and tries to minimize the maximum effective channel utilization. We note that the set of active stations includes stations associated with a particular access point as well as stations using the same channel and whose transmission power levels is above the access point's receiver sensitivity threshold. The work in [10] proposes a new channel assignment mechanism for infrastructure-based IEEE 802.11 wireless networks in decentralized scenarios. The proposed mechanism operates at the access point level to select the operating channel automatically based on client station measurements exchanged through the IEEE 802.11k standard [3]. In [11], the authors evaluate the throughput performance of different channel assignment strategies in an experimental trial involving IEEE 802.11b systems while [12] proposes a set of algorithms to simultaneously solve the channel selection and user association problems in a fully distributed manner.

Another approach [13], [14] introduces a fully decentralised stochastic algorithm for graph colouring and applies it to the isomorphic channel assignment problem in IEEE 802.11 WLAN systems. Their algorithm does not require communication among WLAN access points and employs learning to ultimately converge to a conflict-free channel assignment. Given the widespread deployment of WLAN technology in urban areas and the lack of coordination among WLAN administrations with regard to channel assignment, we find this approach particularly interesting. Hence, we adopt it for our studies and extend it to accommodate the existence of communication among WLAN access points for purposes of radio resource management tasks. An obvious question is whether the performance of the learning algorithm can be further improved by allowing for the exchange of radio resource information among adjacent WLAN access points and what kind of information can support such an improvement.

B. Working assumptions

The IEEE 802.11 standards define a finite set of wireless channel resources and the respective access and usage controlling mechanisms [3]. For instance, IEEE 802.11b/g specifies the details of 14 distinct channels, with minor differences between world regions due to regulatory constraints. As Fig.

 ISBN: 978-1-61208-331-5

2 depicts, only channels 1, 6 and 11 are interference-free to each other and some amount of spectrum overlap occurs in the remaining subset of the IEEE 802.11 channels.

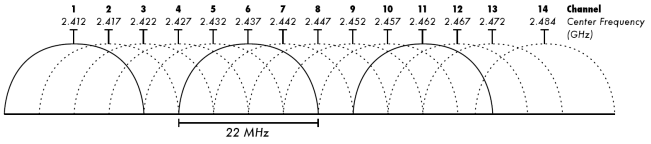

Fig. 2. IEEE 802.11b/g channel overlap (2.4 GHz band) [3].

The extent to which interference attributed to partially overlapping channels in IEEE 802.11 type systems is a determinant of system performance remains a matter of debate [15]–[17]. Analytical and simulation studies [15] suggest that the assignment of partially overlapping channels to neighbouring wireless access points can be justified, depending on the relative utilization of each channel. Increasing path loss decreases the level of interference, therefore, the actual physical distance among wireless stations is an important performance factor [17]. When multiple types of IEEE 802.11 systems coexist in the same geographical area, their topological distribution will largely affect overall performance, with proximal transceivers experiencing dramatic degradation in their channel throughput [16]. Nonetheless, it remains that a finite set of non-overlapping channels is available to wireless devices, a fact aligned to our design assumptions.

C. Channel assignment as graph colouring

Channel assignment in IEEE 802.11 type systems is equivalent to the graph colouring problem [18], a well-known NP-hard problem in graph theory [19]. For a simple graph $G = [V, E]$, colouring is about determining the number ψ of colours sufficient for assigning a color to each vertex $v \in V$ without any two adjacent vertices being assigned the same colour. Several algorithms based on contraction and sequential ordering have been developed to attack the graph colouring problem, including Largest First (LF), Recursive-Largest-First (RLF), Backtracking Sequential Coloring (BSC) and Degree of Saturation (DSATUR) [20].

The channel assignment problem is isomorphically mapped to the graph colouring problem as follows:

- The set of access points $U = \{u_1, u_2, \ldots, u_N\}$ is mapped to the set of vertices V in the graph.
- The pairs of access points $\{(u, v) : u, v \in U\}$ within transmission and reception range of each other is mapped to the set of edges E.
- The set of available channels $C = \{c_1, c_2, \ldots, c_D\}$ is mapped to the set of available colours.

We note that, depending on whether cross-channel interference is assumed to be substantial or not, the set of channels available for assignment by an access point may or may not include those with partially overlapping spectrum bands. The following section discusses prior work on these issues.

III. The NetLogo Agent Modeling Environment

A. Principles

NetLogo is an integrated modeling environment for developing multi-agent simulations in the synonymous programming language. It facilitates rapid development of simulation models with a focus on emergent behavior in systems with a large number of agents (e.g., predator-prey relationships in an ecosystem). As such, it has been used in developing models in a variety of domains, such as economics, biology, physics, chemistry, psychology and system dynamics, to name a few.

NetLogo models consist of independently operating agents in a shared finite environment. The environment is termed world and is composed as a grid of square patches. Thus it provides a two dimensional model that define the space coordinates within which agents will exhibit their behavior and interact with each other. Within this space, distance is quantified, but not qualified.

In NetLogo the evolution of time is modeled through a so-called ticks counter, which is typically incremented through integer arithmetic, though floating point increments are also possible. Thus modeling of time supports both discrete time and continuous time models, albeit the latter would require additional development effort to complete.

NetLogo also supports extensions, with several extensions being already contributed and leveraged by the research community (e.g., associative arrays, network graphs, an interface to the R projecct for statistical computing, etc.).

B. Agents

An agent in NetLogo may comprise state information (i.e., state variables and their values). The set of agents that comprise the same set of state variables form a so-called breed. In manifesting its behaviour, an agent may access the values of its own state variables, as well as the values of any global variables defined in the environment. NetLogo offers powerful filtering and selection mechanisms that operate on agent sets (i.e., breeds). These include generic primitives that allow an agent to select the set of agents (possibly out of a breed) with particular values in their state variables.

IV. Autonomic IEEE 802.11 networks in NetLogo

A. Modeling wireless communication

Aligned to the autonomic networking premise, IEEE 802.11 access points map to autonomous agents that manifest their behavior driven by their individual policies. To model system properties resulting from the wireless nature of IEEE 802.11 networks, we employed directed links between agents. Given that the propagation profile of the radio signal varies in space and time, we understand that topological proximity will be determined individually for each pair of access points. For any given pair of access points (u, v) in the IEEE 802.11 network, reception by u of a radio transmission by v will depend on the the propagation profile $R_{(u,v)}$ of the radio signal paths followed between v and u, the power P_v employed at the transmitter at v, and the sensitivity S_u of

 ISBN: 978-1-61208-331-5

the receiver employed at u. Hence, for any given pair of access points (u, v), the capacity to exchange information (i.e., to receive a transmission) is determined directionally. Consequently, for any access point, being able to receiver another access point's transmission does not necessarily imply the reverse proposition. The typical disk graph model results by relaxing the assumptions of this propagation model to the same transmission power and receiver sensitivity for all access points.

In the NetLogo environment, for any given pair of access points (u, v), we model the capacity to communicate as a link $e = (u, v)$ between the respective agents. NetLogo provides directed and undirected links between agents, thus supporting both approaches in modeling radio propagation. To control the complexity of the simulation model at this stage of our work, we opted for the disk graph model and used undirectional links between agents. This is supported elegantly by NetLogo through its builtin *in-radius* filtering primitive that selects all agents whose center coordinates are within a given distance of the center coordinates of the invoking agent in the world model. Fig. 3 illustrates a screenshot of the NetLogo model we developed for the distributed channel assignment algorithms, including simulation parameters and monitors of state information.

B. Modeling autonomous behavior

In our NetLogo model we leverage each agent's state variables (as defined in Table I) to drive the autonomous behavior of each access point. The set of available channels and the channel in use are common resources management variables included in the Simple Network Management Protocol (SNMP) Management Information Base (MIB) definitions for an IEEE 802.11 access point. The set of neighboring access points can be built and maintained by an access point that is periodically surveying its known channels for foreign Beacon management frames, a capacity that is supported by the IEEE 802.11 standard. The remaining variables in Table I are not standardized and would have to be included in the state information of an autonomous access point.

Within the NetLogo environment's control flow, the bevahiour of each access point is invoked in its respective state scope. When addressing a group of access point with a particular simulation command (e.g., to scan the radio channel and identify neighboring access points), NetLogo randomly iterates over each access point in the group. This simplifies model development by omitting timing concerns from its design and aligns to the real world situation where independently operating access points do not comply to a global timing sequence related to state changes. For instance, an IEEE 802.11 access point is able to iterate over its known channels and observe foreign Beacon management frames, but the timing sequence of these observations (e.g., the order in which known channels are observed, the amount of time spent in observing each channel, etc.) may differ from those of other IEEE 802.11 access points, even if those are geographically collocated.

TABLE I
STATE VARIABLES OF EACH ACCESS POINT.

Variable	Description
Channels	The number M of channels available for assignment to this access point.
Channel	The channel c currently assigned to this access point.
Distribution	The distribution $P = \{p_1, p_2, \ldots, p_D\}$ defining the probability of each channel to be assigned to this access point by a random selection process.
Neighbor	The set access points $N = \{u_{w_1}, w_{a_2}, \ldots, w_{a_{n(u)}}\}$ that are neighbors to this access point u (according to the disk graph model).
Conflicts	The set of access points $W = \{w_{b_1}, w_{b_2}, \ldots, w_{b_{c(u)}}\}$ that are assigned a conflicting channel to the channel assigned to this access point u.
Penalty	The percentage $b \in [0, 1]$ by which the assignment probability of an assigned channel is discounted when existence of a channel conflicts is determined.
Stability	A boolean variable q set to true when this access point is locked on to the currently assigned channel and false otherwise. For convenience, an access point locked onto its assigned channel is termed a stable access point.

Some of the channel assignment algorithms addressed here assume the capacity to exchange state information between IEEE 802.11 access points in a peer-to-peer manner, subject to sufficient radio coverage of course. The format of the Beacon Request and Radio Measurement Response management frames defined in the IEEE 802.11 specification provides for the definition of extension fields, thus supporting this assumption [3]. In this regard, the IEEE 802.11k standard provides a more efficient and versatile level of support by explicitly supporting peer-to-peer communications through the Neighbor Request and Neighbor Response management frames [7]. Hence, the assumptions underpinning the NetLogo environment regarding peer-to-peer communication between access points are properly met by the IEEE 802.11 standard and its extensions.

V. DISTRIBUTED CHANNEL ASSIGNMENT ALGORITHMS

Herein, we detail each distributed channel assignment algorithm under study, in order of increasing complexity.

Random Walk—The Random Walk (RW) algorithm provides the crudest variant of distributed channel assignment. In each iteration, the algorithm checks if any of the neighboring access points is assinged a channel that interferes with the channel assigned to its hosting access point. If such a so-called conflicting access point is found, the algorithm probabilistically selects and assigns a channel anew; otherwise, it takes no action.

Random Walk with Stickiness—The Random Walk with Stickiness (RWS) algorithm differs from the RW algorithm in that once an access point finds its currently assigned channel to be conflict-free, it locks on to that channel. This is realized by

 ISBN: 978-1-61208-331-5

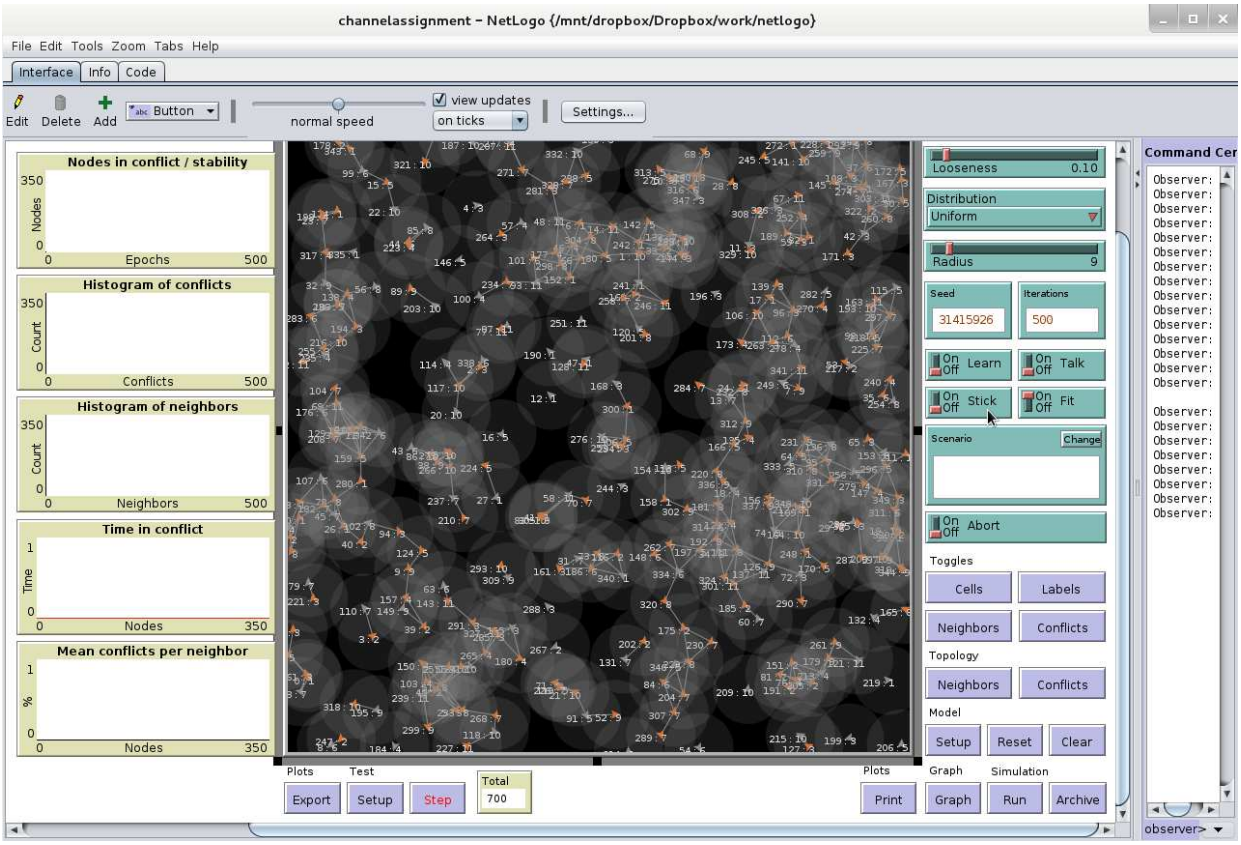

Fig. 3. Settings of the NetLogo modeling environment.

setting the assignment probability for that particular channel to one and for all other channels to zero.

Commumication Free Learning—The Commumication Free Learning (CFL) algorithm [13], [14] provides the basic learning-capable algorithm that uses past observations regarding each channel in adapting its probability of being assinged in the future. More specifically, whenever the channel currently assigned to the hosting access point is found in conflict to the channel assigned by a neighboring access point, its assignment probability is discounted by the penalty factor (see Table I).

Commumication Free Learning with Stickiness—The Commumication Free Learning with Stickiness (CFLS) algorithm [13], [14] differs from the CFL algorithm in that once an access point finds its currently assigned channel to be conflict-free, it locks on to that channel.

Learning From Communication—The Learning From Communication (LFC) algorithm [21] leverages communication with neighboring access points in adapting the probability of each channel being assigned by the access point hosting the LFC algorithm. Each access point communicates its conflict status (i.e., indicating thus whether it will probabilistically attempt to change its assigned channel in the next iteration) to its neighboring access points. By knowing which of its neighboring access points will preserve their currently assigned channel, an access point with a conflicting channel assignment is able to render its future channel assignments less prone to conflict, by zeroing out the assignment probability for all the channels its so-called stable neighboring access points are currently occupying.

Learning From Communication with Stickiness—The Learning From Communication with Stickiness (LFCS) algorithm differs from the LFC algorithm in that once an access point finds its currently assigned channel to be conflict-free, it locks on to that channel.

VI. Experiments

We establish a square world environment with an area of 160×160 units and a patch size of 4 units. Within it, we instantiate different populations of agents representing autonomous access points distributed randomly in 2D space according to the uniform distribution. Each access point runs one of the aforementioned channel assignment algorithms independently of other access points. The performance of a particular algorithm is assessed under different radius settings for the disk graph model which, in combination with the coordinates of access points, determines the connectivity of the resulting network graph (e.g., Fig. 4).

We study the performance of the CFLS algorithm by varying the radio coverage experienced by each access point and the number of access points in the environment. The CFLS algorithm is a fitting subject for this as it was also studied in [13], [14]. We consider from 50 to 400 access points

 ISBN: 978-1-61208-331-5

in increments of 50 and a radius of 10 to 60 for the disk graph model. We measure the number of iterations it takes to converge the entire network to a conflict-free channel assignment (Fig. 4).

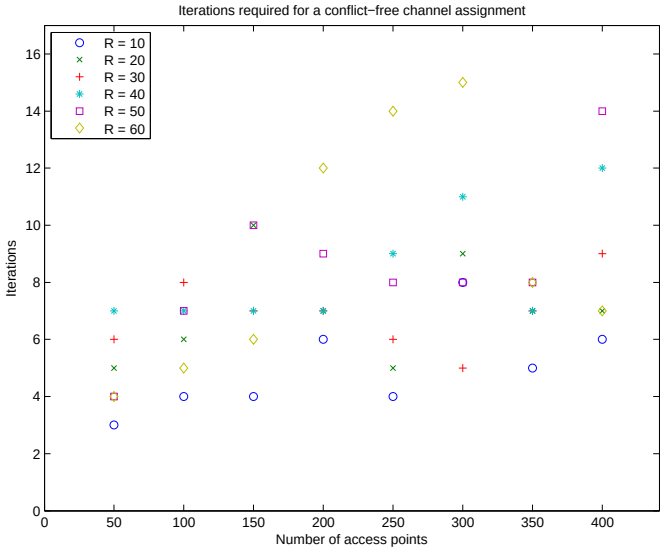

Fig. 4. Performance of the CFLS algorithm.

Aligned to our prior work in this area [21] where we leveraged the Matlab environment for simulation tasks, we find that the convergence delay of the CFLS algorithm scales nicely with an increase in the IEEE 802.11 network density.

VII. Conclusions

With the deployment of IEEE 802.11 type systems increasing in density, particularly in urban areas, attention is drawn to optimization aspects. And though the IEEE 802.11k standard has been developed specifically to support radio resource management applications, an understanding of radio resource management approaches and algorithms suitable for these settings is not fully established. To address this shortcoming, we have developed a simulation model for distributed channel assignment problems in an autonomic networking arrangement. We intend to continue studying dense IEEE 802.11 networks under a more dynamic arrangement where a mix of channel assignment algorithms may manifest.

References

[1] V. Gazis, A. Kousaridas, C. Polychronopoulos, T. Raptis, and N. Alonistioti, "Self-Management Capacities in Future Internet Wireless Systems," in *The First International Conference on Adaptive and Self-adaptive Systems and Applications*, 2009.

[2] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2005, pp. 185–199.

[3] IEEE Standards Association. (2007) IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications. [Online]. Available: http://standards.ieee.org/getieee802/download/802.11-2007.pdf

[4] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for wlans," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 3, pp. 19–31, 2005.

[5] Cisco, "Radio resource management under unified wireless networks," Online, Tech. Rep., 2010. [Online]. Available: http://www.cisco.com/en/US/tech/tk722/tk809/technologies_tech_note09186a008072c759.shtml

[6] HP, "Radio resource management and spectrum analysis best practices," Online, Tech. Rep., 2012 2012. [Online]. Available: http://h20195.www2.hp.com/V2/GetPDF.aspx%2F4AA4-4997ENW.pdf

[7] IEEE Standards Association. (2008, June) IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs.

[8] K. Leung and B.-J. Kim, "Frequency assignment for ieee 802.11 wireless networks," in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, vol. 3, 6-9 2003, pp. 1422 – 1426 Vol.3.

[9] M. Yu, H. Luo, and K. Leung, "A dynamic radio resource management technique for multiple aps in wlans," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 7, pp. 1910 –1919, july 2006.

[10] M. da Silva and J. Ferreira de Rezende, "A dynamic channel allocation mechanism for ieee 802.11 networks," Sept. 2006, pp. 225–230.

[11] J. Riihijarvi, M. Petrova, P. Mahonen, and J. Barbosa, "Performance evaluation of automatic channel assignment mechanism for ieee 802.11 based on graph colouring," Sept. 2006, pp. 1–5.

[12] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot, "Measurement-based self organization of interfering 802.11 wireless access networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 6-12 2007, pp. 1451 –1459.

[13] D. Leith and P. Clifford, "A self-managed distributed channel selection algorithm for wlans," April 2006, pp. 1–9.

[14] D. Malone, P. Clifford, D. Reid, and D. Leith, "Experimental implementation of optimal wlan channel selection without communication," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, 17-20 2007, pp. 316 –319.

[15] E. G. Villegas, E. Lopez-Aguilera, R. Vidal, and J. Paradells, "Effect of adjacent-channel interference in ieee 802.11 wlans," in *Cognitive Radio Oriented Wireless Networks and Communications, 2007. CrownCom 2007. 2nd International Conference on*, 1-3 2007, pp. 118 –125.

[16] J. Nachtigall, A. Zubow, and J.-P. Redlich, "The impact of adjacent channel interference in multi-radio systems using ieee 802.11," in *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International*, 6-8 2008, pp. 874 –881.

[17] W. L. Tan, K. Bialkowski, and M. Portmann, "Evaluating adjacent channel interference in ieee 802.11 networks," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, 16-19 2010, pp. 1 –5.

[18] J. Riihijarvi, M. Petrova, and P. Mahonen, "Frequency allocation for wlans using graph colouring techniques," Jan. 2005, pp. 216–222.

[19] R. Diestel, *Graph Theory*, 3rd ed., ser. Graduate Texts in Mathematics. Springer-Verlag, 2005, vol. 173. [Online]. Available: http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[20] W. Klotz, "Graph coloring algorithms," Technical University of Clausthal, Tech. Rep., 2002. [Online]. Available: http://www.math.tuclausthal.de/Arbeitsgruppen/Diskrete/Optimierung/publications/2002/gca.pdf

[21] V. Gazis, K. Sasloglou, N. Frangiadakis, P. Kikiras, A. Merentitis, K. Mathioudakis, and G. Mazarakis, "Cooperative communication in channel assignment strategies for IEEE 802.11k WLAN systems," in *IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications: Mobile and Wireless Networks (PIMRC'13 - Mobile and Wireless Networks)*, London, United Kingdom, Sep. 2013.

 ISBN: 978-1-61208-331-5