A Semantic Approach for Efficient and Customized Management of IaaS Resources

Luis H. V. Nakamura*, Julio C. Estrella*, Regina H. C. Santana*,
Marcos J. Santana*, Stephan Reiff-Marganiec †
* University of São Paulo (USP)
Institute of Mathematics and Computer Science (ICMC), São Carlos-SP, Brazil
Email: {nakamura, jcezar, mjs, rcs}@icmc.usp.br
† University of Leicester
University Road, Leicester, LE1 7RH - UK
Email: srm13@le.ac.uk

Abstract—**This paper presents a semantic approach to custom management of IaaS (Infrastructure as a Service) resources in a cloud computing environment requiring minimal human intervention from both the cloud provider and the user. The proposal differs from other approaches by using autonomic computing and semantic web techniques together to provide a self-configuring and self-optimizing environment that aims to satisfy SLAs (Service Level Agreements). The approach monitors the virtualized resources to guarantee a customized and optimized use based on financial criteria and energy consumption policies.**

***Keywords*-IaaS; Cloud Computing, Autonomic Computing, Semantic Web, Management;**

I. Introdution

Nowadays, efficient management for Infrastructure as a Service (IaaS) resources is a challenge in the cloud computing context. The basic idea of resource management is to allocate appropriate resources to a service according to the SLAs (Service Level Agreements) agreed between cloud computing providers and cloud computing users. Providers typically offer virtualized infrastructure instead of real hardware directly. An example of IaaS services is Amazon Web Services (AWS)[1] with processing services (Elastic Compute Cloud - EC2) and storage (Simple Storage Service - S3) [1].

Both providers and users would share the same management goals, for example, SLAs satisfaction. But they also would have different visions of an efficient management. For instance, while users would like to use as few machines-hours as possible, providers would like to have the maximum of tasks running in the mimimum of real machines to save energy and consequently increase profit.

There are several resource management approaches for cloud computing providing efficient solutions [2][3][4][5][6] for different aims like financial (economic-based/marketing-based, return on investment, predictions, etc.), energy saving (Green Computing) and to avoid SLA violations (SLA-Aware). However, to provide an automatic resource management solution involving all these scopes and also customizing desired settings for both users and provider is an innovative approach.

In this paper, we propose an efficient and customized management of IaaS resources through the use of Semantic Web and Autonomic Computing concepts and techniques. We use an ontology to represent an IaaS environment, storing relevant information about virtualized resources, SLAs and custom polices regarding financial, energy saving and Quality of Services (QoS). Mechanisms can **M**onitor the resources and store information in this ontology; **A**nalyze this data; **P**lan actions; and **E**xecute new reconfigurations in the infrastructure (self-configuration). To perform an efficient and custom management, optimization algorithms consider the providers' and users' information stored in the ontology (**K**nowledgement Base) (self-optimization). This **MAPE-K**[2] *loop control* is an Autonomic Computing concept [7] used to provide self-management in computer systems.

This paper is structured as follows: Section II presents a short review of the concepts involved in this study. In Section III the proposed framework is briefly described. Related work is discussed in Section III-D. Finally, Section IV presents the conclusions and future work.

II. Background

A. Semantic Web

Semantic Web proposes a vision in which information is delivered explicitly allowing machines to process and integrate information more easily [8]. Ontologies facilitate the representation of semantic information, avoid redundancy of information and provide a formal representation of a knowledge base. The creation and use of computational agents that are able to interpret this information and make inferences is one of the great advantages of the Semantic Web. Inferences, e.g. based on existing rules in the ontology, make these agents capable of answering questions and making decisions.

B. Autonomic Computing

Autonomic Computing is a computing paradigm inspired by biological systems (e.g., autonomic nervous system) that aim to handle the administration of complex systems offering

[1]https://aws.amazon.com/

[2]Monitor, Analyze, Plan, Execute and Knowledgement

possibilities for self-management, minimizing the need for human intervention [9]. The autonomic computing concept defines four properties; *Self-Configuration*, *Self-Optimization*, *Self-Healing* and *Self-Protection*.

The autonomic computing theory also defines a *MAPE-K loop control* mentioned earlier in this paper. However, this theory does not describe what features or technologies must be used. At this point, we can use some features from the Semantic Web, for example, an ontology can be used as a knowledgement base and computional agents can use inference process and rules to plan, analyze and make decisions.

III. Framework for IaaS Resource Management

Although autonomic computing theory covers four properties, In the scope of this paper, focus on *Self-Configuration* and *Self-Optimization*. *Self-Healing* and *Self-Protection* are to be addressed in future work.

Our proposal uses an ontology to represent the knowledge. Monitors observe the cloud infrastructure to obtain relevant information that will be stored in the ontology. Next step, this information is recovered and by using inference mechanisms and custom rules we determine whether reconfiguration is necessary. Before executing any change in the infrastructure, optimization algorithms plan efficient utilization of the available resources. In the whole process the restrictions/constraints included in the rules by the providers or users are respected, enabling a customized configuration.

A. Ontology

The structure of an ontology should reflect the domain of its application in the real "world"; here the ontology should represent an IaaS infrastructure model. Using the Web Ontology Language (OWL)[8] it is possible to add create richer vocabularies to describe the classes, relationships between classes, comparison between classes, cardinality constraints and characteristics of the properties. An OWL class represents an element in the domain, for example, an instance of the class "Machine" would represent a machine. Also the "Machine" class has two subclasses ("Real" and "Virtual") and it can be related to another subclass from the "Resources" class like: "Compute", "Memory", "Network", "Storage" or "Volume".

The ontology (Figure 1) is an extention of an unified taxonomy for IaaS proposed by Dukaric and Juric [10].

B. Mechanisms

Mechanisms to monitor, analyze, plan and execute tasks that ensure self-configuration and self-optimization of the infrastructure.

1) Self-Configuration: The self-configuration mechanism must have at least a purpose that can be governed by static or dynamic policies. As an example of static policies, we can consider the hibernation of an idle computer (less than 10% utilization) for resource savings or increasing memory when this feature become bottleneck (90% utilization or high execution of swapping). Static policies are set at design

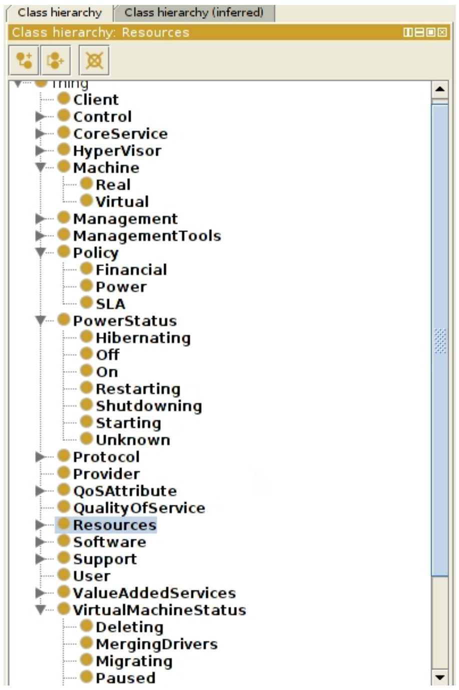

Fig. 1. Proposed Ontology View.

time, during configuration of the mechanisms by provider's administrators together with users of the system.

Dynamic policies do not use fixed values for self-configuration, but rather use a stimulus requiring a self-configuration stating what needs to be reconfigured. This stimulus is usually part of other mechanisms that form the autonomous system, such as requiring a new configuration to increase performance (e.g., self-optimization).

The computional agents monitor the infrastructure resources continuously. The proposal for the agent development is to use JADE (Java Agent DEvelopment Framework)[3] which is a distributed framework that has support to implement multiagent systems. The initial prototype uses scripts that execute the Virsh Tool[4] to change the configuration of the resources.

As mentioned earlier, the mechanisms use semantic web resources to store information in an OWL ontology and make decisions using reasoners and rules (e.g., Pellet[5] or Hermit[6] and SWRL[7] respectively).

2) Self-Optimization: Self-optimization allows detecting ideal behaviors, adapting the system voluntarily for better configuration and is a very effective proposal to satisfy the user needs [11].

Mechanisms for self-optimization such as policies and optimization techniques seek to satisfy certain requirements of the system efficiently. Policies indicate what should be optimized

[3]http://jade.tilab.com/
[4]http://libvirt.org/virshcmdref.html
[5]http://clarkparsia.com/pellet/
[6]http://hermit-reasoner.com/
[7]http://www.w3.org/Submission/SWRL/

when this optimization should occur and additionally may include restrictions on optimization techniques that may assist in achieving the problem goal.

The mechanisms for self-optimization should initially maximize performance and minimize costs. For this, the policies involved should be related to the custom criteria (SLA, QoS, Financial and Energy saving). Optimization algorithms available in the literature (e.g., Ant Colony Optimization, Integer Linear Programming, Combinatorial and Allocation Optimization, etc.) provide suitable mechanisms.

C. Architecture

The proposed architecture is shown in Figure 2. The mechanisms in this architecture provide the following functional steps:

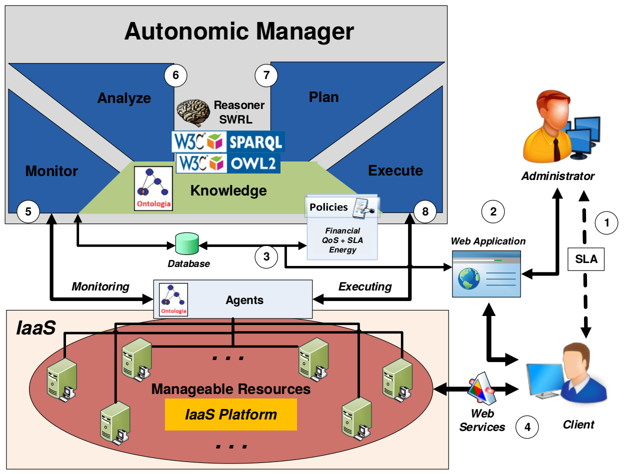

Fig. 2. Proposal Architecture

1) The first step is the creation of a SLA between the cloud client and the cloud provider's administrator. This process is accomplished outside of the architecture, but the agreed parameters should be included in the next step;
2) The administrator will use a web application to create a client account, register the financial constraints, SLA and Energy parameters. The client can choose some custom policies regarding financial issues (e.g., restrictions and constraints about machine-hours consumption, limits on the number of online VMs or boost performance for special dates);
3) All configurations from step 2 are save in a database and transformed in static policies and parameters to be used in SWRL rules;
4) Using a web service, an API or a web application (Dashboard) the client will create an infrastructure. The users' client will use this infrastructure to consume services and the workload changes will be noted by the autonomic mechanisms (next steps);
5) Computational agents monitor the infrastructure constantly and collect information that is stored in the knowledge base (KB) (ontology);
6) The *loop control* will be executed from time to time, making inferences based on the data from the KB and SWRL rules;
7) The inference process classifies and relates the information from provider, clients, resources and polices to be used by the optimization algorithms. These algorithms should analyze the data and decide if a new reconfiguration is necessary. They also need to optimize the reconfiguration itself by scaling resources appropriately and respecting the custom policies;
8) Finally, mechanisms trigger the computational agents to promote the custom auto-scaling.

The relevant contribution of this architecture and its mechanisms is the ability to create and apply new policies dynamically. For example, Amazon, one of the biggest players in cloud computing, simply offers a system for dynamic scaling based on load balancing that considers only three types of adjustments (AdjustmentTypes): *ChangeInCapacity*, *ExactCapacity* and *PercentChangeInCapacity* [12] and, basically, three kinds of instances (ondemand, reserved and spot) for billing [13]. Users need handling with both auto-scaling and billing to get a simple financial control which, sometimes, is not suitable to them.

The framework presented here offers mechanisms to the provider and user create personalized polices. There are three policies: Financial, Power and SLA.

The Financial policy should determine how much an user can spend and how the financial value is distributed among the contracted period.

The Power policy is more related to the provider side and it is supported by an optimization mechanism which deals with a combinational and allocation problem. The user can choose to agree or not with a Power policy. If the user agrees then their virtual machines may be migrate when necessary to save energy and consequently its SLA will be relaxed, but on the other hand the provider may offer a discount.

The SLA policy controls the service level agreed between user and providers, this policy uses the values of QoS attributes which were collected by the monitors in each virtual machine and composes all this information in a single QoS index for the entire user's infrastructure to determine if the SLA is been respected.

D. Related Work

Resource management in cloud computing has become a very interesting research topic because it involves other issues such as economic factors, rational use of energy, auto scaling, customization, etc.

In this section, we discuss some related work:

[6] proposes an algorithm for energy efficiency in service centers. Although the authors have not mentioned the "cloud computing" term, the work applies in data centers and cloud providers. They propose an algorithm based on context awareness as a solution for energy efficiency at run time through self-adaptation based on autonomic computing.

[2] proposes two self-configuration approaches for IaaS, one based on a predictive model (Holt-Winter) and the other on computational agents to monitor the VMs resources performance in real time. [3] presents an adaptive configuration resource management for cloud infrastructures with the aim to investigate SLA violations. To archive their objectives, the authors use two approaches one using CBR (Case-Based Reasoning) and other using semantic web (ontology) and rules, the results proved that the rule-based method is better than the CBR in respect to SLA and performance.

[4] provides a framework to manage tasks. The authors use semantic web concepts to disseminate context. However, that work does not directly address cloud computing, the focus is autonomous collaborative networks. [5] presents a proposal to provide self-managing applications for platforms of cloud computing (PaaS). The authors also use semantic web and autonomous computing concepts, presenting a new vision where cloud platforms are seen as networks of distributed data sources (sensor network).

Although these studies share ideas with our proposal, there are notable differences. For example, our focus is on the infrastructure layer (IaaS) and we offer custom management. Also, the approaches focus only on some management goals. Table I shows a summary of related work comparing to this paper.

TABLE I
BRIEF LIST OF RELATED WORK WITH CLOSEST APPROACHES

	Autonomic Computing		**Semantic Web**		**Management**			
Work	**Concepts**	**Knowledge**	**Ontology**	**Rules**	**Energy**	**Financial**	**QoS/SLA**	**Optimization**
Anghel [6]	Yes	Ontology	Yes	SWRL	Yes	N/A	Yes	N/A
Latre [4]	Yes	Ontology	Yes	SWRL/Jena	N/A	N/A	Yes	N/A
Dautov [5]	Yes	Ontology	Yes	SWRL	N/A	N/A	Yes	N/A
Our Proposal	Yes	Ontology	Yes	SWRL	Yes	Yes	Yes	Yes

IV. CONCLUSION AND FUTURE WORK

This paper proposes an semantic approach for automatic and customized management of IaaS resources in cloud computing. The archtecture and its mechanisms presented in this paper work according to autonomic computing concepts and use semantic web technologies to provide self-configuration and self-optimization features.

In our approach, providers and users are able to create and apply new customized policies dynamically with the aim of obtaining a more adequate resource management.

In future work, we intend to use the OpenStack, a cloud computing platform, instead of scripts and the Virsh tool. We will use the Openstack API to reconfigure the distributed infrastructure and take advantage of the benefits offered by this platform. In addition, we will conduct performance evaluations in our prototype. Finally, other autonomic computing concepts as *Self-Healing* and *Self-Protection* could be contemplated in the future.

ACKNOWLEDGMENTS

The authors thank the Brazilian Foundations CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo, Process: 2011/12670-5) for the support given to the development of this work. S. Reiff-Marganiec was on study leave from the University of Leicester while some of this work was conducted.

REFERENCES

[1] S. Patidar, D. Rane, and P. Jain, "A survey paper on cloud computing," in *Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on*, 2012, pp. 394–398.

[2] A. S. Dias, L. H. V. Nakamura, J. C. Estrella, R. H. C. Santana, and M. J. Santana, "Providing IaaS resources automatically through prediction and monitoring approaches," in *Computers and Communications (ISCC), 2014 IEEE Symposium on*, June 2014, pp. –.

[3] M. Maurer, I. Brandic, and R. Sakellariou, "Adaptive resource configuration for cloud infrastructure management," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 472 – 487, 2013, special section: Recent advances in e-Science.

[4] S. Latre, J. Famaey, J. Strassner, and F. D. Turck, "Automated context dissemination for autonomic collaborative networks through semantic subscription filter generation," *Journal of Network and Computer Applications*, pp. 1405 – 1417, 2013.

[5] R. Dautov, D. Kourtesis, I. Paraskakis, and M. Stannett, "Addressing self-management in cloud platforms: a semantic sensor web approach," in *Proceedings of the 2013 international workshop on Hot topics in cloud services*, ser. HotTopiCS '13. New York, NY, USA: ACM, 2013, pp. 11–18.

[6] I. Anghel, T. Cioara, I. Salomie, G. Copil, and D. Moldovan, "An autonomic algorithm for energy efficiency in service centers," in *Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference on*, aug 2010, pp. 281 –288.

[7] Y. Cheng, A. Leon-Garcia, and I. Foster, "Toward an autonomic service management framework: A holistic vision of soa, aon, and autonomic computing," *Communications Magazine, IEEE*, vol. 46, no. 5, pp. 138 –146, may 2008.

[8] W3C, "Web ontology language overview," 2004. [Online]. Available: http://www.w3.org/TR/owl-features/

[9] A. Khalid, M. Haye, M. Khan, and S. Shamail, "Survey of frameworks, architectures and techniques in autonomic computing," in *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on*, april 2009, pp. 220 –225.

[10] R. Dukaric and M. B. Juric, "Towards a unified taxonomy and architecture of cloud frameworks," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1196 – 1210, 2013, special section: Hybrid Cloud Computing.

[11] S. Hassan, D. Al-Jumeily, and A. Hussain, "Autonomic computing paradigm to support system's development," in *Developments in eSystems Engineering (DESE), 2009 Second International Conference on*, 2009, pp. 273–278.

[12] Amazon, "Developer guide - dynamic scaling," 2010, acessed in August 03, 2014. [Online]. Available: http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-scale-based-on-demand.html

[13] Amazon-EC2, "Amazon ec2 pricing," 2014, acessed in August 03, 2014. [Online]. Available: http://aws.amazon.com/pt/ec2/pricing/