SLA4CLOUD: Measurement and SLA Management of Heterogeneous Cloud Infrastructures Testbeds

El Hadi Cherkaoui, Elie Rachkidy, Marcelo F. Santos, Paulo A. L. Rego, Javier Baliosian, Jose N de Souza, Nazim Agoulmine

SLA4CLOUD: Measurement and SLA Management of Heterogeneous Cloud Infrastructures Testbeds

El Hadi Cherkaoui[1], Elie Rachkidy[1], Marcelo Santos[2], Paulo A. L. Rego[3], Javier Baliosian[4], Jose N. de Souza[3] and Nazim Agoulmine[1]
[1]IBISC/LRSM Lab, University of Evry Val d'Essonne, France.
[2]Federal University of Pernambuco, Brazil.
[3]Federal University of Ceara, Brazil, Brazil.
[4]University of the Republic, Uruguay
E-mail: cherkaoui,rachkidy,agoulmine@ibisc.fr, marcelo@upfe.br, paaloalr,neuman@ufc.br, javierba@fing.edu.uy

***Abstract*—There is an increasing number of cloud platforms emerging in both academia and industry. They often allow the collaboration of a pool of resources from multiple infrastructures (IaaS) in order to benefit from the unique features that each presents. AmSud SLA4CLOUD project is a collaboration between research groups from South America and France on Cloud Computing with the aim to develop different offers of Cloud Service with Service a Level Agreement (SLA) representation. This project builds on different existing projects such as the EU EasiClouds project. After introducing the main capabilities and features of OpenStack, this document addresses the integration of OpenStack-based platforms into a larger and heterogeneous multi-cloud infrastructures distributed in different continents. Finally, we aim to implement a strategy for dynamic services composition and optimal placement of virtual machines in order to improve network capabilities without compromising performance requirements as specified in a SLA.**

***Index Terms*—cloud computing, Service Level Agreements, Smart Placement Algorithms, IaaS, Openstack, Testbed**

I. Introduction

Cloud computing introduces interactions between cloud providers (or infrastructure providers IaaS) and cloud service providers [13]. Many existing cloud service providers (e.g. Amazon, Rackspace or GoGrid) offer to their customers the possibility to deploy services in different datacenters located in different world regions. These datacenters are different in term of hardware capacity and support different variants of cloud services. However, although the same provider manages these datacenters, they are independent systems in terms of resources management, scheduling and provisioning. Each datacenter can be considered as a complete cloud infrastructure capable of providing full cloud capabilities. In an IaaS environment, a cloud provider acquires a hardware infrastructure such as main frames, servers, networks and storage equipments. The cloud provider deploys then an IaaS cloud platform such as Openstack [5], OpenNebula [10] or AmazonS3 [1]) which permits to manage the pool of physical resources, and make them available on-demand to cloud consumers (service providers) through a set of service interfaces and computing resource abstractions (virtual machines and virtual network interfaces). Different deployment models may exist. Each one defining how exclusive the computing resources of these clouds are made available to a cloud customer, how their usage is priced, how multi-tenancy is achieved,..., etc. allowing Cloud Providers to define their own business models.

In SLA4CLOUD project, we aim to build an environment where a customer could request the deployment of its services anywhere in the underlying infrastructure given a set of defined constraints. Services can be modeled as service graphs of service components that could be deployed in different datacenters located in France, Brazil and Uruguay, depending on the SLA objectives such as localization objectives, QoS objectives or pricing. This paper describes some preliminary experimental results as well as some benchmarks. The rest of this paper is described as follows. In Section II we present an overview of the SLA4CLOUD project highlighting the objective and the requirements of the project in term of resource provisioning and self-healing. In Section III, we introduce the concept of service composition and smart placement algorithms identifying the main components that have developed on the top of the orchestration system. Next, we describe the infrastructure platform of the SLA4CLOUD testbed. Finally, conclusions and

future works on the project are presented.

II. Project Overview and Global Architecture

We propose in this project (i) the development of different offers of Cloud Service with a SLA representation that could be used for offline and on-line negotiation in cloud environments; (ii) the implementation of a strategy for dynamic consolidations of virtual machines in order to reduce energy consumption without compromising performance requirements concerning availability and SLA violation; (iii) the development of a semi- or fully-automated security policy composition mechanism for composite services in Cloud, while maintaining consistency with the security policies of the external services; (iv) a rule-based pricing system that implements the same intuitive ideas in the shape of policy-rules to improve the quality of service and to increase the global income of a Cloud Computing provider; and v) to deploy the developed mechanisms in a Mobile Cloud Computing scenario as case study and proof-of-concept demonstrator. To achieve the above goals, we began the implementation of a Cloud Computing architecture described below.

A. Backgrounds

IaaS cloud platforms are software solutions installed (completely or partially) on servers in order to manage the underlying physical resources and offer the cloud consumer a set of services. These services are accessible via APIs, where each API require certain authorizations from the cloud consumer in order to be used. Differences between IaaS cloud platforms lies in the virtualization system (hypervisors they support), the set of services provided by APIs, the users managing techniques, and the network configurations. Virtualization shifts the thinking from physical to logical infrastructure, where physical resources of a set of hardware components (e.g. physical servers) are considered as logical resources rather than separated physical resources. Therefore virtualization creates an abstraction layer between actual computing, storage and network hardware, and the software running on them. Thus allowing different operating systems contained in isolated virtual machines running on the same physical substrate. This abstraction layer is called virtualization layer which is created and managed by a software or firmware component known ‘hypervisor”.

Thus, to meet the project goals, we have general definitions and components in our architecture:

- Cloud Developer: It is the type of user that requests resources available in the cloud to the cloud provider. A request is a set of virtual machines or services that have restrictions based on SLAs (Service Level Agreements).
- SLAs: We consider a Service Level Agreement (SLA) as a document of performance expectations, responsibilities and obligations between the cloud provider and its customers.
- Requests: The Open Cloud Computing Interface (OCCI [4]) is one of the standards for modeling cloud computing resource. Physical and virtual resources are modeled following the API provided by OCCI.
- REST API: The system uses an interface RESTful [7] to perform the communication between modules. With the use of a REST API is possible to implement the system modules independently, for example, a module may be implemented in JAVA and another Python.

The global architecture of the SLA4CLOUD project is presented in Figure1.

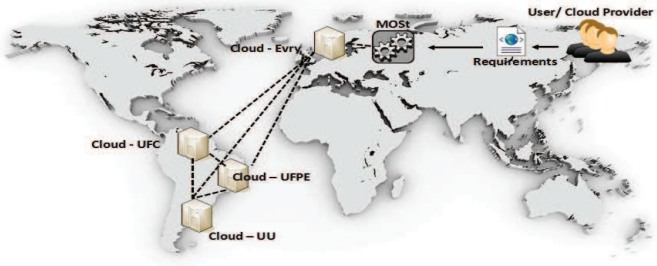

Figure 1. Entities Overview for the SLA4CLOUD Testbed

The architecture implemented between universities is based on the OpenStack platform. OpenStack is a complete platform for Cloud Computing initially developed by Nasa and Rackspace [6]. Currently, OpenStack has more than 200 members integrating the list of contributors of its project as Cisco, IBM, Dell, VMware, AT&T and Ericsson. Due to the maturity of the development of its OpenStack software was used as a tool for the deployment of Cloud Computing System in France and Brazil. OpenStack modules that were used have a brief description below and the interaction between universities can be seen in Figure2:

- Identity (Keystone):Operate as an authentication system that supports multiple forms of authentication and provide four services: (1) Identity; (2) Token; (3) Catalog and (4) Policy.

- Image (Glance): provides services for discovering, registering, and retrieving virtual machine images. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.
- Compute (Nova): is a cloud computing fabric controller, the main part of an IaaS system. Individuals and organizations can use Nova to host and manage their own cloud computing systems.

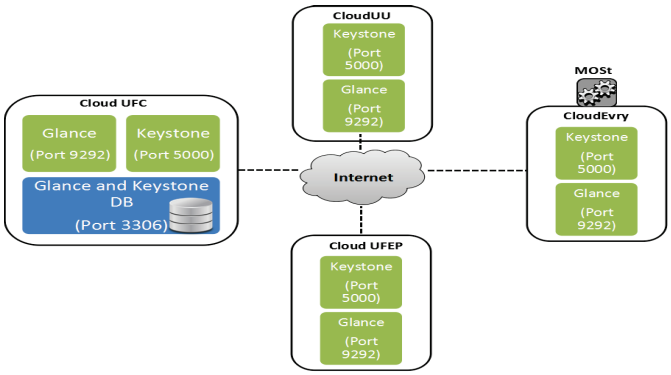

Figure 2. The global architecture of both OpenStack sites in Brazil and France

As the installation was performed independently at each university, the version of OpenStack installed at each institution was different. The Computer Science Department of Federal University of Ceara (UFC), Brazil, has adopted a hybrid cloud to assist the resources management and experimentation. The infrastructure is being used for educational purpose and to develop research in several areas such as benchmarking of cloud infrastructures, mobile cloud computing, quality of service for management system database. Icehouse version of Openstack and OpenNebula were configured in *CloudUFC*. Whereas, in University of Evry in France, Folsom version of Openstack was used for the resource allocation of the University facilities and for deployment of scientific and distributed applications in *CloudEvry*. There is a plan to add a new cloud platform in Recife at Federal University of Pernambuco, using the OpenNebula platform integrated with OpenStack.

B. Service Requests and Multisite Orchestration System

Multisite Orchestration System (MOSt) [3] is a standalone component developed under the umbrella of the ITEA EU EASI-CLOUDS project [2] with the aim is to provide an optimal provisioning plan in a distributed cloud infrastructure of a service request in form of a service graph. The service graph represents the request of a customer (called MANIFEST) in terms of a set of basic services to instantiate (i.e. virtual machines, storage, computation,..., etc.) and the links between these services (i.e. required network links between the nodes, available bandwidth,...,etc.). MOSt permits to find the best placement of the graph service onto a networked cloud infrastructure while minimizing the mapping cost. Substrate nodes in the network graph are sites located in different geographical locations as it is show in figure 1. Each site is a full IaaS cloud operated by a Site Manager which has its own service portfolio and pricing (this may depends on the effective cost of the datacenter CAPEX/OPEX in a particular region or country) described in the SLA contract.

MOSt is responsible for the global provisioning only, meaning that it will decide which service component will be initiated where (i.e. which datacenter site). Local provisioning depends on the local implementation, therefore sites may use different scheduling mechanisms to deploy service component on each country. Therefore the core system of the MOSt is responsible for building the provisioning plan and requesting the underlying sites to instantiate the resources based on the SLA terms.

The main requirements that are presented in the design of MOSt are:

- Multi-site deployment: MOSt must be capable of deploying a complex service over different cloud datacenters (called cloud sites or sites) that are geographically distributed.
- Optimal provisioning plan: MOSt must provide the customer with the best (optimal) provisioning plan and relies on an algorithm called IGM (Iterative Graph Mapping) described in [12].
- Site Independence: MOSt should be independent from any underlying site. This means that the system should be stand-alone and not connected by the internal details.

The handling of a customer service request deployment is achieved in three phases:

- Multi-site provisioning phase: MOSt calculates the optimal provisioning plan and engage the resources (e.g., Virtual Machines) in the multiple underlying sites
- Post-configuration phase: MOSt launches the post-configuration of the deployed virtual machines based on the customers request (i.e. Manifest in OpenStack).
- Networking provisioning phase: MOSt launches the configuration of the network connections between datacenters sites to fulfill nodes communication requirements of the service. This requires the instantiation of specific network gateways in each site to build the necessary VPL (Virtual Private Links) to

allow Virtual Machines (VMs) of the same project to communicate.

MOSt system is responsible for:

- The global provisioning only: meaning that it will decide which service component will be initiated (i.e. which datacenter site).
- Allocating resources to zones/sites.
- Orchestrating other sub-components such as network configuration and post-configuration.
- Parsing requests.
- Initiating deployment of requested applications in different zones.
- Consulting current state of deployed applications.
- For applications with reconfiguration, requesting/deleting more resources in case of scale up/down.

III. Service composition and Smart Placement Algorithms

Network Management strategies rely on dynamic allocation mechanisms over physical resources for efficient virtual network deployment. For this purpose, several algorithms have been proposed to achieve a convenient use of network resources such as links, routers, and nodes, mainly due to the NP-hard characteristic of the problem [8] [14] [9].

Although efficient network resource allocation is a fundamental question to be addressed, it is important assess the risks involved with a virtual network request. Obviously risks are inherent to physical infrastructure (nodes and links) because its hardware is prone failure as well as the respective software infrastructure. Thus, to perform the allocation of resources and services optimally are working on a random heuristic based on the meta-heuristic GRASP (Greedy Randomized Adaptive Search Procedures) [11] considering a set of constraints and SLAs as: node load, link Load, link delay, availability, cost, geographic position, impact on the violation of SLAs Network, ..., etc.

The resource allocation algorithm that is being developed considering SLAs restrictions is in initial testing phase. The first step is to perform simulations to validate the efficiency of the algorithm.

The considered metrics that will be collected are shown in Table I:

IV. SLA4CLOUD Platform Testbed

In this section we present the infrastructure of the SLA4CLOUD platform and the initial configuration for the testbed.

Table I
Simulation metrics

Metric	Description
Availability	Average availability of all virtual requests
Acceptance rate	Average rate of virtual request acceptance
Nodes utilization	Average physical node utilization
Links utilization	Average physical link utilization
Execution time	Average execution time for each problem

A. Infrastructure

The SLA4CLOUD architecture envisions the emergence of large scale distributed applications supported by services and processes based on distributed, integrated facilities and infrastructure among different countries covering Europe and South America. In this view, the infrastructure should be an extremely flexible programmable platform that allows new functions and capabilities to easily be created and implemented. In order to establish the previous architecture first we need full OpenStack installed in two sites, one in Brazil and the other in France. As illustrated in Figure 2, the SLA4CLOUD testbed is located in different data centers at IBISC Laboratory, Evry University (France), UFC Fortaleza (Brazil) and UFPE Recife (Brazil).

CloudEvry comprises 1 environment of production which use the Openstack's platform containing 4 racks servers. Each server contains 32 nodes of Quad core Intel 2.4GHz CPU and 8GB memory, which then connects to the outside by a 1Gb/s uplink. The Table II shows the configuration of the CloudEvry infrastructure.

Table II
Environment Summary

Production Cloud (OpenStack Folsom4.0)		
# machines	**Processor model**	**Memory**
1 controller	1x Intel(R) Xeon(R) CPU E5 2609 2.40GHz	8 GB
4 compute nodes	4x Intel(R) Xeon(R) CPU E5 2609 2.40GHz	8 GB

CloudUFC: comprises two environments. The production one uses the default OpenNebula's architecture based on front-end and compute nodes. An overview of the CloudUFC production architecture is shown in Figure 3.

For high availability purpose, 2 front-ends are used and 8 compute nodes host the virtual machines. All machines are connected through a Gigabit Ethernet network and they use a shared file system (with NFS), which allows the use of the live migration technique once all virtual machine images can be accessed from all compute nodes. Users can manage virtual resources

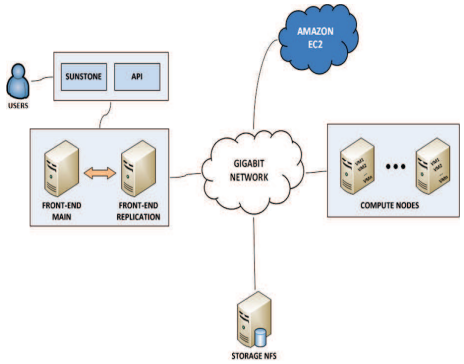

Figure 3. CloudUFC production environment (based on OpenNebula).

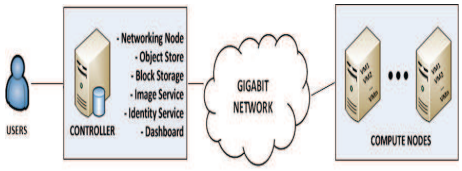

Figure 4. CloudUFC test environment (based on OpenStack).

by using the cloud operation center (Sunstone) through a website or by using the OpenNebula or OCCI APIs. The access to virtual machines can be done through SSH or VNC/Spice. CloudUFC extends local resources by using Amazon EC2 to create a hybrid cloud when necessary. It is possible to scale out the infrastructure according to research demands in a transparent way to users. Our test cloud environment uses the Icehouse version of OpenStack, and follows the architecture shown in Figure 4. This newer version presents several new features, mainly related to the management of the network stack, which provides better network isolation between the virtual machines, as well as improves the resources control, enabling the development of modules to provide better security and quality of services. The controller runs Object Store, Block Storage, Image, Identity and Dashboard services. Besides that, the controller runs also all networking services. To host virtual machines, we have two compute nodes. Users can manage virtual resources by using the dashboard (Horizon) through a website or by using OpenStack or OCCI APIs. The access to virtual machines can be done through SSH or VNC.

The local infrastructure is composed of a set of 13 heterogeneous machines, totaling 116 cores and 232 GB of RAM. Beyond these resources, Amazon EC2 instances can be initialized to fulfill the researchers needs on the production environment.

All physical machines are running Ubuntu Server 14.04 and KVM hypervisor. The Table III shows the configuration of the CloudUFC infrastructure.

B. Openstack services and MOSt Deployment

In order to deploy MOSt, first we need to create its Mysql databases containing initial information about both sites.

Then, we configure the Openstack modules such that the Keystone and Glance modules of both sites are connected to the same databases. In order to use the database deployed in *CloudUFC* as the shared database, we update the following connection string in *CoudEvry*:

connection = mysql://DB user:DB pass@Site IP/DB name in both files: /etc/keystone/keystone.conf and /etc/glance/glance-registry.conf

V. Conclusion

AmSud SLA4CLOUD project is a collaboration between France, BRazil and Uruguay to conduct research in the area of Federated Cloud Computing with emphasise on Service a Level Agreement (SLA) . This project builds on different existing projects such as the EU Easi-Clouds project. In this paper, we have presented the advances in the project in term of infrastructure deployment in different countries. In the next phase of the project, we plan to test the MOSt component from Easi-Clouds project to deploy complex services in the distributed sites. This testbed will help us also to evaluate new placement algorithms as well as new security and realibility solutions.

Acknowledgment

This research is partially funded by the EU EASI-CLOUDS project (ITEA 2 #10014) and STIC-AmSud SLA4CLOUD project(14 STIC #11). Thanks to all the partners of the project who have helped with their discussions to improve the research work presented in this paper.

References

[1] *AmazonEC2 Available on: http://aws.amazon.com/ec2.*
[2] *EASI-CLOUDS Available on: http://easi-clouds.eu/.*
[3] *Multisite Orchestration System (MOSt) Available on: https://www.lrsm.ibisc.univ-evry.fr/MOSt-api.*
[4] *Open Cloud Computing Interface Open Standard Open Community Available on: http://occi-wg.org/.*
[5] *OpenStack Cloud Software Available on: http://www.openstack.org.*
[6] Rackspace available on: http://www.rackspace.com.
[7] Subbu Allamaraju. *Restful web services cookbook: solutions for improving scalability and simplicity. "* O'Reilly Media, Inc.", 2010.

Table III
ENVIRONMENT SUMMARY OF CLOUDUFC

Production Cloud (OpenNebula 4.0)		
# machines	**Processor model**	**Memory**
2 controllers	1x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	8 GB
5 compute nodes	2x Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	16 GB
2 compute nodes	2x Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	32 GB
1 compute node	2x Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	16 GB
Test Cloud (OpenStack Icehouse)		
# machines	**Processor model**	**Memory**
1 controller	1x Intel(R) Xeon(R) CPU X3430 @ 2.40GHz	8 GB
2 compute nodes	1x Intel(R) Core(TM) i7 CPU 930 @ 2.80GHz	24 GB

[8] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking (TON)*, 20(1):206–219, 2012.

[9] Andreas Fischer, Juan Felipe Botero, M Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *Communications Surveys & Tutorials, IEEE*, 15(4):1888–1906, 2013.

[10] Dejan Milojičić, Ignacio M Llorente, and Ruben S Montero. Opennebula: A cloud management tool. *IEEE Internet Computing*, 15(2):0011–14, 2011.

[11] Mauricio GC Resende and Celso C Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of Metaheuristics*, pages 283–319. Springer, 2010.

[12] Khanh-Toan Tran, N. Agoulmine, and Y. Iraqi. Cost-effective complex service mapping in cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1–8, April 2012.

[13] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

[14] Fida-E Zaheer, Jin Xiao, and Raouf Boutaba. Multi-provider service negotiation and contracting in network virtualization. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 471–478. IEEE, 2010.