Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 19 (2013) 116 – 123

Procedia Computer Science

The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013)

Experimental Evaluation of OpenStack Compute Scheduler

Oleg Litvinski and Abdelouahed Gherbi

Department of Software and IT Engineering, École de technologie supérieure (ÉTS), Montreal, Canada

Abstract

In the Cloud Computing model, the computing resources are provided as a service in an on-demand and dynamic fashion. Efficient and flexible resource management is among the current research issues in the cloud computing context. We present in this paper a study focusing on the dynamic behavior of the scheduling functionality of an IaaS cloud built using OpenStack. Resorting to the principles of Design of Experiment (DOE), we use a screening design applied on a small-scale private Cloud in order to explore OpenStack Compute Scheduler. We present and discuss in this paper the main outcomes of an enhanced two-level fractional factorial balanced design and we outline our future work.

Keywords: Cloud Computing; Scheduler; OpenStack; Design of Experiment;

1. Introduction

Cloud computing enables the provision of computing resources dynamically. A cloud is a large-scale distributed system which provides computing resources as a service. In this computing model, the services provided are generally classified into different levels such as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-service (SaaS) [3]. The Cloud computing model is implemented using virtualization as a key technology. In particular, virtualization enables to achieve a flexible and efficient resource usage and management [1] [2].

Some open source cloud platforms are nowadays available and can be used to deploy private IaaS clouds. In order to undertake our research, we are using OpenStack [4]. We describe the architecture of OpenStack more details subsequently with a focus on its scheduling functionality, which is the main subject to our present study.

Cloud computing presents several challenging research issues. These include, in particular, how to achieve an efficient, flexible and dynamic resource management at the IaaS level. In this context, we have undertaken an empirical study of the scheduling functionality in order to characterize its dynamic behavior. In this paper, we present the results of a Design of Experiment (DOE)-based screening experiment [5]. The objective of this experiment is to evaluate the OpenStack Scheduler behavior with the aim at identifying the factors (i.e. the memory and the CPU cores assigned to a virtual machine (VM) and the memory and the number of cores on a physical node) and their interactions. We have achieved this using a two-level fractional factorial balanced with the resolution IV and four center points experimental design with no replication. In order to carry out this experiment, we have introduced a lightweight extension to the architecture of the OpenStack. This extension is outlined in this paper.

The remaining part of this paper is organized as follows. In Section 2, we present an overview of the software architecture of OpenStack. We present in Section 3 our OpenStack-based experimental IaaS Cloud and introduce our extension to enable the interaction with OpenStack Compute and its scheduler. We detail in Section 4 our DOE-based methodology. In Section 5 we present the main outcomes of the experiment. We discuss and interpret these outcomes in Section 6. We present a succinct overview of the related work in Section 7. Finally, we conclude our paper in Section 8.

2. Openstack architecture

The objective of open source project OpenStack is to build a "massively scalable cloud operating system" [4]. With elasticity and horizontal scalability in mind, all the OpenStack services and their components follow a shared-nothing, asynchronous messaging (local and queue-based) and distribute-everything guidelines.

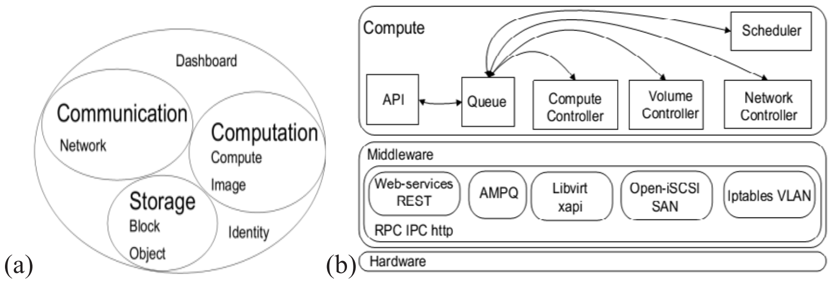

Fig. 1 (a) OpenStack Main Services (b) Compute Service and the Underlying Platform.

OpenStack is composed of seven main services which can be grouped into three principal areas: Communication, Storage, and Computation as depicted in Figure 1.a. These services are backed by two support services, namely Identity and Dashboard. Compute service manages the virtual disks and associated metadata in Image. Dashboard provides a web-based front-end to the Compute whereas Network provides virtual networking for Compute. Block Storage provides storage volumes for Compute. Image can store the actual virtual disk files in the Object Storage and all the services authenticate with Identity [4].

The Compute service consists of the following components as depicted in Figure 1.b: Web-based API ensures the command and control aspects of the computation, storage and networking. The component Queue brokers the interaction between the Compute service components, i.e., Volume, Network and Compute Controllers, Scheduler and API. Compute Controller manages the life-cycle of computing instances (i.e. VMs) on the nodes. The Network Controller manages the networking resources and the Volume Controller ensures the interaction between the instances and the Block Storage [4].

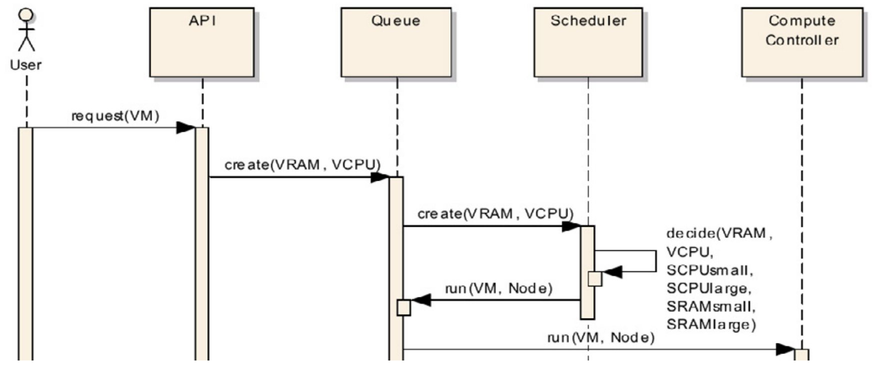

Fig.2 Request Flow

The underlying platform is composed of the middleware and the hardware as shown in Figure 1.b. The middleware includes several components such as Web-services, AMPQ, etc. These components are communicating using normalized protocols, such as RPC, IPC and http, and sharing the same hardware layer [4].

Once API receives user issued request for launching an instance (VM), it interprets this request and send it to the Queue that is providing messaging between all other components of Nova. After receiving the request for creating an instance, Scheduler decides which Node will get the instance according to the available resources (the amount of free memory and the number of available CPU on nodes) and the specification in the request (the memory and the number of CPU). The Queue dispatches this association between an instance and the Node to the Compute Controller which is launching the requested VM on one particular Node. This request flow is depicted in Figure 2.

OpenStack Compute includes three types of Scheduler: Filter, Chance and Simple. Figure 3.a illustrates the dynamic of the Filter, which is a two-stage process that supports filtering and weighting using predefined costs (c1, c2, c3) and their associated weights(w1, w2, w3) in order to schedule the deployment of instances (i.e. VM) on some physical node. The Chance Scheduler chooses randomly an available node regardless of its characteristics while the Simple scheduler tries to find an available node with the least load. Both of the Chance and Simple schedulers are using only one common stage as depicted in Figure 3.b.

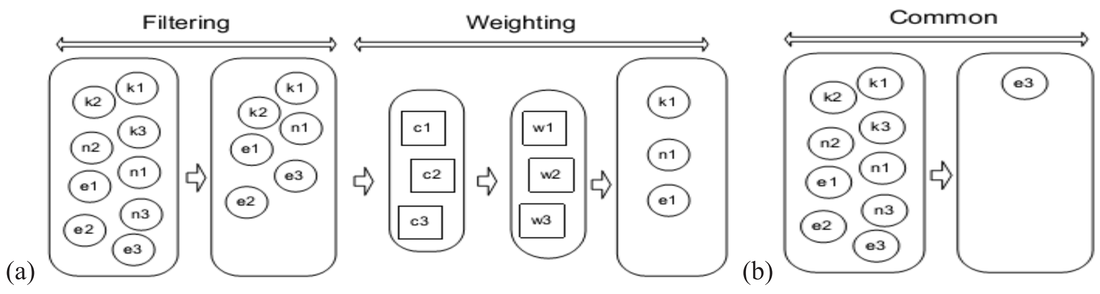

Fig. 3 (a) Filter Scheduler (b) Chance or Simple Scheduler

3. Experimental Private Cloud

Our experimental private cloud is composed of six nodes of Athlon 64 X2 dual-core at 3 GHz with 4 GB and 8 GB of RAM with the same size hard drives and dual gigabit NICs (one is dedicated to OpenStack traffic and the second one for management purposes). All the nodes are interconnected using a dedicated gigabit switch. We deployed Linux Ubuntu Server [6] with OpenStack [4]. The latter is configured using multiple-node mode using flat-DHCP networking. We apply the black-box testing approach [7] in a controlled environment with OpenStack Scheduler as the main component along with its supporting components: API, Queue, and Compute Controller. The main motivation for this is to observe the behavior of Scheduler which is shielded from any external influence.

The experimentation test bed is composed of three distinct modules which are shown in Figure 3. The Request Generator automates the generation of user requests to the API component using constant and predefined rate. Each request specifies two parameters related to the instance: the amount of RAM and the number of CPU requested. The Scheduler Observer allows gathering of information about the Schedulers activity. The purpose of the Load Generator is to automate the process of generating a measurable and controlled memory and CPU load on the nodes. These modules constitute a lightweight extension to OpenStack Compute and enable the interaction with the System under Test. The latter is composed of the following Compute's components as shown in Figure 3: the Scheduler, the Compute Controller, the Queue and the API. Upon the reception of a request for launching an instance from Request Generator module, the API places it in Queue. The Scheduler handles the request using its parameters and the available resources on all available nodes. Then, it informs the Compute Controller to assign and deploy the instance on a specific node.

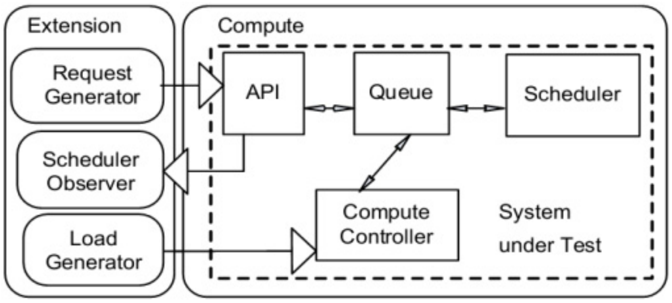

Fig. 4 Extension and System under Test

4. Design of Experiment

Design of Experiment is a branch of applied statistic with well-established principles (such as factorial, randomization, blocking and replication) that aiming at maximizing the gathering of information about a system or process through using a formal set of tests without looking at it inner parts [5]. The aim of the screening type of experiment is to define the factors with significant effects on the responses. In our context, these factors are the amount of memory and the number of CPU cores assigned to virtual machine (VM) and the amount of memory and the number of cores on a physical node.

We have achieved this using a two-level fractional factorial balanced with the resolution IV and four center points experimental design with no replication. The objective of our experimental design is to identify the most important factors while clearly separating them from their interactions as well as estimating if second-degree polynomial model could be appropriate later on. This design is generated using the R statistical environment [7] with 20 runs (i.e. test cycle). The same design is applied to Filter, Chance and Simple schedulers. The level values are chosen in order to maximize the likelihood of identifying any causality links between the factors and responses. Table 1 shows their values in more details. We also note that VRAM and VCPU are characterizing Request Generator and the SCPU and SRAM, the Load Generator.

Table 1 Factors and their definition

Factor	Name	Description	Min	Mean	Max
A	VRAM	Amount of RAM allocated to VM (GB)	0.5	4	8
B	VCPU	Number of CPU allocated to VM (GB)	0	2	4
C	SCPU small	Number of CPU available on node with 2 core	0	1	2
D	SCPU large	Number of CPU available on node with 4 core	0	2	4
E	SRAM small	Amount of RAM available on node with 4 GB (GB)	0	2	4
F	SRAM large	Amount of RAM available on node with 8 GB (GB)	0	4	8

5. Results

In the following presentation of the results (see Table 2 and Table 3), the factors and their interactions are sorted by importance in a decreasing order with the confidence level of 0.05 as a threshold. We did not need to apply any statistical transformation to the data and we note that the results are mostly coherent in both the Adjusted and Unadjusted models.

Table 2 Responses and their confidence interval (p-value)

Model	Scheduler	CPU small	CPU large	RAM small	RAM large
Adjusted	Filter	0.0513	0.0092	0.0076	0.0091
	Chance	0.0513	0.0092	0.0036	0.0013
	Simple	0.0513	0.0092	0.1497	0.0028
Adjusted Curvature	Filter	0.4558	0.3589	0.0230	0.0134
	Chance	0.4558	0.3589	0.0145	0.0009
	Simple	0.4558	0.3589	0.8641	0.0025
Unadjusted	Filter	0.0427	0.0073	0.0236	0.0469
	Chance	0.0427	0.0073	0.0213	0.0489
	Simple	0.0427	0.0073	0.1296	0.0473

Table 3 Importance of Factors and theirs combination in the Schedulers choice of node with corresponding R^2

Scheduler	CPU small (R^2)	CPU large (R^2)	RAM small (R^2)	RAM large (R^2)
Filter	A,B,C,E,AB,AE (0.5859)	A,B,C,E,F,AC,BF (0.7460)	A,B,E,AB,AE (0.5708)	A, B, D, F,AB,AF,BD,BF (0.6865)
Chance	A,B,C,E,AB,AE (0.5859)	A,B,C,E,F,AC,BF (0.7460)	A,B,C,E,F,AB,AF,BF,ABF (0.7812)	A,B,D,E,F,AB,AF,BD,BF (0.6835)
Simple	A,B,C,E,AB,AE (0.5859)	A,B,C,E,F,AC,BF (0.7460)	A,B,BF (0.3266)	A,B,D,F,AB,AD,BD (0.6858)

Our first observation is that (I) all the OpenStack Scheduler types (i.e. Filter, Chance and Simple) have the same factors in the same order for the small and large number of CPU. In addition, (II) the amount of memory and the number of CPU are at the top of the list of factors in all responses for all the Schedulers types. Further, (III) in most cases, the factor interactions involve two factors. Finally, (IV), the behavior of Simple Scheduler is abnormal regarding the small amount of memory.

The second observation, as it is shown in Table 3, is that in almost all cases, the R^2 values are well above 50% except for small amount of memory for the Simple type of OpenStack scheduler.

Our third observation, as shown in Table 3, is that the response for large RAM is directly linked with large numbers of CPU on host node for all the scheduler types. Also, the combination of virtual CPU and large number of CPU on node appears in all the types of schedulers for the nodes with large amount of memory. In addition, the factors involved in the case of Simple and Filter schedulers are mostly the same for large amount of memory.

Our fourth observation is that for the nodes with small amount of RAM, there is a noticeable difference in the factors involved in comparison to large amount of memory.

The last observation regards the factors interactions. The large amount of memory and large number of CPU are dominating the small ones with the ratio three to one. Comparing the memory to CPU, this ratio is two to one.

6. Analysis and Discussion

The part (I) of our first observation is the most striking outcome. Considering the fact that our component Scheduler Observer does not show any relevant information about the CPU utilization, this observation may suggest that OpenStack Scheduler decision making is not based on the CPU load. The amount of RAM and number of CPU associated with an instance are the most important factors. With respect to the interactions of these factors, the large amount of CPU and memory are clearly dominating the small amount and the CPU.

Considering that all the main effects of certain factors are well defined and isolated from their interactions, we notice a large percentage of noise (see R^2, which is a ratio between predicted (modeled) values and the observed (experimental) values, in Table 3) which needs further investigation to be thoroughly explained. Regarding the part (III) of the first observation, the single case of interaction involving three factors may be discarded because of the resolution IV design.

In practical terms, the behavior of all types of Scheduler is of little difference with respect to the initial assignment of an instance. This finding corroborates the observation made by [8]. In addition, in the case of nodes with the large amount of RAM, the behavior of Simple and Filter Scheduler is almost the same.

Our conservative design aims at facilitating the interpretation of the obtained experimental results. One alternative to this choice would include some statistical techniques, such as single-factor fold-over, to untangle some points mentioned above in the follow-up experiments.

In the case of Filter Scheduler, all filters are by default and it may be interesting to study different filters combination and their relation with respect to the characteristics of the nodes. Our choice of KVM as a virtualization technology is motivated by the fact that the Linux kernel is serving as a hypervisor. This allowed carrying out load on the resources directly without the hypervisor.

The usage profile data of private Cloud are unavailable publicly. As a consequence, we have adopted a constant rate for the generation of user requests. This has simplified the investigation but with a tradeoff that the requests arrival model is unrealistic.

7. Related Work

We distinguish the following categories of research works related to our research. The first category revolves around the black-box approach [9], [10], [11]. The authors of [9] use this approach to identify the capacity of multi-tier application hosted on virtualized platforms while the authors of [10] employ this approach to investigate different parameters influencing applications running inside a VM. In [11], the authors use a modified black-box model for "run-time modeling and performance control of Web service in virtualized hosting environments".

The second category includes works focusing on the VM allocation strategy [12], [13], and [14]. The authors of [12] investigate the relationship between different categories of workloads on various virtual network configuration strategies. In [13], the authors propose and evaluate an application-centric energy-aware strategy for VM allocation in order to maximize resource utilization and energy efficiency. The authors of [14] use a probabilistic model for the same purpose but in the context of reallocation with simulated-based evaluation.

With respect to the initial assignment of a VM in the context of Cloud, the authors of [8] undertake a simulation-based comparison of a set of 18 resource-allocation algorithms for large distributed systems.

The category targeting the evaluation of the clouds scheduler includes data processing presented in [15] and [16]. In addition, the authors of [17] propose a survey and analysis of meta-scheduler and present in [18] a model to instantiate dynamically virtual machines considering the current job characteristics. The authors of [19] discuss how the OpenStack scheduler performance can be enhanced from its developer standpoints.

Finally, OpenStack is attracting an interest in the cloud computing research community. The authors of [20] present a detailed comparison of features of OpenStack with another Cloud platform, OpenNebula. In [21], the authors identify some performance issues with previous version of OpenStack, namely Cactus and in [22] the authors propose a survey of Cloud offerings in term of IaaS, including OpenStack.

8. Conclusion

In this work, we investigate the Scheduler behavior in the context of small-scale private Cloud in a controlled environment using the principles, tools and methodology of Design of Experiment. In particular, we used a two-level fractional factorial balanced design with resolution IV enhanced by four center points. In order to achieve this, we introduced a lightweight extension to the OpenStack Compute service. This extension consists in three modules, a Load and Request Generators as well as a Scheduler Observer, which allowed us to carry out our experimentation on the Filter, Chance and Simple types of OpenStack Scheduler. The main findings of this experimentation suggest that (1) the Scheduler's decision making for the initial instance assignment does not depend on the CPU load; (2) the amount of RAM and number of CPU requested are the most important factors in initial assignment of instances to the nodes in all types of OpenStack Schedulers; (3) with respect to the factor interactions, the large amount of CPU and memory are clearly dominating the small ones. Finally, we notice a large percentage of noise in all experiments which is suggesting further investigation.

Our plans for the future work include an enhancement of our current experimental methodology with techniques such as fold-over for further analysis of scheduler behavior as well as using our current design as a first step of Response-Surface Methodology [5]. Other venues include an enhancement to the Scheduler taking into consideration the dynamic state of CPU and memory load. The latter may require the assessment of Scheduler's performance with an appropriate metric. Also, the experimental data may serve for comparing Scheduler behavior with different mathematical model.

Acknowledgements

This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors are grateful to Dr. Michel Rioux from the Automated Manufacturing Engineering Department at the École de Technologie Supérieure (ÉTS) in Montreal for his input regarding theoretical foundation and practical application of Design of Experiment.

References

[1] L.M. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: towards a cloud definition, *ACM SIGCOMM Computer Communication Review*, **39** (2009) 50-55.
[2] T. Grandison, E.M. Maximilien, S. Thorpe, A. Alba, Towards a Formal Definition of a Computing Cloud, in: *Services (SERVICES-1), 2010 6th World Congress on*, 2010, pp. 191-192.
[3] B.P. Rimal, E. Choi, A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing, *International Journal of Communication Systems*, **25** (2012) 796-819.
[4] OpenStack LLC, Open source software for building private and public clouds., in, 2012, Available:http://www.openstack.org/.
[5] D.C. Montgomery, *Design and analysis of experiments*, 7th ed., Wiley, Hoboken, NJ, 2008.
[6] Canonical Ltd, (2012, august, 2nd, 2012). Ubuntu Server LTS 12.04. Available: http://www.ubuntu.com/business/server
[7] R Core Team, *R: A language and environment for statistical computing*, in, R Foundation for Statistical Computing, 2012.
[8] K. Mills, J. Filliben, C. Dabrowski, Comparing VM-placement algorithms for on-demand clouds, in: *2011 IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, 29 Nov.-1 Dec. 2011, IEEE Computer Society, Los Alamitos, CA, USA, 2011, pp. 91-98.
[9] W. Iqbal, M.N. Dailey, D. Carrera, Black-box approach to capacity identification for multi-tier applications hosted on virtualized platforms, in: *2011 International Conference on Cloud and Service Computing (CSC 2011)*, 12-14 Dec. 2011, IEEE, Piscataway, NJ, USA, 2011, pp. 111-117.
[10] G. Kousiouris, T. Cucinotta, T. Varvarigou, The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks, *Journal of Systems and Software*, **84** (2011) 1270-1291.
[11] D. Ardagna, M. Tanelli, M. Lovera, L. Zhang, Black-box performance models for virtualized Web service applications, in: *1st Joint WOSP/SIPEW International Conference on Performance Engineering, WOSP/SIPEW'10*, January 28, 2010 - January 30, 2010, Association for Computing Machinery, San Jose, CA, United states, 2010, pp. 153-163.
[12] W. Qingling, C.A. Varela, Impact of Cloud Computing Virtualization Strategies on Workloads' Performance, in: *2011 IEEE 4th International Conference on Utility and Cloud Computing (UCC 2011)*, 5-8 Dec. 2011, IEEE Computer Society, Los Alamitos, CA, USA, 2011, pp. 130-137.
[13] H. Viswanathan, E.K. Lee, I. Rodero, D. Pompili, M. Parashar, M. Gamell, Energy-aware application-centric VM allocation for HPC workloads, in: *25th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum, IPDPSW 2011*, May 16, 2011 - May 20, 2011, IEEE Computer Society, Anchorage, AK, United states, 2011, pp. 890-897.
[14] H. Sijin, G. Li, M. Ghanem, G. Yike, Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models, in: *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, 24-29 June 2012, IEEE Computer Society, Los Alamitos, CA, USA, 2012, pp. 574-581.
[15] H. Kllapi, E. Sitaridi, M.M. Tsangaris, Y. Ioannidis, Schedule optimization for data processing flows on the cloud, in: *Proceedings of the 2011 international conference on Management of data*, ACM, Athens, Greece, 2011, pp. 289-300.
[16] L.T.X. Phan, Z. Zhang, Q. Zheng, B.T. Loo, I. Lee, An empirical analysis of scheduling techniques for real-time cloud-based data processing, in: *2011 IEEE International Conference on Service-Oriented Computing and Applications*, SOCA 2011, December 12, 2011 - December 14, 2011, IEEE Computer Society, Irvine, CA, United states, 2011, pp. IEEE Comput. Soc. Tech. Comm. Bus. Informatics Syst. (TCBIS).
[17] S. Sotiriadis, N. Bessis, N. Antonopoulos, Towards inter-cloud schedulers: A survey of meta-scheduling approaches, in: *6th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC-2011*, October 26, 2011 - October 28, 2011, IEEE Computer Society, Barcelona, Catalonia, Spain, 2011, pp. 59-66.
[18] S. Sotiriadis, N. Bessis, F. Xhafa, N. Antonopoulos, Cloud virtual machine scheduling: Modelling the cloud virtual machine instantiation, in: *2012 6th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2012*, July 4, 2012 - July 6, 2012, IEEE Computer Society, Palermo, Italy, 2012, pp. 233-240.
[19] OpenStack LLC. (2012, july, 29th, 2012). Essex Scheduler and Scaling Improvements. Available: http://wiki.openstack.org/EssexSchedulerImprovements.
[20] X. Wen, G. Gu, Q. Li, Y. Gao, X. Zhang, Comparison of open-source cloud management platforms: OpenStack and OpenNebula, in: *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, 2012, pp. 2457-2461.
[21] G. Von Laszewski, J. Diaz, F. Wang, G.C. Fox, Comparison of multiple cloud frameworks, in: *2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, June 24, 2012 - June 29, 2012, IEEE Computer Society, Honolulu, HI, United states, 2012, pp. 734-741.
[22] M. Mahjoub, A. Mdhaffar, R.B. Halima, M. Jmaiel, A Comparative Study of the Current Cloud Computing Technologies and Offers, in: *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, 2011, pp. 131-134.