Eucalyptus LSS: Load-Based Scheduling on Virtual Servers Using Eucalyptus Private Cloud

Shenlene A. Cabigting and Joseph Anthony C. Hermocilla

***Abstract*— The University of the Philippines Los Baños currently offers a number of applications to its students and staffs. However, a limited number of servers and hundreds of users accessing different services at the same time would results to heavy traffic and a huge response time. A private cloud was setup to enable easy and flexible deployment of virtual servers within the network using Eucalyptus Private Cloud. A load balancer was deployed in the cloud to properly distribute task among the virtual servers, provide faster response time and prevent the possibility of data lost due to server failure. The load balancer was developed using Java and is composed of four classes: MyServlet, Balancer, Server and Request. The load balancing scheme used took into consideration not only the load and capacity of the servers but also the resources needed by each incoming request. A graphical web interface was also provided for the network administrator to visualize and monitor the current status of each virtual machine in the cloud.**

***Index Terms*— Eucalyptus, cloud computing, load balancing, virtualization, server scheduling**

I. Introduction

A. Background of the Study

Cloud computing, while still at its early stage, is already making a scene in the Information Technology (IT) industry. Continuous development in high-speed broadband internet enables this new technology to offer possibilities to both its end-users and service providers.

Cloud computing is the means by which services can easily and conveniently be accessed from a shared pool of configurable computing resources through the internet or a local area network [1]. These resources make up what is known as the cloud and consist of both the hardware and the software in the datacenters from which services are provided [2]. A cloud user must send a request to the cloud to access these services. Whenever such request is granted, a fraction of the cloud is provisioned to the requesting user and will remain dedicated to that user until it is released [3].

One way of classifying clouds is based on the portion of the resources delivered as a service. They are referred to as the three service models namely: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). They are also sometimes referred to as the cloud layers where IaaS is the lowermost layer.

SaaS provides the user access to applications deployed in the providers cloud infrastructure through a client interface and is often referred to as Software on Demand. In this service model, the user has no control over the underlying cloud infrastructure. PaaS provides the user the ability to create, manage and/or control applications in the cloud. These applications can either be created by the user or acquired from other services as long as they are supported in the cloud environment of the service provider. IaaS provides the user with their own virtual cluster on which they can provision processing, storage, networks and other fundamental computing resources and on which they can deploy and run applications and even operating systems. [1]

Another way of classifying clouds is based on the nature of access and control with respect to how the provisioned resources will be used [3]. This classification is more commonly known as the deployment models and includes the following: private, community, public and hybrid. Private cloud refers to the internal datacenter of a company or organizations sharing the same concerns. These organizations may have the same policy, security requirements, compliance considerations, etc. The public cloud infrastructure, unlike the first two deployment models mentioned, is made available to the public and is usually owned by companies or organizations selling cloud services. A hybrid cloud is the resulting infrastructure when two or more of the other deployment models (private, community and public) are combined. In this deployment model, each cloud component remains unique yet they are bound together by a standardized technology. [1]

One key characteristic of cloud computing is scalability. A cloud must be able to accommodate rapid increase or decrease in demands and elastically scale in or scale out computing resources based on these demands [3]. Applications deployed on a cloud should be available to any number of users at any time. To achieve this goal, virtualization and load balancing must be supported in every cloud.

Virtualization is the ability to run instances of virtual machines (VM) on top of a hypervisor. A hypervisor provides a uniform abstraction of the underlying physical machine allowing multiple VMs to execute simultaneously. [3] Virtualization provides the cloud the flexibility to allocate and deallocate computing resources to satisfy user requests. Load balancing, on the other hand, is the process of transparently and efficiently distributing user requests to the number of available servers [4]. With these two working hand in hand, cloud computing creates the illusion of an infinite computing resources available on demand to the users at any time [2].

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

In this study, IaaS was deployed as a private cloud to suit the University of the Philippines Los Baños (UPLB) environment. The study aimed to provide better traffic handling to the UPLB network and thus, its major concerns include both virtualization and load balancing. Since load balancing is not yet supported in most open source cloud computing platforms, a load balancer was developed using Java and was deployed in Eucalyptus Private Cloud. Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems (EUCALYPTUS) is an open source cloud computing platform that can be installed on most Linux-based systems and is compatible with the Amazon EC2 and S3 Cloud API [5].

B. Statement of the Problem

UPLB currently offers a number of services to the university students and staffs. These include courses@UPLB, UPLB Webmail, UPLB SystemOne, UPLB Payroll Services, UPLB iLib, etc. Like most applications today, these services require processing power which can no longer be provided by a single server. A limited number of servers and hundreds of users accessing different services at the same time would result to heavy traffic and a huge response time. The solution is to deploy additional servers to the UPLB network. This, however, would be too costly and is not always feasible for the university. A load balancer for the UPLB network must be developed to handle this traffic and provide faster response time for the users. To solve the problem regarding the limited number of servers, the load balancer must be deployed in a private cloud where instances of virtual servers can be added and removed with ease through horizontal scaling.

Load balancing is one of the key functionalities of cloud computing but since most open source cloud computing platforms currently do not support load balancing, open source cloud developers must each develop their own load balancers.

C. Significance of the Study

With a load balancer deployed in a private cloud, requests to applications were scheduled more efficiently to the most capable server depending on its availability and capacity in handling such requests. Better scheduling resulted to a lighter traffic within the network which, in turn, provided faster response time to applications and easier access to users.

Load balancing and virtualization also provided better risk management to the network. The load balancer helped prevent the possibility of data lost due to server failure by automatically redistributing requests to the next most capable server within the network. With load balancing, a faulty server can be removed from the network without compromising the performance of the other servers.

The study also aimed to contribute to the development of Eucalyptus and provide a load balancing feature to future open source developers.

D. Objectives of the Study

The main objective of this study is to create a system that would provide better traffic handling within the UPLB network, provide faster response time to the users and prevent the possibility of data lost due to server failure.

The study specifically aimed to:

1) Setup a private cloud on which the system will be deployed;
2) Use load-based load balancing scheme to better distribute requests to the servers; and
3) Provide a graphical web interface to enable the administrator to view the status of the traffic within the network.

E. Dates and Place of the Study

This study was conducted in the Institute of Computer Science, College of Arts and Sciences, UPLB from November 2011 to March 2012.

II. Review of Related Literature

Cloud computing is a relatively new field of study in computer science and its applications are still currently being developed. Very limited literature about this field is available on the internet and on printed materials and even less about load balancing since this feature is not readily available on most open source cloud computing platforms.

OpenNebula is an open source cloud computing platform suitable for building private, public and hybrid clouds. It was developed in 2005 by Llorente and Montero. [6] OpenNebula has no built-in load balancing feature and relies on its match making scheduler (mm_sched) in distributing task assignments. mm_sched implements the Rank Scheduling Policy where resources most suitable for a VM are first prioritized by that VM. This scheduling framework is designed to be highly modifiable and can easily be replaced by other developments. [7]

Eucalyptus is the most widely deployed cloud computing platform for building private clouds. It was developed in 2007 by Wolski. Eucalyptus is open source and is open for contributions from outside developers. This cloud computing platform currently does not support load balancing. To achieve load balancing in Eucalyptus, developers must each develop their own load balancer and run it in the Front End of their cloud setup. [7]

Nimbus, AbiCloud and vSphere are cloud computing platforms developed in 2009 [6]. Nimbus offers Infrastructure as a Service (IaaS) and was developed by Keahey. AbiCloud was developed by Abiquo. It is suitable for building and managing private and public clouds. vSphere is highly specialized in virtualization and was developed by VMWare. [8] None of these three cloud computing platforms supports load balancing.

Load balancing is one of the key functionalities of cloud computing but it is evident that most open source cloud computing platforms currently does not support this feature. In this study, a load balancer for Eucalyptus was developed. Since Eucalyptus is the most highly deployed open source cloud computing platform, it was expected that this study will prove helpful for future open source cloud developers.

III. Theoretical Framework

A. Virtual Machine (VM)

A VM is a software implementation of a machine that does the same functionalities as its physical counterpart. It has its own kernel, operating system, supporting libraries and applications. [3] A VM does not physically exist and is created within another environment to execute an instruction set different than that of the other environment [9].

B. Horizontal Scaling

In horizontal scaling, machines (physical or virtual) are added to the cluster to allow multiple simutaneous processing and distributed workload. Compared to vertical scaling, better risk management is exhibited in horizontal scaling since information does not depend on a single server. [10]

IV. Methodology

A. System Requirements

For this study, the following tools were used:

- Two (2) units of machine
- CentOS 5.6
- Eucalyptus 2.0.3 32-bit CentOS 5 RPMs
- Euca2ools 1.3.1
- Java OpenJDK 1.6.0
- Apache 2 Web Server
- Apache Tomcat 7
- Xen Hypervisor 4.1.1
- JQuery 1.5.1

B. Private Cloud Setup

Eucalyptus Cloud has three major components: Cloud Controller, Cluster Controller and Node Controller. Cloud Controller serves as the brain of the cloud and is responsible for the ”logic decisions” of the cloud. Cluster Controller is responsible for determining the current state and availability of VMs running in its corresponding cluster in the cloud. A cloud setup may consist of several Cluster Controllers, each of which is a collection of several Node Controllers. Node Controllers have direct access to the VMs and are responsible for their initialization and termination. [9] In this study, only one Cluster Controller and one Node Controller were used. Relationship among the components can be further understood in the following figure.

Two units of machine were setup in one of the Computer Laboratories of the Institute of Computer Science. One machine acted as the Front End, the other as the Compute Node. The Front End was the machine controlling the cloud and was the administrators access to the cluster. The Compute Node was where VM instances were run. Both machines were installed with CentOS 5.6 as their operating system. Both were connected to a local area network (LAN) for them to function as a cluster.

The Front End has one network interface (eth1) which was connected to the LAN. The Front End was installed with Java OpenJDK 1.6.0. The Cloud Controller, Cluster Controller and Storage Controller packages were also installed to the Front End together with Walrus and Euca2ools 1.3.1.

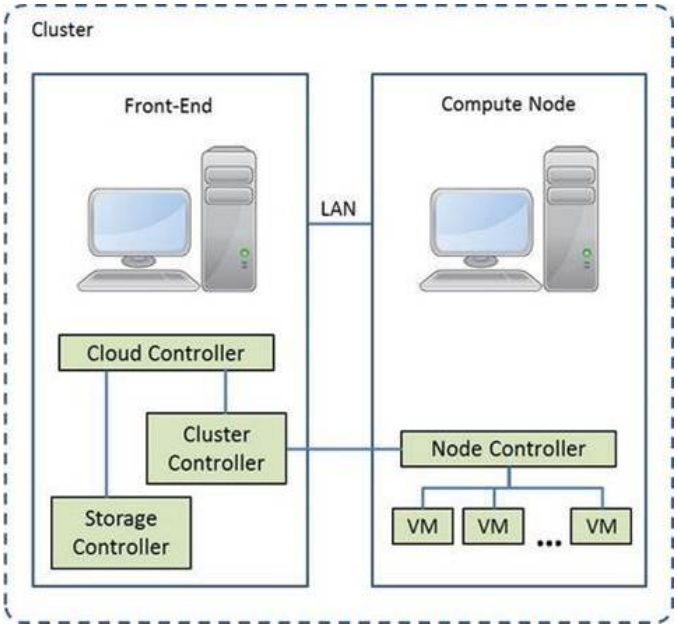

Fig. 1. Private Cloud Setup with One Cluster Controller and One Node Controller

The Compute Node has one network interface (eth0) which was connected to the Front End through the LAN. It was installed with Xen Hypervisor 4.1.1 and the Node Controller package.

Eucalyptus offers four different networking modes: MANAGED, MANAGED-NOVLAN, SYSTEM and STATIC. MANAGED and MANAGED-NOVLAN modes provide the same set of networking features: connectivity, IP control, security groups, elastic IPs, metadata service and VM isolation. The only difference between the two is that in MANAGED mode, the underlying physical machine must be VLAN clean meaning the network must provide Virtual LANs (VLANs) usable by Eucalyptus. In STATIC and SYSTEM modes, connectivity and metadata service were supported but the other networking features were not. In these modes, VM instances appear as physical machines on the physical network. [11]

For this study, the cloud must support IP control, security groups, elastic IPs and VM isolation. Since the physical network did not pass the ”VLAN Clean Test”, MANAGED-NOVLAN mode was used.

C. Load Balancing

The load balancer was implemented using four Java classes: MyServlet, Balancer, Server and Request. Apache 2 Web Server and Apache Tomcat 7 were installed to the Front End to house the load balancer.

MyServlet is an extension of the Java HttpServlet class and uses the doGet and doPost methods to receive incoming GET and POST HTTP requests. Balancer is in charge of the actual load balancing on running VMs in the cloud. It creates and terminates VM instances using euca2ools commands depending on the amount of incoming requests. A Server instance

is created and terminated with each VM instance and serves as Balancer's interface to the actual VMs in the cloud. A Request instance is created with each incoming request. It uses threads to monitor the status of the request paired with it and terminates itself once the said request has been served.

The load balancing scheme used was summarized in the following figure.

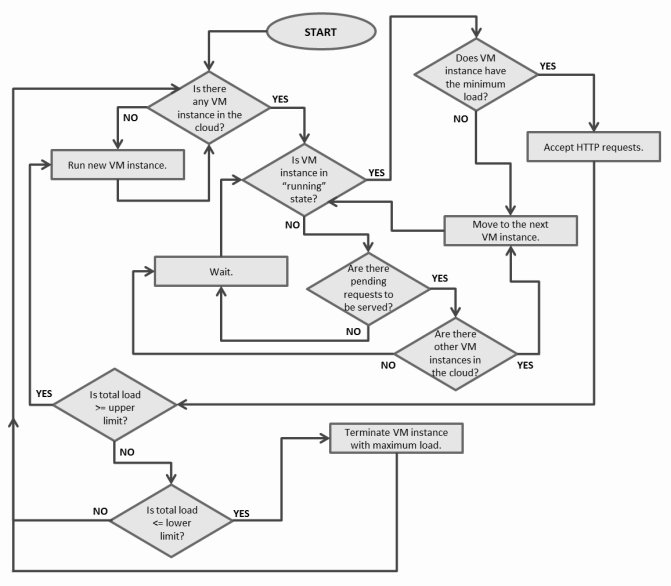

Fig. 2. Load Balancing Scheme Used for Eucalyptus LSS

Initially, a Balancer instance (Balancer) is created by MyServlet. Balancer will automatically check whether there are running VM instances in the cloud using the euca2ools command euca-describe-instances. If no VM instance is present, Balancer will call euca-run-instance to create a new VM instance in the cloud. MyServlet must wait for that VM instance to be in "running" state before it starts receiving requests. The first VM instance created will receive all incoming requests until the Balancer instance creates another one. A threshold (upper limit) is set by the administrator to determine when to create a new VM instance. Balancer checks the total load of the running VMs and compares it to the threshold. A new VM instance is created once the threshold is reached. On the same note, Balancer may terminate a VM instance if there is light traffic in the network. For this case, Balancer determines the VM instance with the heaviest load and marks it for termination. Balancer must wait for that VM instance to serve all its pending requests before terminating it using euca-terminate-instances. A threshold (lower limit) is also set by the administrator to determine when to terminate VM instances from the cloud.

D. Graphical Web Interface

The graphical web interface was developed using basic HTML, JQuery 1.5.1 and PHP 5. It was hosted on the Front End using Apache 2 Web Server. Basically, the graphical web interface is divided into two parts: the Monitor Tab and the Configurations Tab.

The Monitor Tab provides the administrator with real-time updates from the cloud. A PHP script executes the Eucalyptus command euca-describe-instances and parses its output to provide a visual representation of the status of each running VM instance in the cloud. Another PHP script fetches and displays information from the log files generated by the load balancer.

The Configurations Tab enables the administrator to change the value of the parameters used by the load balancer. These parameters are as follows: upper limit, lower limit, maximum capacity for each VM type and amount of resources to allocate to each request. It should be noted that these parameters are crucial to the load balancing process and changing their values would greatly affect the results of the operation. Data are stored in a configuration file in the Eucalyptus directory.

E. Testing

RequestSender was developed using Java to automatically send requests to the cloud. It has a graphical user interface (GUI) in which it takes URL as input. Once started, RequestSender floods that URL with HTTP requests and displays the responses from the receiving servers.

V. Results and Discussion

A. Cloud Setup

Two units of machine were setup in one of the Computer Laboratories of the Institute of Computer Science. Both were installed with CentOS 5.6 as their operating system. The Front-End was assigned with IP address 10.0.5.176, the Compute Node with 10.0.5.172. Eucalyptus and all its dependencies were successfully installed to both machines. Walrus and Euca2ools 1.3.1 were successfully installed to the Front End.

MANAGED-NOVLAN mode was used with the following configurations:

- VNET_MODE = "MANAGED-NOVLAN"
- VNET_SUBNET = "192.168.0.0"
- VNET_NETMASK = "255.255.0.0"
- VNET_DNS = "172.0.0.1"
- VNET_ADDRSPERNET = "32"
- VNET_PUBLICIPS = "10.0.5.10 - 10.0.5.14"

IP addresses 10.0.5.10, 10.0.5.11, 10.0.5.12, 10.0.5.13 and 10.0.5.14 were reserved to be dynamically allocated to the VM instances.

B. Virtual Machine

Customized VM images were created. One was installed with Apache 2 Web Server and PHP 5. UPLB iList was included with the image to serve as a test application. Another VM image was installed with Apache 2 Web Server, Apache Tomcat 7 and Java OpenJDK 1.6.0. Java servlets were included to serve as test applications.

The customized images were successfully uploaded to the cloud together with a kernel/ramdisk pair from Eucalyptus. VM instances were successfully created from the VM images using the kernel/ramdisk pair.

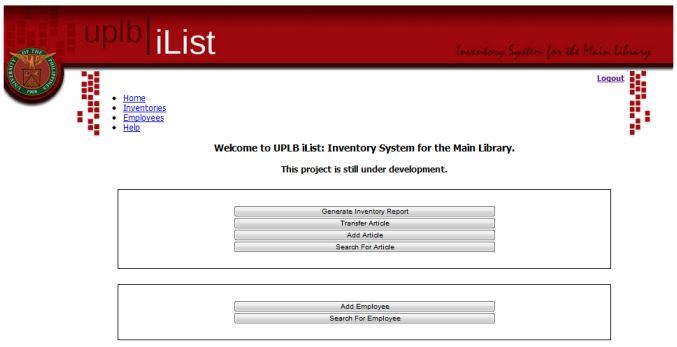

Fig. 3. UPLB iList Uploaded to the Cloud

Fig. 4. Eucalyptus LSS Test Page at 10.0.5.10

C. Load Balancing

The thresholds were set to 70% for the upper limit and 30% for the lower limit. Maximum capacity for m1.small VM type is set to 50%. The amount of resource to allocate to each request is set to 2%.

An initial VM instance was successfully created after the load balancer was started. It was assigned with public IP address 10.0.5.10. Requests to the cloud were successfully redirected to the VM instance.

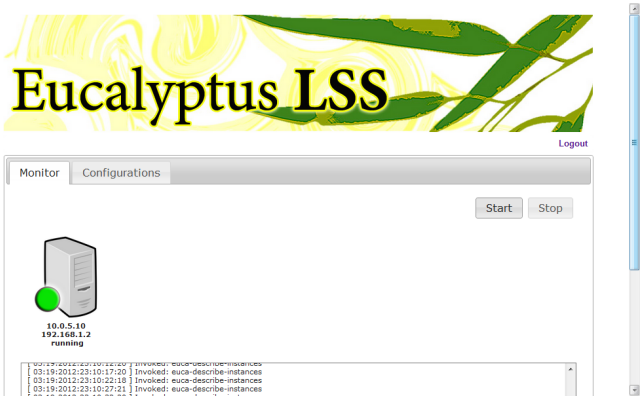

Fig. 5. Eucalyptus LSS with One Running VM Instance

The cloud was flooded with HTTP requests using Request-Sender and another VM instance was successfully created. It was assigned with public IP address 10.0.5.11. Requests to the cloud was successfully divided between the two VM instances.

After the flooding, one of the VM instances was terminated by the load balancer. The other one was to be kept running until the administrator stops the cloud services.

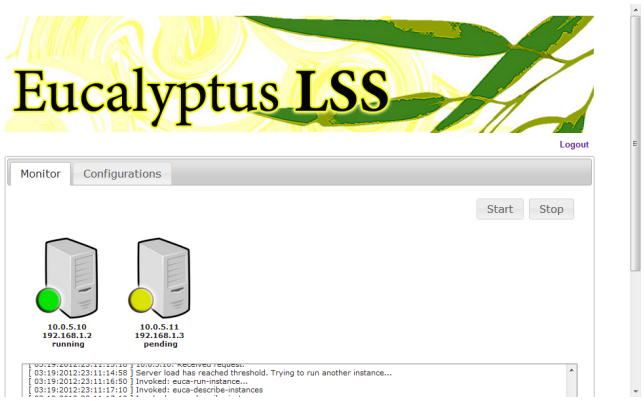

Fig. 6. Eucalyptus LSS with One Pending and One Running VM Instance

D. Log Files

Messages were sent by the load balancer during critical stages in its operation. These messages were successfully stored as log files in the Eucalyptus directory.

E. Graphical Web Interface

The graphical web interface was successfully developed and was successfully deployed to the Front End. A login page was provided to prevent non-administrators from viewing and/or modifying the status of the cloud. Login information is stored in an encrypted text file in the Eucalyptus directory.

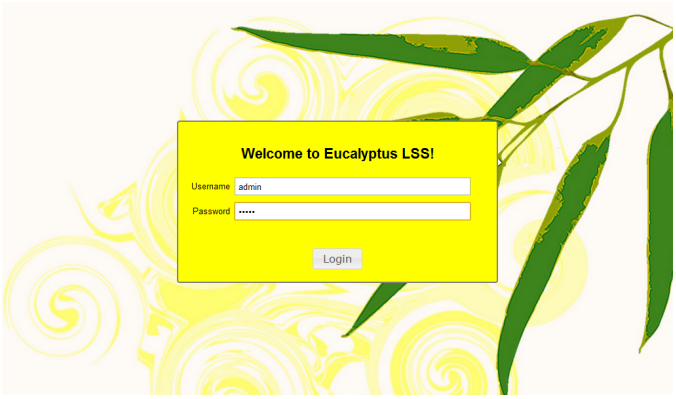

Fig. 7. Eucalyptus LSS Login Page

Status of each running VM instance in the cloud was successfully displayed in the Monitor Tab using the following convention: a green light in the server image indicates ”running” state, yellow light indicates ”pending” and red light indicates either ”shutting down” or ”terminated” state. Private and public IP addresses of each running VM instance were also displayed.

The Configurations Tab was successfully created. Changes in the values of the parameters were successfully stored in a configuration file in the Eucalyptus directory.

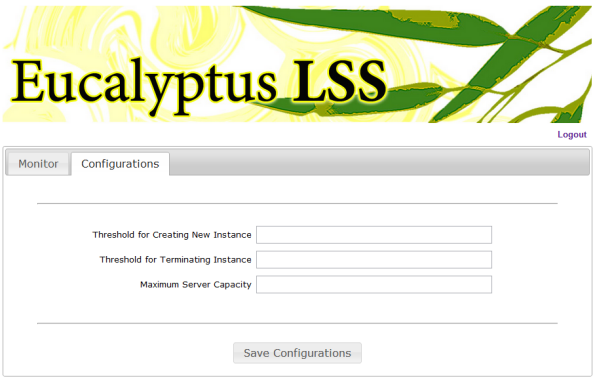

Fig. 8. Eucalyptus LSS Configurations Tab

VI. Conclusion and Future Work

Eucalyptus LSS is a load balancer designed to handle incoming HTTP requests and distribute them among VM instances in a private cloud setup. The load balancing scheme used took into account the capacity and the current load of each VM and the resources needed by the requests. In doing so, a better traffic handling is achieved. Security issues were also addressed in this study. Virtualization of services keeps intact the original data and thus they are safe even after a server failure.

Future development may improve Eucalyptus LSS by providing an Applications Tab in the graphical web interface to allow the administrator to view the applications supported by the cloud and define the amount of resources needed by each application. A functionality to change the administrator's password will also prove beneficial.

Acknowledgment

Shenlene A. Cabigting offers her sincerest gratitude to her adviser, Prof. Joseph Anthony C. Hermocilla, for guiding her throughout her Undergraduate Special Problem with his patience and knowledge; to the Institute of Computer Science, University of the Philippines Los Baños for providing her with two units of machine and permitting her to work in one of the PC Laboratories of the Institute; to her good friend Mr. Christian John M. Lo for his remarks and suggestions; to Mr. Pelayo and Mr. Jusay of the Institute for always assisting her while working in the PC Laboratory; to the Open Source community for providing her the tools that she used in this study; to her family and friends for the unwavering love and support; and lastly, to God for always giving her strength to accomplish all her endeavours.

References

[1] P. Mell and T. Grance. (2011) The nist definition of cloud computing. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwnski, G. Lee, Gunho, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. (2009) Above the clouds: A berkeley view of cloud computing. [Online]. Available: http://www.eecs.berkeley.edu/Pub/TechRpts/2009/EECS-2009-28.pdf

[3] Learn about cloud computing. [Online]. Available: http://open.eucalyptus.com/learn

[4] Extreme networks server load balancing. [Online]. Available: http://www.ibscentre.com.my/ibsweb/components/com_eform/views/form/tmpl/download.php?file_type=application/pdf&file_name=IBS_Training.pdf

[5] What is eucalyptus. [Online]. Available: http://open.eucalyptus.com/learn/what-is-eucalyptus

[6] R. Banacia and C. A. Belaguin, "Billing system: An accounting solution for the private cloud with eucalyptus," *University of the Philippines Los Baños Technical Report*.

[7] Scheduling policies 2.0. [Online]. Available: http://www.opennebula.org/documentation:archives:rel2.0:schg

[8] L. MacVittie. (2009) Load balancing is key to successful cloud-based (dynamic) architectures. [Online]. Available: http://devcentral.f5.com/weblogs/macvittie/archive/2009/01/23/load-balancing-is-key-to-successful-cloud-based-dynamic-architectures.aspx

[9] A. Desai. (2007) Virtual machine (vm). [Online]. Available: http://searchservervirtualization.techtarget.com/definition/virtual-machine

[10] (2010) Horizontal scaling. [Online]. Available: http://download.oracle.com/docs/cd/E195583-01/819-2582/addcd/index.html

[11] R. McCarty. (2009) Setting up cloud computing with eucalyptus. [Online]. Available: http://searchenterpriselinux.techtarget.com/tip/Setting-up-cloud-computing-with-Eucalyptus

Shenlene A. Cabigting is an undergraduate student under the BS Computer Science program of Institute of Computer Science, University of the Philippines Los Baños and a scholar of the Department of Science and Technology for the past four years. She is a proud member of the Young Software Engineers Society and was the organizations Liaison Officer for the Academic Year 2011-2012.