2012 4th International Conference on Computer Engineering and Technology (ICCET 2012)
IPCSIT vol.40 (2012) © (2012) IACSIT Press, Singapore

Introducing Low Cost Virtual Lab Environment for the University by Using Cloud Environment

Thimira Darshana Upatissa [1], Ajantha Atukorale [2]
[1] University of Colombo School of Computing, Sri Lanka
[2] University of Colombo School of Computing, Sri Lanka

Abstract. Cloud computing is a concept that comprises a lot of underlying technologies such as virtualization[1], utility computing and system oriented architecture. Eucalyptus is a Java based cloud management tool that consists with four major components: Cloud Controller, Cluster Controller, Node Controller and Walrus. The main objective of this research is to introduce a low cost virtual lab environment for an institute by keeping a consistent copy of Virtual Machine (VM) image when the multiple users access the same VM image. Basically there are lots of approaches which have been taken to improve the effectiveness of managing virtual machine images in Eucalyptus based Cloud environment. Here mainly we have been used two approaches to enhance the current virtual image sharing mechanism, without damaging the flow of the Cloud environment. In our approach we implemented a locking mechanism for the cloud controller which acts as the main control unit of the Cloud environment to keep the consistency and then we integrate the version enabling mechanism for the Walrus which acts as the main storage controller of the cloud environment. For the version enabling mechanism we used virtual machine images with Btrfs file system which contains the copy on write feature to keep the multiple copies of same VM image when required.

Keywords: Virtualization, Cloud Computing, Cloud Controller, Cluster Controller, Node Controller, Walrus, Virtual Lab , Eucalyptus

1. Introduction

For developing countries computer hardware / software resources are very important for their growth but prices keep them inaccessible. Thin Client lab environment provides them a greater opportunity specially for campus networks. It is important that this region can use flexible and low cost thin client lab environment for their growth. Our research work is to find a flexible and low cost lab environment based on cloud computing technologies.

Cloud environment is a layered architecture and cloud consists with the main layers. The bottom layer of the cloud computing environment is called Infrastructure as a Service in other words IaaS layer. Mainly the IaaS layer concerns about cloud infrastructure. For a better service the Cloud should consists of reliable hardware resources and those resources must be interconnected with each other by using virtualization techniques. The middle layer is the PaaS layer which means Platform as a Service. PaaS layer mainly concern on Platform of the cloud environment and to provide the cloud services to end users by establishing the cloud platform inside the cloud environment. The top layer is the SaaS layer that is Software as a service layer. End users can access the cloud environment and request for the services and cloud provides the particular service by using software that are stored in SaaS layer. To run the particular service the PaaS layer establishes the platform on top of the IaaS layer which provides infrastructure for the cloud environment. Currently there are very good cloud environments which provide effective, efficient and reliable services to the cloud users such as Amazon, Google and Microsoft clouds.

Most of the cloud providers support proprietary cloud environments and software. Without having an accessible and appropriate open source cloud environment for the research community it is difficult to

conduct researches on cloud environment. As a result some of the major questions in the cloud computing are still unanswered and open area for the research community. Cloud scheduling, multiple cloud merging, cloud security and resource sharing some of the areas that more researches is needed [2]. In order to fill this gap with proprietary systems, several open source projects have been implemented. Eucalyptus is an open source cloud management tool and which provides the public and private cloud environments. Virtual computing Lab (VCL) is another open source cloud environment which provides the cloud resources to the end users [3]. In our research we will be proposing our system based on Eucalyptus system.

1.1. Overview of Eucalyptus Cloud Environment

Eucalyptus consists with four major components. Cloud Controller, Cluster Controller, Node Controller and the Walrus. The node controller can manage the execution, inspection, and terminating of VM instances on the particular host where it runs. Node controller is responsible for the virtual machine instance. When the user requests for the particular virtual machine image the node controller would be able to start and run the particular virtual machine image and available the particular virtual machine instance in the network to make accessible for the end user. The cluster controller gathers the information and schedules virtual machine execution on specific node controllers, as well as manages virtual instance network that run inside the cloud environment. Walrus is a storage service that supports third party interfaces, providing a mechanism for storing and accessing virtual machine images and user data. The cloud controller is the entry point into the cloud for users and administrators. It queries node managers for information about resources, makes high level scheduling decisions, and implements them by making requests to cluster controller. Then Cluster Controller makes quarries by using Node Controllers to implement the Cloud Controller requests. The users interact with the Cloud Controller via the user interface by using SOAP or REST messages. Cloud Controller moves the user requests to the Cluster Controller. A Cloud Controller can have multiple cluster controllers and one particular cluster controller can have multiple node controllers.

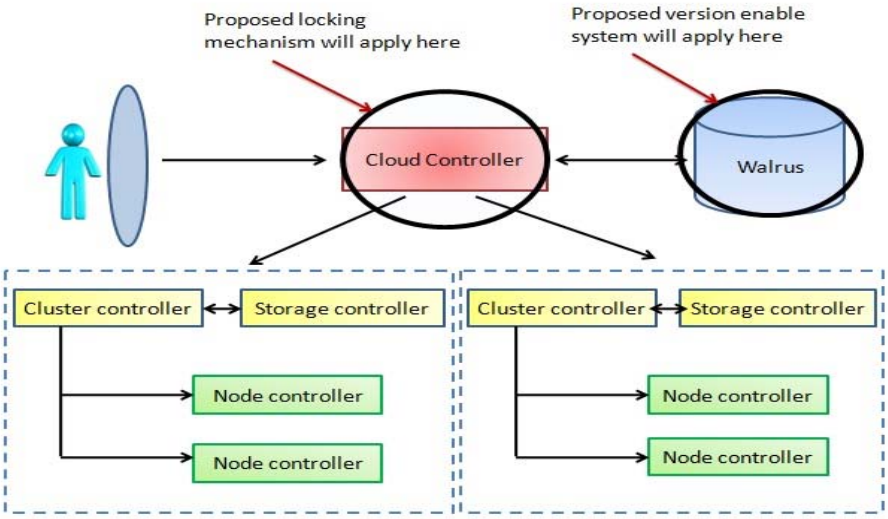

Fig. 1: Overview architecture of the proposed system

Walrus is a storage service included with Eucalyptus that supports standard web service technologies such as Axis 2, Mule etc. Walrus interface implements the REST and SOAP interfaces and is compatible with Amazon's Simple Storage Services (S3). Users employ the Walrus for image storage. Particular image contains the root file-system and guest operating system kernel code. Those two components are compulsory for the image file. In addition to that, optionally we can store the ram disk image with the relevant image file. Users can upload images to the Walrus by using standard tools such as EC2 tools provided by Amazon. As a First step of the uploading process tools compress the images and then encrypt image file by using user credentials. The next step encrypted image splits in to multiple chunks and keep the detailed about chunks image description file which is called manifest file. When the Node Controller requests for the image from Walrus, firstly the user is authenticated as a valid user by using user credentials that are stored in Cloud controller's canonical database user credentials and then images are verified and decrypted and transferred.

For performance optimization Walrus maintains a cache of images that have already decrypted according to the user requests. Walrus supports store persistent data, organized as buckets and objects. Major drawback of the Walrus is that does not provide locking for object writes. But walrus will provide a consistent copy of the object when the multiple users access the same object in the bucket and concurrent writes to the same object. If a write to an object is encountered while there is a previous write for the same object in progress, the previous write concern as an invalidated write [5]. Walrus supports MD5 checksum and when the user has been authenticated as a valid user, that user has permission to write and read for the objects over HTTP [4]. In this work we will discuss enabling a versioning and locking mechanisms in a Eucalyptus based cloud environment.

2. Implementation

In this section we discuss about the implementation process of the proposed architecture of cloud environment. The implementation process will be divided in to two parts. For the first part we implemented the locking mechanism and for the second part we implemented the version enabling mechanism. Before integrating the two parts we implemented the Eucalyptus based cloud environment with the Xen Hypervisor [7]. After the implementation of cloud environment and integrated parts we accessed the cloud by using euca2-tools and then managed the cloud environment. By using Eucalyptus user interface also we can manage the cloud resources and services. Here we have used both approaches to manage and accessed the cloud environment.

2.1. Eucalyptus based cloud creation with Xen hypervisor

For the cloud creation we used a server machine with a RAM of 16GB with Debian 5.0.7 operating system. Because of resource constraints we deployed cloud controller, cluster controller, node controller and Walrus in the same machine on top of a Xen hypervisor.

- Eucalyptus installation: We have been used Xen 3.0 as the hypervisor for the cloud environment for this system and Eucalyptus 1.6 source package for the Eucalyptus installation. Before we install the Eucalyptus source package we had to install dependencies. Mainly C compiler, Java Developer Kit (SDK) version 1.6 or above, Apache ANT 1.6.5 and Axis2C and rampart development files packages are required for the Eucalyptus installation. After installing required packages we installed Eucalyptus and register the cloud components.
- Euca2ools installation: For the cloud management we used Euca2ools which supports for the Eucalyptus based cloud environment
- Integration of locking mechanism: We added new classes for the Eucalyptus source code under Cloud controller package and recompiled and run the updated code. To edit and rebuild the Eucalyptus source code we used eclipse IDE and ANT as a building tool. The added lock is basically mutual lock which prevents the multiple writes at the same time.
- Version enabling mechanism: Though we discussed two main approaches for the version enabling mechanism, we implemented only one approach due to time constraints that is by enabling Copy On Write (COW) feature in VM image by using Btrfs file system. To create the VM images with Btrfs file system we used Btrfs-tools [8].
- Integration of the system: After integration of the locking mechanism once again we deployed the cloud environment with new changes. To run the VM images in cloud environment we had to bind, upload and register them. VM images were bound by using Euca2ools. We uploaded two types of VM images with ext3 file system and Btrfs file system to gather profiling details and check the difference between both.

3. Evaluation

For the evaluation we gathered profiling data with our new system under initrd.img-2.6.26-2-xen-amd64 kernel code with vmlinux-2.6.26-2-xen-amd64 ramdisk image. We started 10 Debian5.0-x86_amd64 VM image instances under ext3 filesystem with 600 seconds time difference by using our cloud environment and gathered profiling data. The 600 seconds time difference is taken because the systat process writes VM's

profiling data every 600 seconds. This time difference parameters can be adjusted according to user requirements and we found empirically the 600 seconds difference is ideal for our experiment. In order to compare the profiling data collected according to the above setup we started another 10 Dapper VM instances with Btrfs file system and obtained the same profiling data as in the previous setup.

To get the profiling data we used systat and proc details of the cloud environment and filtered them. Using the filtered data we plot the number of processes in the proc per second when the DOM 0 running in the cloud environment with other guest operating systems under ext3 file system and Btrfs file system .

According to that the number of processes in the proc increased with the number of instances. When considered the incremental levels, ext3 increment is higher than the Btrfs. This means to run ext3 VM instances needed more CPU power when compare to the Btrfs VM instances. We found that an interesting observation with regard to the Btrfs VM image instances. In this cloud environment it always kept the total number of up and running instances to a constant number. Under this environment when a request is made. For a new instance, one of the currently running instances is automatically shut down and new instance is started. Because of this observation the comparison we did in Figure 4 is not fair. To overcome that issue we again did the profiling comparison by considering the processes per instance to make the comparison uniform and fair among the two file system.

Because of that we took the number of newly created processes in the proc per instance in both ext3 and Btrfs VM images when they were running. To find the results for a newly added process per instance we used the following equations and profiling details. First we calculated the mean number of created processes in the proc when the time of DOM 0 is up and running. For the next step we derived the number of context switches occurred in the cloud environment. By observing the results we realised that the number of context switches occurred in the cloud environment increased with the number of instances that were running in the cloud environment.

At the beginning number of context switches occurred with ext3 VM image took lower value when compared to the Btrfs VM image. But after a few instances were running, ext3 VM image context switches increased more than Btrfs VM image. In both situation number of context switches increased gradually when we increased the number of running instances in the cloud environment. Basically cloud environment consists of the some underlying hardware resources and network structure. When the cloud environment started, it established its own virtual LAN environment and run VM images by using cloud resources. Because of that we managed to measure the total number of transfers per second occurred in cloud environment. A transfer means I/O request to a physical device.

4. Conclusion

Contributions to the congress are welcome from throughout the world. When consider about the creation of VM image copy we realized that, only using locking mechanism and Btrfs file system we couldn't generate the VM image copy. Because when the time of VM instance is running there were lots of dependencies in the cloud environment. Hardware, network and hypervisor were main dependencies in the cloud environment. Such snapshots can be used by several users to carry on their work without dependencies on other users.

When consider about the challenges of the system scalability problem was the main challenge that we had to overcome in this research. It is really difficult to create multiple copies of same VM image without using additional storage capacity. In order to make our proposed system scalable we need to address the challenge of additional storage for the copies of VM images.

5. Future Works

As future works of this research there is a need of new file system which can keep the updates of the VM images and when in such a system a user wants to up and run a VM image, the file system should quickly map the updates for the VM image and provide the consistent copy of that VM image. If such a file system in Walrus, we can address the scalability problem in cloud environment when multiple copies are generated.

In our experiment we have used Xen hypervisor to establish the cloud environment. If future work can follow same approach with kvm hypervisor to gather profiling data and could compare both hypervisors and visualize the clear understanding regarding the effectiveness and flexibility of our research.

6. Acknowledgements

The authors would like to thank Network Operation Centre (NOC) of the University of Colombo School Of Computing (UCSC) for required hardware resources and also the staff who helped in various ways to make this research a success. .

7. References

[1] G. J. Popek and R. P. Goldberg, Formal requirements for virtualizable third generation architectures, Communications of the ACM, vol. 17, no. 7, pp. 412-421, 1974.

[2] J. Shafer, I/O virtualization bottlenecks in cloud computing today, *Proceedings of the 2nd conference on I/O virtualization*, 2010.

[3] H. Scha_er, S. Averitt, M. Hoit, A. Peeler, E. D. Sills, and M. Vouk, NCSU's Virtual Computing Lab: A Cloud Computing Solution, pp. 94{97, 2009.

[4] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youse_,*and D. Zagorodnov, The Eucalyptus Open-Source Cloud-Computing System,"IEEE Communications Magazine, 2009.*

[5] M. A. Vouk, Cloud Computing Issues,Research and Implementations, *Journal of Computing and Information Technology*, pp. 235-246, 2007

[6] (2010)IBM website [Online].Available:http://www.ibm.com/developerworks/cloud/library/cl- modblockstore/?S_CMP=dw&S_CPD=cld&S_CT=dwcon&S_CR=devx&S_CCY=xx /

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer,I. Pratt, and A. War_eld, Xen and the art of virtualization, vol. 37, no. 5, pp. 164-177, 2003.

(2010)Virttopia web site[Online].Available: http://www.virtuatopia.com/index.php/Building_a_Xen_Virtual_Guest_Filesystem_on_a_Disk_Image_%28Cloning_Host_System%29