Design and Building of a Scalable VM Cloud System

P. Jatesiktat, N. Sornchumni, Y. Yuttakonkit and P. Uthayopas[C]
HPCNC, Department of Computer Engineering, Faculty of Engineering, Kasetsart University
50 Phaholyothin Rd., Jatujak, Bangkok, 10900, Thailand
[C] **E-mail**: pu@ku.ac.th

ABSTRACT

The need for flexible and highly reliable computing resources has led to the growth of cloud computing system deployment in many organizations. Anyway, a cost-effective software solution for a cloud system is still needed. Thus, an open source implementation of a cloud solution is one of a very attractive solution for the organization need. In this paper, the design and building of a scalable open source VM cloud system called *Maekin* is presented. The design of Maekin system focuses on a simple installation using network installation and the use of automatic workload management to enhance the performance. A sophisticated cloud middleware has been developed that allow a collection of servers to interoperate and form a scalable cloud platform. A simple system management is provided through web-based interface. The experimental application of Maekin illustrates the simplicity and effective power consumption as expected.

Keywords: Cloud Computing, Virtual Machine, Virtualization, Open Source Cloud.

1. INTRODUCTION

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). These resources can be provisioned rapidly and released with minimal management effort or service provider interaction [1]. An organization can use infrastructure layer service to serve the need for scalable and inexpensive on-demand computing infrastructure and reduce the operation cost.[2] Organizations that need to converge computing resources can employ the cloud system to integrate with existing solutions and maintain the compliance with the industry standards.[3] Many service-oriented businesses has been emerged with cloud computing. These include the company like Amazon and Microsoft that focus on providing online resources for the main stream businesses. In addition, many cloud applications such as Google App and Dropbox provide an unlimited service to users. Moreover, Social network such as facebook take advantage of cloud computing technology to connect people around the world. Although many cloud software solutions are available such as VMWare, OpenNebula, Openstack, and Eucalyptus. There is still a need for simple, usable, open source cloud software system that users can use to build a cost effective cloud computing system.

This work presents the design and building of a VM cloud system called Maekin system. This system is a simple open source cloud computing software stack that offer an easy to install, easy to operate solution for users. The rest of this paper is organized as follows. First, the related work is present in Section 2. Then, the design of Maekin system is explain in Section 3. Section 4 presents the experimental evaluation and discussion. Finally Section 5 gives the conclusion.

2. RELATED WORKS

Most of the cloud software solution is built using the virtualization technology. This is usually provided by a hypervisor system such as KVM [4] and Xen [5]. These hypervisors enable new ways to provide resources to users, transparent operating system hosting, and complete separation of multiple user environments. As a result, the system resource provisioning is faster and more flexible. By coupling this virtualization technology with cluster and grid technology, one can built a scalable and dynamic computing system called cloud computing system. Cloud computing is a sibling to Grid computing [6], Cloud computing focus on scaling to support large numbers of users using virtualized centralized system while Grid computing focus on harvesting geographically distributed resources across many domains for user application.

Currently, there are many existing software user can use to build a cloud computing system. Eucalyptus [7] is one of them. Eucalyptus designs can be divided into four top-level components; Node controller, Cluster controller, Storage controller, Cloud controller. The design clearly evolves from a cluster computing system. OpenStack [8], another broadly used cloud software, consists of these main components [9]; API server, Cloud controller, Scheduler, Network controller, Volume controller, Auth manager, and Compute controller. These simple, flexible and modular with hierarchical design reflecting common resource environments found in many academics settings. The design goal is to make the system fault tolerance while easy to manage. A good comparison is given in [9]. OpenStack main focus is on scalability while Eucalyptus will focus more on extensibility and quick installation.

3. DESIGN OF MAEKIN CLOUD SYSTEM

In order to design an IaaS (Infrastructure as a Service) Cloud system, one has to understand the underlining requirement to properly set the design goal. For this work, the design goal is to build a software that is easy to installed by users, allow the dynamic change in workload to be properly handled by the system, and provides an easy-to-use user interface for user and system administrator.

After the design goal as mentioned earlier was set, the technology aspect of the project is considered. In principle, an IaaS cloud is a cluster of computer servers that is managed together as a collective. Each machine must support some form of virtualization. In this work, an open source virtualization control library called *libvirt* is used to control the virtualization subsystem on each machine. The benefit of this approach is to abstract the virtualization layer out of the design so Maekin software can be easily ported and run on various virtualization systems such as Zen, KVM, or VMware. Thus, there are three separate parts that need to be developed. First, the Cloud Distribution must be developed to facilitate the installation and software packaging process. Then, the Cloud Middleware must be developed to present a collective management view of the cloud to users and administrator. The cloud middleware must also expose a set of API to enable a value added third party development. Finally, a cloud management tool must be developed to allow system administrator and user to access and control the cloud system. In the following sections, the design and structure of each major component will be described in more detail.

3.1 The Design of Cloud System Installer

To make system installation easy, the installation method that are selected is divided into two steps. First, one node in the cloud is selected as a master cloud control node. Then, Maekin system is installed using customized anaconda and kickstart method. The component of Maekin system such as middleware, cluster management tool and development library is added to standard CentOS distribution by modifying the anaconda and kickstart script. After the master cloud node was installed, the rest of the cloud node can be installed by two methods. One way is to use Maekin distribution CD rom to install each node in the system. This is a good approach for small cloud system. For the last cloud system, Maekin provide a network kickstart installation that enables a new machine to be booted and install remotely from the master node. Once master node and the rest of cloud node are booted, the middleware will be activated and connect together automatically to form the cloud collective.

3.2 Maekin Cloud Middleware

Maekin cloud middleware is the heart of Maekin system. Currently, Maekin cloud middleware uses Kernel-based Virtual Machine (KVM) hypervisor to provide a virtualization on each machine. Global share storage must be provided to store the VM images so they can be accessed and shared among the cloud nodes. The internal network on each cloud node employs the bridge connection to share network resources among local operating system and guest VM. Network File System (NFS) is used to share virtual machine template files and virtual machine image files among cloud nodes in the system. The main reason for using shared storage is to enable smooth live migration feature. The organization of Maekin cloud middleware is as shown in Figure 1.

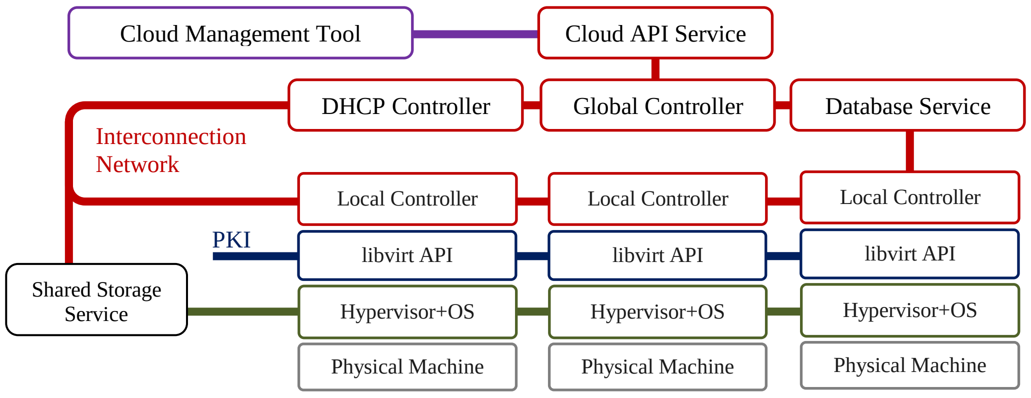

Figure 1. The Architecture of Maekin Cloud Middleware

Maekin middleware composed of 4 main components; Local controller, Global Controller, Database Service, and DHCP controller. These distributed components communicate using socket service over TCP/IP network inside the cloud. The functionality of each component is described as follows.

3.2.1 Local Controller

Local controller is an agent installed on each cloud node. Local controller waits for command from other services and perform the indicated action such as start stop the VM, report system status, and many others. Another function of local controller is to monitor the cloud node and guest VMs on each node. The information monitored such as CPU, I/O and memory usage statistics is logged in a database. This information will be used in virtual machine migration planning and management policy.

The libvirt virtualization API is an important tool used by local controller to manage virtual machine life cycle. The libvirt API in each cloud node will be connected together with public key infrastructure (PKI) which had been setup at start up time of the cloud system. Common PKI allows libvirt on each machine to connect to each other as a collective. Therefore, virtual machine migration can be done at the layer of libvirt API.

3.2.2 Global Controller

Global Controller plays an important role in the cloud system. First, global controller waits for API message from external software, pass the request to appropriate module and collecting the response and result back to caller. Second, this component handles the task queue and dispatches the task to other modules inside the cloud middleware. Finally, global controller controls the resource management policy such as when and how to migrate virtual machines, when to power off or boot up a cloud node dynamically. The decision is based on the data collected from local host and monitoring subsystem.

3.2.3 Database Service

During the operation, there is much information that must be collected. Thus, Maekin cloud middleware provide a database service for internal and external use. This service is based on MySQL Database. The data collected include general cloud information (cloud variables, IP address pool, template, and host, guest), task queue information, and monitoring log (from both cloud node and guest VM). Every module inside Maekin Cloud middleware can access this centralized database. In this work, database service can be moved to any cloud node in cloud system as needed to provide the reliability and flexibility.

3.2.4 DHCP Controller

A DHCP controller role is to control the binding of IP address with MAC address of each physical and virtual machine. This is important since it to make all machines accessible from internet. This component must be installed and operated on the same cloud node as the global controller.

3.2.5 Host Monitoring System

Monitoring System consists of 2 daemons; *Host Monitor Module* and *Host Information Service*. Host Monitor Module collects local host information such as CPUs load, amount of memory, storage space, network transfer rate, etc. This information is broadcasted to other hosts using IP multicasting. The Host Information Service acts as cloud node information service and provides system information that middleware and tool needed to function. Information service data can be retrieved from any host by contacting the local Host Information Service daemon.

3.2.6 API Service

Finally, Maekin cloud middleware has also support a service that offers API for the developers. This service allows any management tool to access modules under control of middleware system. The API request is in the form of Representational State Transfer (REST) Web service. So, Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML) are used to communicate between services and management tools. Thanks to multithreading manner of CherryPy web server tool, our API can serve many transactions at a time. One of the very important API clients is a utility called *Maekinsh* or Maekin shell that provide a command line interface that user can use to quickly manage Maekin system.

3.3 Cloud Management Tool Design

The management of Maekin system is based on the concept called *Virtual Infrastructure (VI)*. VI is a set of users, virtual machine templates, and virtual machines that is grouped as a unit. An example of VI is one company that can have multiple VM servers for many tasks in the company such as accounting, payroll, web, mail, and more. Using VI concept, administrator can provide resources such as IP pool, virtual CPU, and virtual machine template to the owner of VI so they can freely manage the VI under their authorization. This is different from most of the tools that do not provide the clear concept of VM group. VI owner can invite another user to collaborate with VI using role to control the permission of each user. By proposing this concept and enforce it in middleware and management tool, better privacy and control of a cloud computing system in organization can be achieved.

To build the cloud management tool, many open source technology has been employed. The cloud management tool is a web based application powered by Django python web framework that is designed to enable a rapid web development. Cloud Management Tool stores information of virtual machine and cloud node from middleware into its own internal database. This technique enables users to quickly access the cloud system status. In addition, the tool also manages the data that represent user and VI configuration as well. The organization of Maekin cluster management tool is as shown in Figure 2(a).

The user interface design can be developed quickly using Django template module. This part can be customized using a Cascading Style Sheets (CSS) to make an easy-to-use interface. A quick user interaction is needed to enhance the users' experience. Many open source library and framework is employed to speed up the development. For example, jQuery, a javascript library, combine with another plugin library such as jQueryUI is used to enable drag-drop objects, jqplot is used to draw a performance graph, and Asynchronous JavaScript (Ajax) library is used to make this interface look and feel similar to a standalone application.

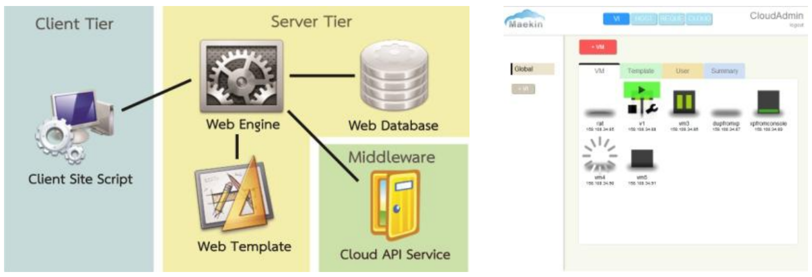

Figure 2. (a) Architecture of Cloud Management Tool (b) User Interface

Management tool communicates with the cloud middleware using Maekin cloud middleware API which is invoked using REST protocol. This design enables a decoupling of user interface component from middleware and open up many possibility such as building the management software on Tablet PC, smart phone in a cross-platform way.

4. EXPERIMENTAL EVALUATION

One of the key issues in operating a cloud is to ensure that the middleware incurs a low overhead and scale very well. In this section, the result of the experiments conducted to test the cloud system is provided.

In this work, the machine used is Dell Optiplex 760, Intel® Core™ 2 Duo E8400 3.00GHz, 1.333 GHz Bus, 2 GB DDR2 800 MHz, 230 GB HDD, and Gigabit Ethernet. These 5 machines were connected together with a Gigabit Ethernet switch and had been installed with Maekin system. For the experiment, the size of Maekin system is varied from 1 cloud node to 5 cloud nodes. In each case, cloud node no.1 is selected to handle all centralized service in the system such as Cloud API Service, Database Service, Shared Storage Service, DHCP Controller and Global Controller. A minimal CentOS 6.2, with 256 MB of RAM and 1 VCPU was installed on deployed virtual machine. This virtual machine was set to consume smallest network bandwidth as possible. So, we could measure the real network bandwidth consumption of our system. In each case, 5 identical virtual machines were deployed in the cloud by manually distribute these virtual machines as seen in Table 1. After the cloud system is booted up, an average receive rate and transmit rate at each booted cloud node is measured for 10 minutes.

Table 1. Distribution of virtual machines in each case of experiment.

	Host #1	**Host #2**	**Host #3**	**Host #4**	**Host #5**
Case #1	host 5 VM	shut off	shut off	shut off	shut off
Case #2	host 1 VM	host 4 VM	shut off	shut off	shut off
Case #3	host 1 VM	host 2 VM	host 2 VM	shut off	shut off
Case #4	host 1 VM	host 1 VM	host 1 VM	host 2 VM	shut off
Case #5	host 1 VM	host 1 VM	host 1 VM	host 1 VM	host 1 VM

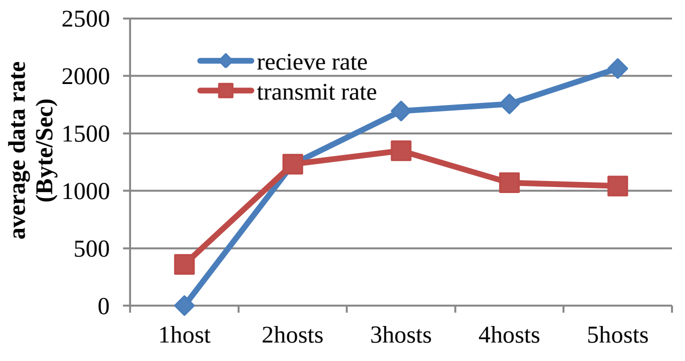

Figure 3. Average data receive and transmit rate per host.

The results show that the overhead of our system when scaling from 2 to 5 cloud nodes is about 1 to 2 kilobytes per second per host. This value is very small compared to a whole capacity of interconnection network inside the cloud. From the experiments, there are a few observations.

First, most bandwidth usage came from cloud node monitoring system that communicate among all cloud nodes frequently. Second, a little bandwidth usage came from database connection which is used to log monitoring status of both cloud nodes and virtual machines. Finally, some bandwidth usage came from IO usage of each virtual machine via Network File System (NFS). In this experiment, IO usage of virtual machine was minimized by running only operating system on those virtual machines. From this result, we can estimate how well Maekin cloud system can scale. Average data receive rate per cloud node grows as linear trend and increases about 500 byte per second for each one cloud node adding. So, this system can be scale to at least 32 cloud nodes with network bandwidth overhead at about 16 kilobyte per second which is only about 0.002% of Gigabit network capacity.

5. CONCLUSION

In this paper, the design of an open source VM cloud called Maekin cloud system is presented. Our proposed design provides a cloud system with user-friendly interface, scalability and flexibility. Moreover, it can be installed easily to any group of machines. The implementation is based on various open source technology such as KVM, Libvirt, and python language. This system is very useful as a research tools to explore a new cloud concept. In addition, the implementation can be adopted to build a small to medium private VM cloud in a very cost effective way. There are many issues that can be explored in the future. A dynamic VM migration for load balancing is one of useful things to be developed. More innovative cloud management model on top of the Virtual Infrastructure can be explorer. The dynamic inter-cloud and personal cloud environment and interaction is also a key area that needs more investigation.

REFERENCES

1. "The NIST Definition of Cloud Computing". National Institute of Science and Technology. Retrieved 24 July 2011.
2. Victor Chang, Gary Wills, David De Roure, "A Review of Cloud Business Models and Sustainability," cloud, pp.43-50, 2010 IEEE 3rd International Conference on Cloud Computing, 2010
3. Yu Chen Zhou, Xin Peng Liu, Xi Ning Wang, Liang Xue, Xiao Xing Liang, Shuang Liang, "Business Process Centric Platform-as-a-Service Model and Technologies for Cloud Enabled Industry Solutions," cloud, pp.534-537, 2010 IEEE 3rd International Conference on Cloud Computing, 2010
4. Haydn, S., *KVM* [Online]. Available: http://www.linux-kvm.org/
5. Paul, B., Boris, D., Keir, F., Steven, H., Tim, H., Alex, H., Rolf, N., Ian, P., Andrew, W., *Xen and the art of virtualization*, Proceedings of the nineteenth ACM symposium on Operating systems principles, New York, 2003.
6. Nurmi et al., *The Eucalyptus Open-source Cloud-computing System*, Cloud Computing and Applications 2008 (CCA 08), 2008.
7. Eucalyptus Systems, *Eucalyptus* [Online]. Available: http://go.eucalyptus.com/rs/eucalyptus/images/Blending_Clouds_Blueprint_for_Hybrid_Future_White_Paper.pdf
8. Rackspace Cloud Computing, *OpenStack Compute* [Online]. Available: http://openstack.org/downloads/openstack-overview-datasheet.pdf
9. Sumayah, A., *Behind the scenes of IaaS implementations*, 2011.