東海大學

資訊工程研究所

碩士論文

指導教授：楊朝棟 博士

整合 KVM 和 OpenNebula 建構出動態遷移的雲端虛擬化環境

On Construction of Cloud Virtualization for Live Migration by Integration of KVM and OpenNebula

研究生：王紹豐

中華民國 一Ｏ一年七月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 王 紹 豐 所提之論文

整合KVM和OpenNebula建構出動態遷移的雲端虛擬化環境

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 時文中

賴冠州

朱正忠

指 導 教 授 楊朝棟 簽章

中華民國 101 年 7 月 5 日

摘要

雲端技術的應用和服務與日俱增，不管是政府或是企業、組織甚至於個人都可能有建構雲端系統的需求。本文著重於利用開放原始碼的技術，讓一般使用者在無需花費高額的軟體授權費用下，也能擁有私有雲的服務。在電腦科學中，虛擬化指的是邏輯群組的呈現或是電腦資源的部分集合。本文中所提到虛擬化所指的是平台虛擬化，也就是一般人所說的虛擬機。虛擬化的好處不勝枚舉，通常也是企業進入雲端前所必須學習的步驟。本文所建構的環境也實做了虛擬化並加以實驗。本研究主題是如何在雲端建立虛擬環境並運行，以及整合 KVM 和 OpenNebula 等開放源始碼軟體，為用戶端提供雲端虛擬化環境。它成功地提供了企業或組織的私有雲解決方案。其重點在於雲端三種服務模式的基礎建設服務。就使用者界面的部分，此系統可以減少使用者存取雲端資源的複雜度。就使用者存取的操作上，所建構出的網頁介面來管理虛擬機的遷移，是極為容易的。本文以虛擬機動態遷移與針對實體機器和虛擬機的性能等實驗項目，來加以實測，並根據數據分析其結果。在實驗環境中我們證明了動態遷移並非如商業廣告所說的不停機轉移。此外我們也得到了全虛擬化技術相當貼近實機效能的結果。最終完整的建構出一個開放原始碼的私有雲解決方案，並且也實做了雲端三大服務的結合。

關鍵字：雲端計算、基礎建設即服務、基於核心的虛擬機器、即時遷移、虛擬機供應

Abstract

Cloud computing services have been increasing that no matter government, organizations and even individuals are likely to construct the cloud system themselves. This thesis focuses on cloud technology in open-source to achieve the private cloud for ordinary people without high software license charge. Virtualization is a process of manifestation of the logical group or subset of computer resources in computer science. However, virtualization we mentioned in this thesis is "platform virtualization". It is VM (virtualization machine) as we often named. Benefits of virtualization are numerous and it is considered to be a previous procedure to learn before adapting cloud technology for the enterprise. The constructed environment in this thesis has implemented virtualization for further experiment. The main subject of this thesis is how to construct virtualization in the cloud with integration of KVM and OpenNebula for users. It provides private cloud solutions for enterprises or organizations and focuses on IaaS of three services in cloud. This system can reduce the complexity of accessing the cloud resources through users' interface. That is to say it is easy to mange deployment of the VMs by the web-based users' interface. The thesis contains of live migration data mesurement, comparison of physical machines and virtual machines, and analysis results. In the experimental environment, we prove that live migration is not a non-stop service and we prove that the performance of full virtualization is closer to the physical machine as well. At last, we have completed construction of the open source solution for private cloud and have combined three cloud services.

Keywords: Cloud Computing, IaaS, KVM, OpenNebula, Live migration, VM provision

Acknowledgements

I would like to express my gratitude to all those who helped me during the writing of this thesis. I gratefully acknowledge the help of my supervisor, Dr. Chao-Tung Yang, who has offered me valuable suggestions in the academic studies. Thanks for giving me the opportunity to be a member of the HPC Lab in past two years. In the preparation of the thesis, he has spent much time reading through each draft and provided me with inspiring advice. Without his encouragement, insightful criticism and expert guidance, the completion of this thesis would not have been possible.

And I feel so sorry to my family. I owe a responsible husband and father to my family for two years. I guessed it is time to return them. I should finally like to express my gratitude to my beloved parents who have always been. My mother always supported my direction which I decided. My father is the alumnus of Tunghai University. He wanted to go to Institute but no chance to achieve.

Now his son is achieved!

Table of Contents

摘要 iii

Abstract iv

Acknowledgements v

Table of Contents vi

List of Figures viii

List of Tables x

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Thesis Contributions 2

1.3 Thesis Organization 2

Chapter 2 Background Review 3

2.1 Cloud Computing 3

2.2 Virtualization 4

2.3 Open Source for Virtualization 7

2.4 Virtualization Management 9

2.5 Live Migration 11

2.6 Related Work 13

Chapter 3 System Implementation 15

3.1 System Overview 15

3.2 System Architecture 16

3.3 Live Migration Mechanism of VM 19

3.4 Applications (MIFAS) 20

Chapter 4 Experimental Results 23

4.1 Experimental Environment 24

4.2 Results 24

4.3 Discussions 35

Chapter 5 Conclusions and Future Work 37

5.1 Conclusions 37

5.2 Future Work 37

Bibliography 39

Appendix – Installation Guide 43

List of Figures

Figure 2-1. The architecture of full virtualization 5
Figure 2-2. The architecture of para-virtualization 6
Figure 2-3. The H/W Enabled 8
Figure 2-4.The S/W (OS) Supported 8
Figure 2-5. The diagram of OpenNebula 10
Figure 2-6. The concept of Live Migration 12
Figure 2-7. The successful process of Live Migration 13
Figure 3-1. the domain of IaaS 15
Figure 3-2. The Concept of OpenNebula 16
Figure 3-3. System architecture 18
Figure 3-4. Initial presentation of OpenNebula 18
Figure 3-5. Live migration of the VM from Node3 to Node2 18
Figure 3-6. Submitted 19
Figure 3-7. Live migrated Success 19
Figure 3-8. Live migration Time Line 20
Figure 3-9. MIFAS login interface 20
Figure 3-10. The System Components of MIFAS 21
Figure 3-11. Quick view of medical images 22
Figure 3-12. The view of medical image. 22
Figure 4-1. The initiation of live migration 25
Figure 4-2. The ping of live migration 25
Figure 4-3. The end of live migration 25
Figure 4-4. The ping during live migration 26
Figure 4-5. The ping result after live migration finish 26
Figure 4-6. Collect ping reply time 27
Figure 4-7. Time of live migration 28
Figure 4-8. The performance results of HPCC by one CPU in different platform. 29
Figure 4-9. The computation results of HPCC by one CPU is in different platforms. 30
Figure 4-10. The performance results of HPCC by two CPUs in different platform. 31
Figure 4-11. The computation results of HPCC by two CPUs in different platform. 31
Figure 4-12. Download a file from Internet 32
Figure 4-13. Download a file from Internet with live migration 32
Figure 4-14. Live migration is success 33
Figure 4-15. When live migration beginning, the transfer rate is down. 33
Figure 4-16. With live migration, it takes more time to download. 33
Figure 4-17. Download a file from Intranet(CentOS-6.2-i386-LiveCD.iso) 33

Figure 4-18. Download a file from Intranet with live migration 34
Figure 4-19. Live migration is success.. 34
Figure 4-20. When live migration beginning, the transfer rate is down. 34
Figure 4-21. With live migration, it take more time to download.. 34
Figure 4-22. The compare between WAN(Internet) and LAN(Intranet)................................. 35

List of Tables

Table 4-1. Environment Specification........ 24
Table 4-2. Environment of Experiment 3 Specification........ 28
Table 4-3. Environment Specification........ 32
Table 4-4. The two situations results........ 34

Chapter 1

Introduction

1.1 Motivation

Cloud computing has been highly important topic in recent IT world, and various services are therfore developed. Cloud computing may not be considered as a new technology but a new concept [1, 10, 19, 23, 21, 22, 24, 28, 30]. The early stages of the laboratory started in the creation and development of grid computing cluster and other distributed computing technologies and related issues, for the vigorous development in recent years is also very interested in cloud computing [13, 15]. There are many companies currently offer a cloud of related services, like Google [30], Amazon [29], Yahoo and other companies. Thousands of servers are used to construct a large-scale computing resource, and provides a variety of services such as: large storage space and a huge amount of computing power. There is no need to download the online features such as edit view because of limited local computing and storage resources. Users can access via the Internet computing resources to obtain what they need.

This thesis focuses on the cloud computing infrastructure, particularly virtual machines and live migration [3, 4, 5, 6, 7, 8, 9, 11, 12, 14, 18, 33, 34]. According to the NIST's proposal, the basic three models are SaaS, Paas, and IaaS. In view of the varied capability of IaaS, we decide to construct a platform by open source and implement the three service models combination of cloud.

1.2 Thesis Contributions

This thesis focuses on cloud computing infrastructure especially virtualization and live migration. The goal is to build a system which belongs to the private cloud, so management and deployment with VM is a major mission. The information supplied by the system can be monitored including CPU utilization, disk usage, virtual machine space, and memory usage. This system can also take live migration. When a problem occurs, the administrator can shift the user's virtual machine to another physical machine so that the user will not feel any abnormalities. It is an important and useful function for virtualization. In fact, the system can be viewed as the private cloud solution for enterprise or organization without software charge.

Meanwhile, this thesis also has experiments about live migration in advanced discussion and carries on the system performance test using the KVM [2, 20, 26, 27]. We can observe the features of live migration by experiments which help us move the VMs in daily operation.

1.3 Thesis Organization

In chapter 2, we briefly sum up the trend of cloud computing so far. We describe how we design the system and demonstrate how we develop the system in chapter 3. In chapter 4 we have the experiments and examined the experimental results. Finally in chapter 5, we discuss the conclusions and future Work.

Chapter 2

Background Review

2.1 Cloud Computing

Cloud computing is an Internet-based computing model. The shared software, hardware, and information in this model be supplied the needs of many computers and devices. This model is simply like an electric net. Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a metered service over a network (typically the Internet).[1].

The National Institute of Standards and Technology (NIST) stated the following definition of cloud computing: "*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) that can be rapidly provisioned and released with minimal management effort or service provider interaction .This cloud model is composed of five essential characteristics, three service models, and four deployment models."*

The three services models are:

- Infrastructure as a service
- Platform as a service
- Software as a service

And according to the definition from the NIST is: "*The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include* *operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).*"

In fact, IaaS is not dimply the management and deployment of VMs. There are many issues about the exchanging between different IaaS. As we know, the cloud can be divided into public cloud, private cloud and hybrid cloud. The public cloud aims to provide service to everyone who can reach the internet. The private cloud is to provide specially service for a company employee or in government or in school. The hybrid cloud means the enterprise have both private cloud and public cloud. How to exchange the VMs smoothly in different cloud type is an important issue.

2.2 Virtualization

Virtualization is simply the logical separation of requests for some services from the physical resources where the service is actually provided. In practical terms, virtualization allows applications, operating systems, or system services in a logically distinct system environment to run independently of a specific physical computer system. Obviously, all of these must run on a certain computer system at any given time, but virtualization provides a level of logical abstraction that liberates applications, system services, and even the operating system that supports them from being tied to a specific piece of hardware. Virtualization, focusing on logical operating environments, makes applications, services, and instances of an operating system portable across different physical computer systems. Virtualization can execute applications under many operating systems, manage IT more efficiently, and allot computing resources with other computers [2].

Virtualization has hardware imitate much hardware through a Virtual Machine Monitor, and each virtual machine functions as a complete individual unit. A virtual machine is composed of memories, CPUs, unique complete hardware equipment, and so on. It can run any operating system as Guest OS without affecting other virtual machines. In general, most virtualization strategies fall into one of two major categories:

Full virtualization also called native virtualization is similar to emulation. As in emulation, unmodified operating systems and applications run within a virtual machine. Full virtualization differs from emulation because operating systems and applications run on the same architecture as the underlying physical machine. This allows a full-virtualization system to run many instructions directly on raw hardware. The hypervisor in this case monitors access to the underlying hardware and gives each guest operating system the illusion of having its own copy. It no longer has to use software to simulate a different basic architecture (Figure 2-1.).

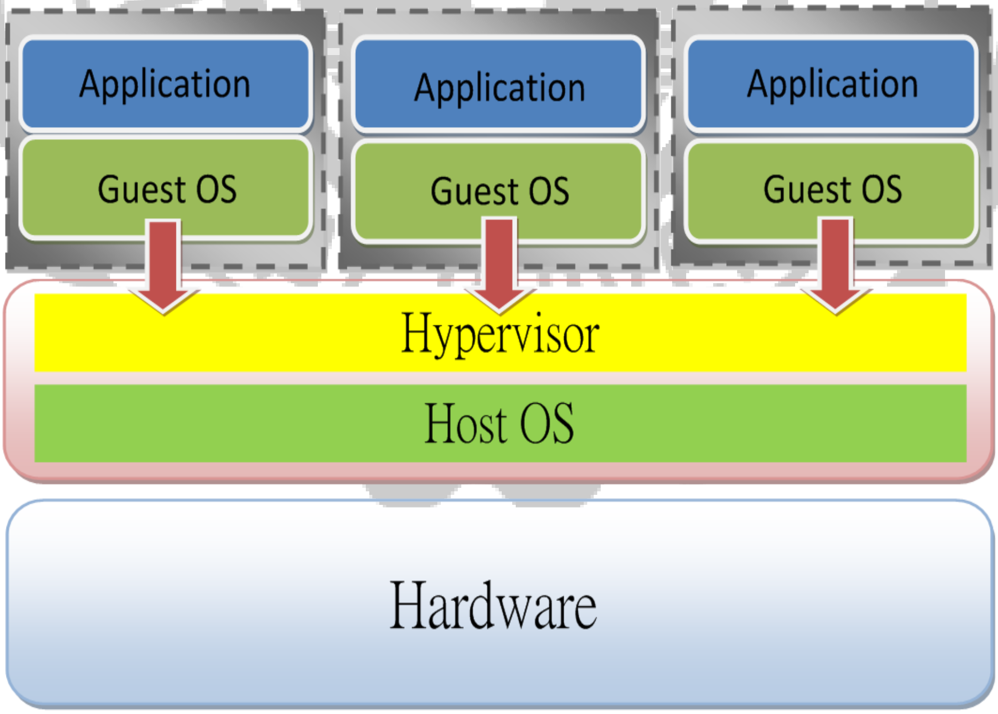

Figure 2-1. The architecture of full virtualization

For Para-virtualization, the hypervisor exports a modified version of the underlying physical hardware. The exported virtual machine has the same architecture, which is not necessarily the case in emulation. Instead, targeted modifications make it simpler and faster to support multiple guest operating systems. For example, the guest operating system might be modified to use a special hyper called application binary interface (ABI) instead of using certain architectural features. This means that only small changes are typically necessary in the guest operating systems, but any changes make it difficult to support closed-source operating systems that are only distributed in binary form, such as Microsoft Windows. As in full virtualization, applications are still in run without modifications.

Para-virtualization such as full virtualization uses a hypervisor and virtual machine; the term refers to its virtualized operating systems as well. However, unlike full virtualization, para-virtualization requires changes to the virtualized operating system. This allows the VM to coordinate with the hypervisor and reduces the use of the privileged instructions typically responsible for major performance penalties in full virtualization.

Para-virtualized virtual machines typically outperform fully-virtualized virtual machines. However, it is necessary to modify the para-virtualized virtual machine or operating system to be hypervisor-aware (Figure 2-2.).

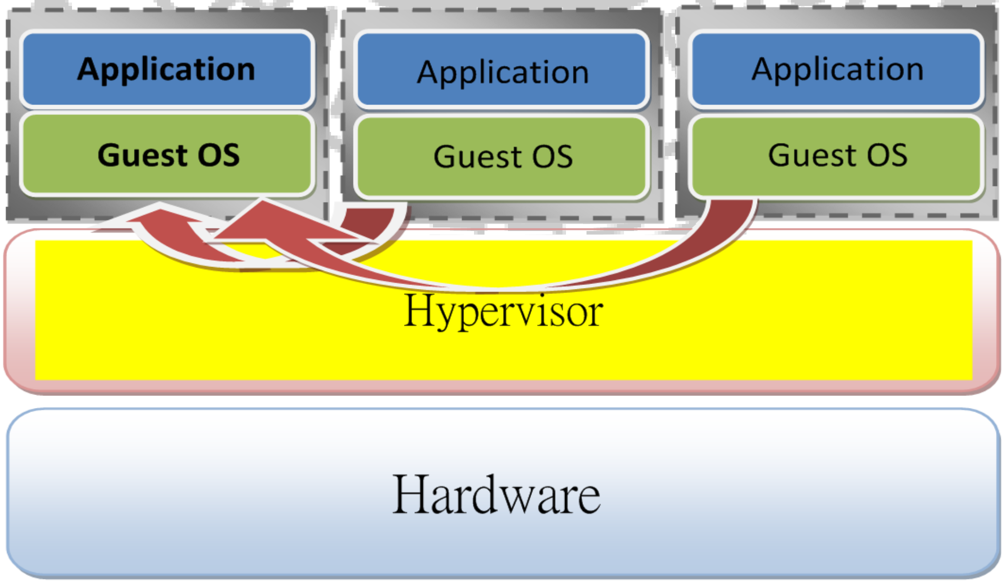

Figure 2-2. The architecture of para-virtualization

To evaluate the viability of differences between virtualization and non-virtualization, this thesis uses the virtualization software Xen. Xen is a virtual machine monitor (hypervisor) that allows you to use one physical computer to run many virtual computers — for example, a web server application and a test server run on the same physical machine or Linux and Windows run simultaneously. Although a virtualization system is not the only one is available, Xen combines features making it uniquely suited for many important applications. Xen runs on commodity hardware platforms and is open-source as well. Xen is fast and scalable, and provides server-class features such as live migration. Xen provides better efficiency, supports different operating system work simultaneously, and gives each operating system an independent system environment.

2.3 Open Source for Virtualization

2.3.1 KVM

KVM is open source software used in the Linux kernel virtualization infrastructure and its full name is: "*kernel-based virtual machine*". In hardware architecture, it is supported x86 and x86-64 processors. Windows Family and most of Linux distribution are supported by KVM. With CPU virtualization infrastructures (such as the Intel VT-x or AMD-V), the virtualization technology of KVM is called “hardware-assisted virtualization”.

KVM contains a processor which provided the kernel virtualization can be loaded core module kvm.ko (kvm-intel.ko or kvm-amd.ko). It also requires a modified QEMU software (qemu-kvm), as the interface of the virtual machine. KVM can operate multiple virtual machines in the same time (It means that multiple virtual machines using the same image) for each virtual machine, configured the hardware environment. In actually, it’s full virtualization with hardware-assisted virtualization. In the Linux kernel 2.6.20 above the core contains the KVM .

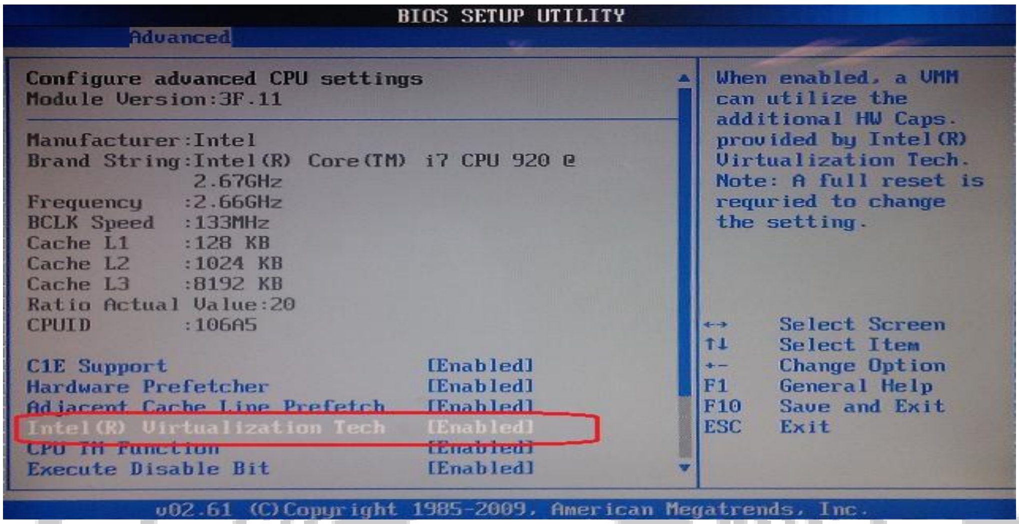

Figure 2-3. The H/W Enabled

Figure 2-4.The S/W (OS) Supported

2.3.2 Xen

Xen is an open source virtual machine monitor, it's developed by the University of Cambridge.

- The para-virtualization of Xen

  Xen can operate virtualization in the old (non-virtual hardware) machine architecture but processor and operating system must be modified ("transplant"). It makes Xen to achieve high-performance virtualization without special hardware support. Even in some extremely unfriendly traditional architecture (x86), the Xen also has an excellent performance. But in the condition with modified OS, para-virtualization can't simulate Windows OS.

- The full virtualization of Xen.

  If system processor support the expansion of virtual hardware (Intel and AMD on the local support virtualization extensions), this technology will allow Xen virtual machine running without modified host OS.

2.3.3 Open VZ

Open VZ is operating system level virtualization technology based on the Linux kernel and operating system. Open VZ allows a physical server to run multiple operating systems, it called virtual private server (VPS, Virtual Private Server) or virtual environment (VE Virtual Environment). Compared with KVM virtual machines and Xen para-virtualized, the Open VZ host OS and guest OS Required Linux (although in different virtual environments with different Linux distributions). According to the Open VZ website, use Open VZ compared to using a separate server, performance will only have a 1-3% loss. Open VZ divide into two parts, a modified operating system kernel and user tools.

2.4 Virtualization Management

The virtual machine is not only available user interface, but it is the computer with actual loading. Management of virtual machines and management of physical systems are equally important. Virtualization Management includes a set of integrated management tools, can be minimize complexity and simplify the operation. It should centrally manage physical and virtual IT infrastructure, increased server utilization, but also across multiple virtualization platforms to optimize dynamic resources.

2.4.1 OpenNebula

OpenNebula is the ***industry standard open-source product for data center virtualization***, offering the most feature-rich, customizable solution to build virtualized enterprise data centers and private cloud infrastructures on Xen, KVM and VMware deployments, and providing cloud consumers with choice of interfaces, from open cloud to de-facto standards, like the EC2 API.[16,32]

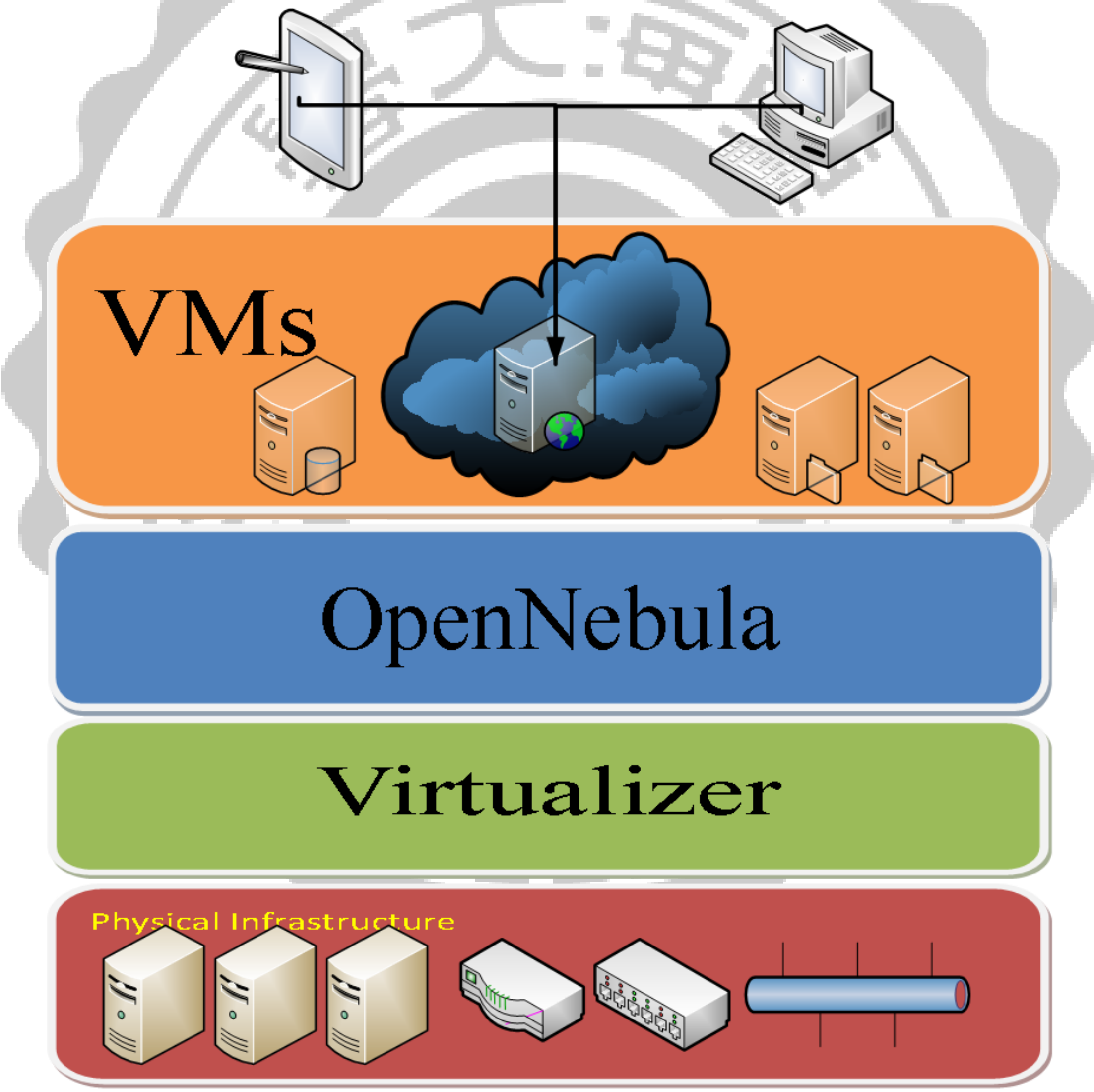

Figure 2-5. The diagram of OpenNebula

2.4.2 Open Stack

The Open Stack is the IaaS (Infrastructure as a Service) software so that anyone can create your own cloud computing services. There are three main components below [31].

- Open Stack Compute: Provision and manage large networks of virtual machines.
- Open Stack Object Store: Create petabytes of secure, reliable storage using standard hardware.
- Open Stack Glance: Catalog and manage massive libraries of server images.

2.5 Live Migration

By adjusting the resources with virtual technology, to make provided services to closer to the actual needs of different users. Live migration of virtual machines is an important technology. The live-migration of VM can transfer VM to other physical servers without shutdown. It achieve the high HA ability with the non-stop services (Figure 2-6.).

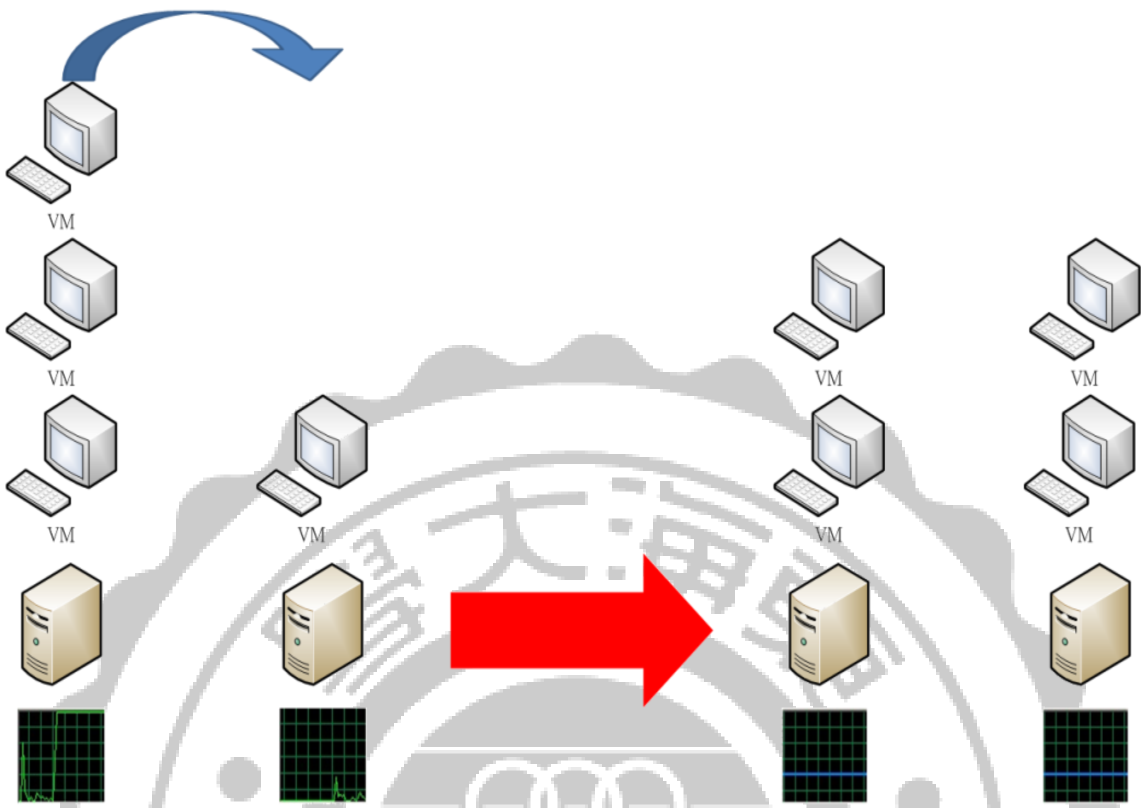

Figure 2-6. The concept of Live Migration

Live migration is the movement of a virtual machine from one physical host to another while continuously powered-up. When this process functions properly, it shows no noticeable effect from the end-user's point of view (Figure 2-7). An administrator don't need to take a virtual machine offline for maintenance or upgrading by live migration. When resources are virtualized, additional management of VMs is needed to create, terminate, clone or move VMs from host to host. Migration of VMs can be done off-line (the guest in the VM is powered off) or on-line (live migration of a running VM to another host).

One of the most significant advantages of live migration is that it facilitates proactive maintenance. If an imminent failure is suspected, the potential problem can be resolved before service disruption. Live migration can also be used for load balancing, in which work is shared among computers to optimize the usage of available CPU resources.

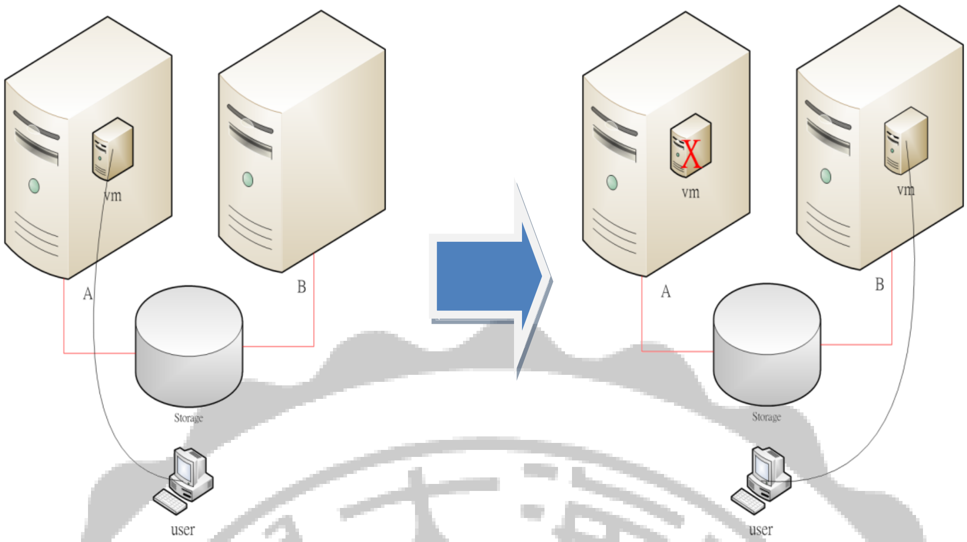

Figure 2-7. The successful process of Live Migration

2.6 Related Work

Stones from other hills may serve to polish the wrong. There are many researches about the live migration and some have been proposed to improve the downtime of live migration. In this section, some related works will be presented briefly.

The most important thesis about live migration is "Live Migration of Virtual Machines" [35]. Discussion in this thesis [35] is also the implementation of the live migration in Xen. To avoid difficulties of the migration in the process level and residual dependency, instead of the process, the VM with its application regarded as the migration unit. The main goal of live migration is to reduce the time of down time and total migration time.

- Downtime[35]: The time of shutdown during migration
- Total migration time: The VM can run in the target host without error and source host discarded the old VM.

The method of live migration in this thesis is "pre-copy". Most methods of live migration in business are “pre-copy”, too. Another important paper about live migration is "post-copy

Live Migration of Virtual Machines" [17]. The method proposed in this thesis is "post-copy". The purpose of pre-copy method is trying to reduce the down time. But the purpose of post-copy method is trying to reduce the total migration time.

In recent years, there are many papers or theses about live migration published. We pick up the the article which is relative to this thesis in Taiwan. First, the author want to find out the balance between “pre-copy” and “post-copy” in the thesis [36]. So he proposed a compromise approach. In limited rounds, if “pre-copy” didn’t complete the migration task, the mechanism which proposed by the author would use “post-copy” to do migration.

Which is good or bad depends on the case. But if we can accept the obvious downtime, we don’t need live migration. And there is a big disadvantage of “post-copy”. If the live migration of “post-copy” is broken, the content of VM will be crash. The same VM crash event will not occur in “pre-copy”. This is because there is always a full image of VM existing during “pre-copy” live migration. So most of cloud platforms implement the live migration with “pre-copy” types. So we will test the features of live migration with “pre-copy” in our experiments.

Chapter 3

System Implementation

The goal of our system is to build an elastic deployment environment of VM. In section 3.1, we will overview the whole system. And then we will introduce our designs one by one.

3.1 System Overview

In Figure 3-1 shows the components of the three Cloud Models and point out the emphasis of this thesis. This system has the web-based interface to manage virtual machine. And the system shows the CPU utilization, host loading, memory utilization and VMs information etc.

Besides managing individual VMs' life cycles, this study also designs the core to support service deployment. Such services typically include a set of interrelated components (for example, a Web server and database back end) requiring several VMs. Thus, a group of related VMs becomes a first-class entity in OpenNebula. Besides managing the VMs as a unit, the core also handles the context information delivery (such as the Web server's IP address, digital certificates, and software licenses) to the VMs [8].

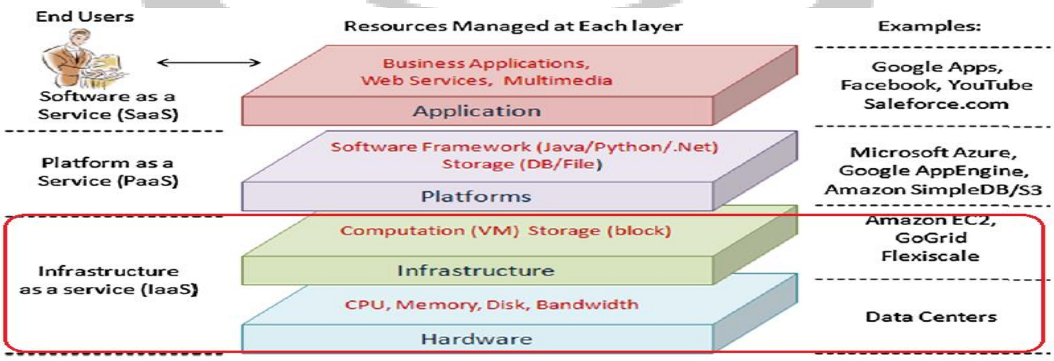

Figure 3-1. the domain of IaaS

3.2 System Architecture

We use OpenNebula to build the IaaS environment. OpenNebula is the open source toolkit that enables the dynamic deployment and reallocation of virtual machines in a pool of physical resources. OpenNebula extends the benefits of virtualization platforms from a single physical resource to the pool of resources, decoupling the server, from both the physical infrastructure and the physical location [4]. OpenNebula contains one front end and multiple back ends. OpenNebula orchestrates storage, network, virtualization, monitoring, and security technologies to enable dynamic placement of multi-tier services (groups of interconnected virtual machines) on distributed infrastructures, combining both data center resources and remote cloud resources, according to allocation policies [4](Figure 3-2.).

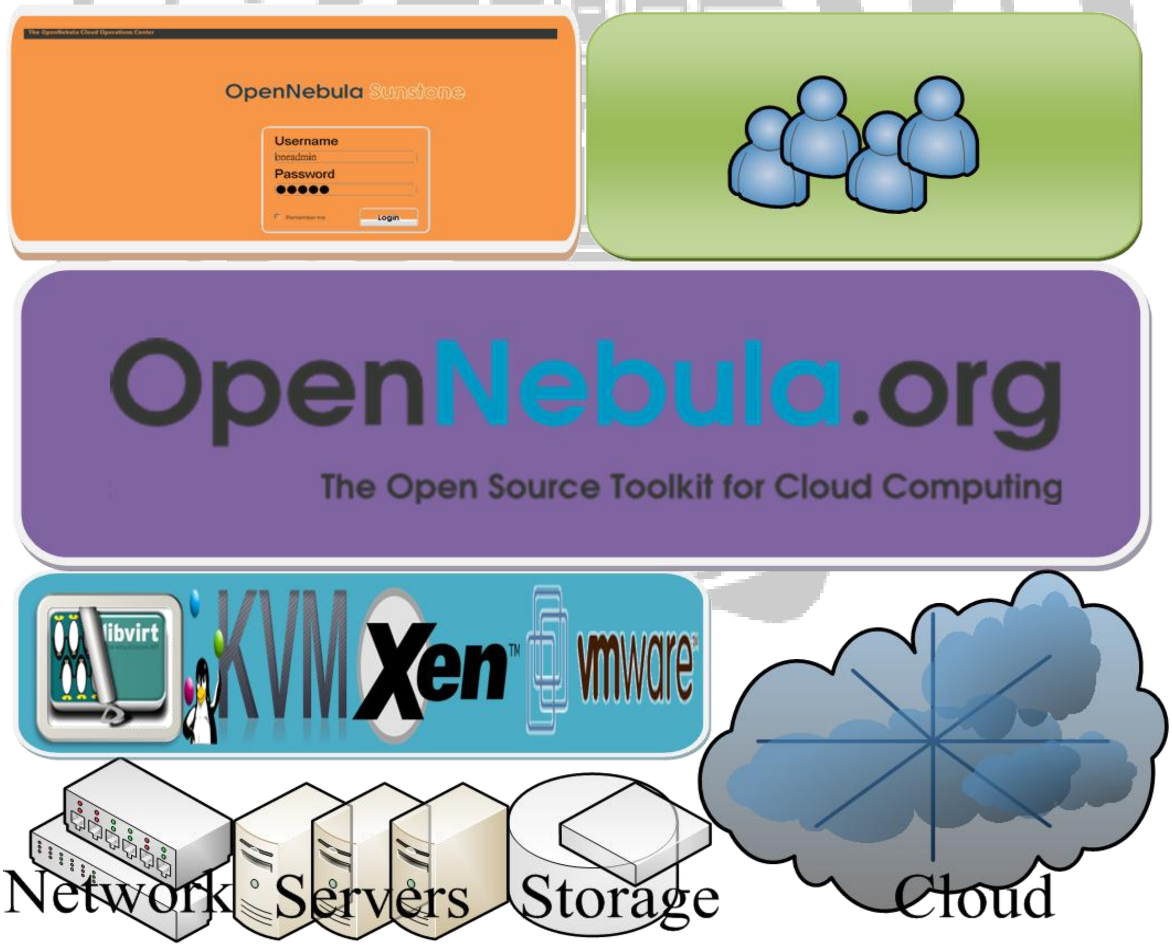

Figure 3-2. The Concept of OpenNebula

OpenNebula is composed of three main components:

- The OpenNebula Core is a centralized component that manages the life cycle of a VM by performing basic VM operations, and also provides a basic management and monitor interface for the physical hosts.
- The Capacity Manager governs the functionality provided by the OpenNebula core. The capacity manager adjusts VM placement based on a set of predefined policies.
- Virtualizer Access Drivers. To provide an abstraction for the underlying virtualization layer, OpenNebula uses pluggable drivers that expose the basic functionality of the hypervisor [5].

As an Infrastructure as a Service (IaaS) provider, this thesis applies an ideas of virtualizes in the cloud system to economize power, web interface and user friendly to manage the virtual machines. Therefore, there are some distinct on framework of cloud; our system architecture is shown in Figure 3-3. About user friendly, users simply connect to the site through the Internet, and then set their own needs, you can create a virtual machine, the user does not need to know what happened back may need to set any object, they can be consistent with their own needs of virtual machines. The Screen is shown from Figures 3-4.

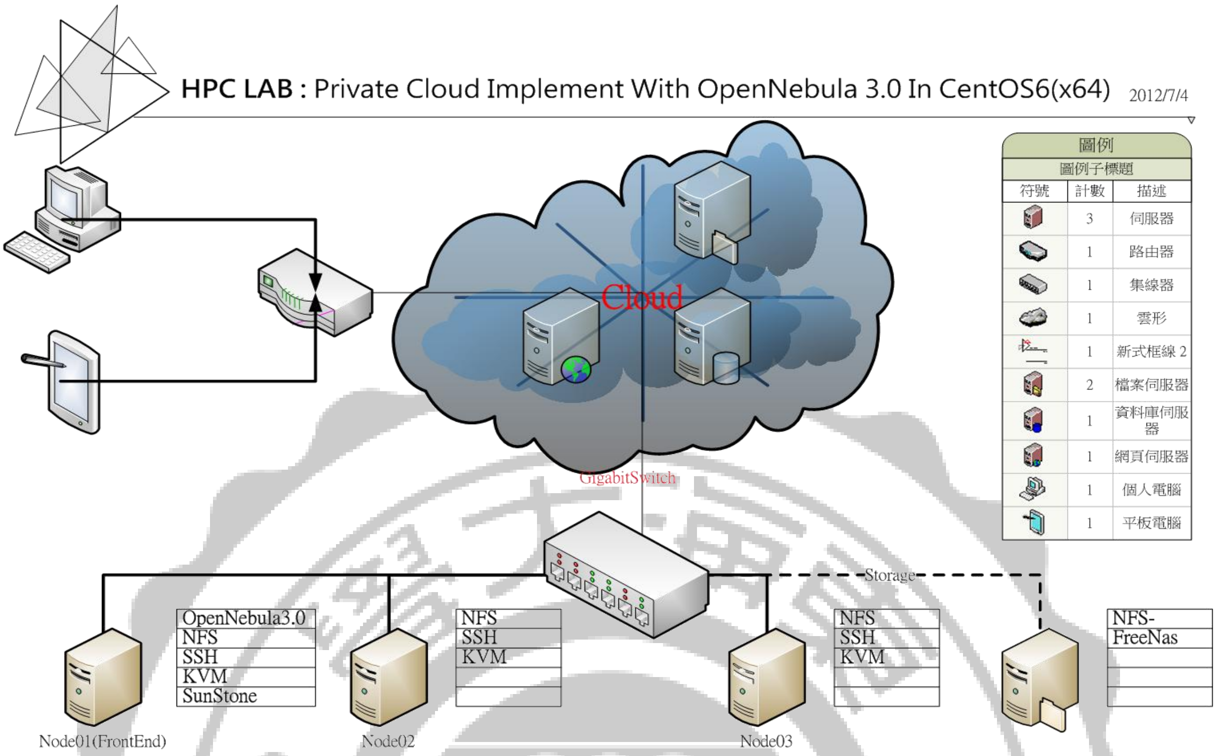

Figure 3-3. System architecture

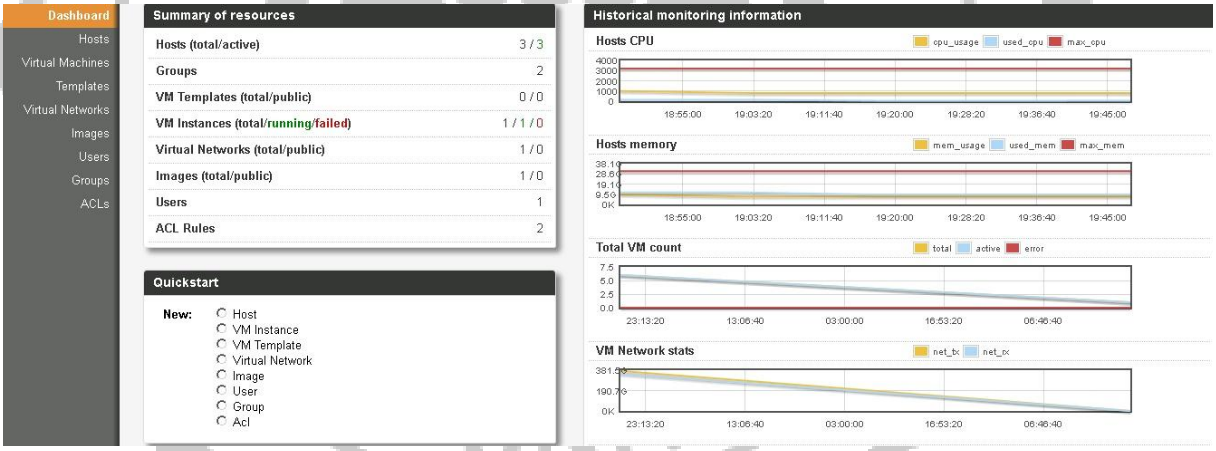

Figure 3-4. Initial presentation of OpenNebula

And we can make VM to live migrated on line by the web-based interface (Figures 3-5).

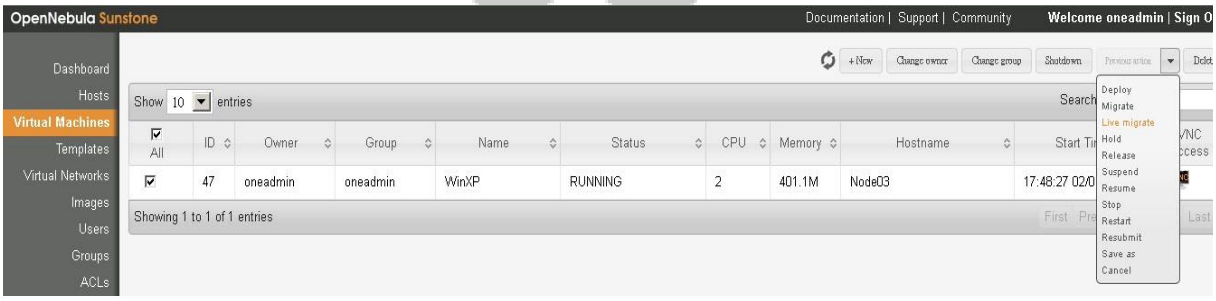

Figure 3-5. Live migration of the VM from Node3 to Node2

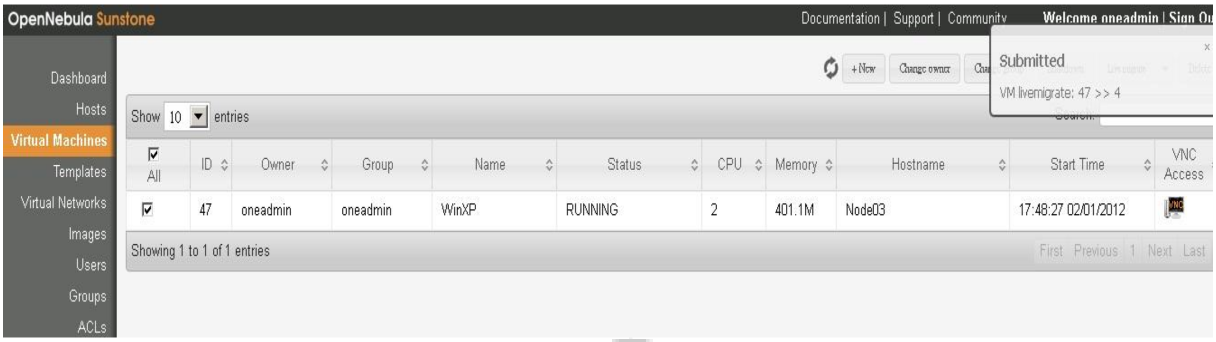

Figure 3-6. Submitted

Figure 3-7. Live migrated Success

3.3 Live Migration Mechanism of VM

Live migration of virtual machines is to keep the virtual machine running and migrate the VM and services running as a migration unit from the source physical machine to the destination machine at the same time. The services running on the virtual machine will always be able to respond to user. When the migration is complete, the virtual machine (service) resume in the destination physical machine. The time of service interrupted is very short. In order to ensure virtual machines can keep running in target physical machine after migration, it must be send adequate information, such as disk, memory, the CPU, the I / O devices, etc.. Among them, the information of the memory is more complex and essential for migration. In figure3-8, we can clearly distinguish the procedure of live migration step by step.

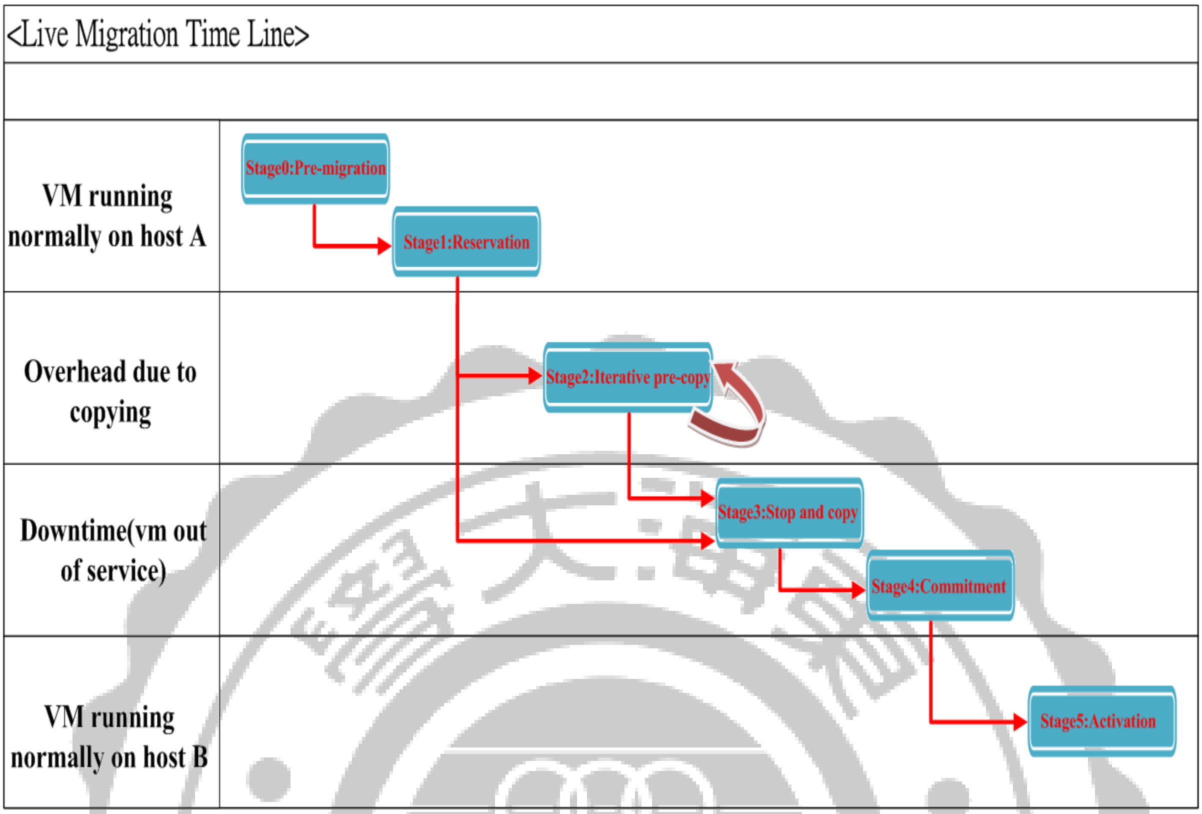

Figure 3-8. Live migration Time Line

3.4 Applications (MIFAS)

We build the application which is called Medical Image File Access System (MIFAS) to implement the SaaS. The purpose of the MIFAS system is to access medical images for telemedicine in the cloud. It is the web-based system like Figure 3-9.

Figure 3-9. MIFAS login interface

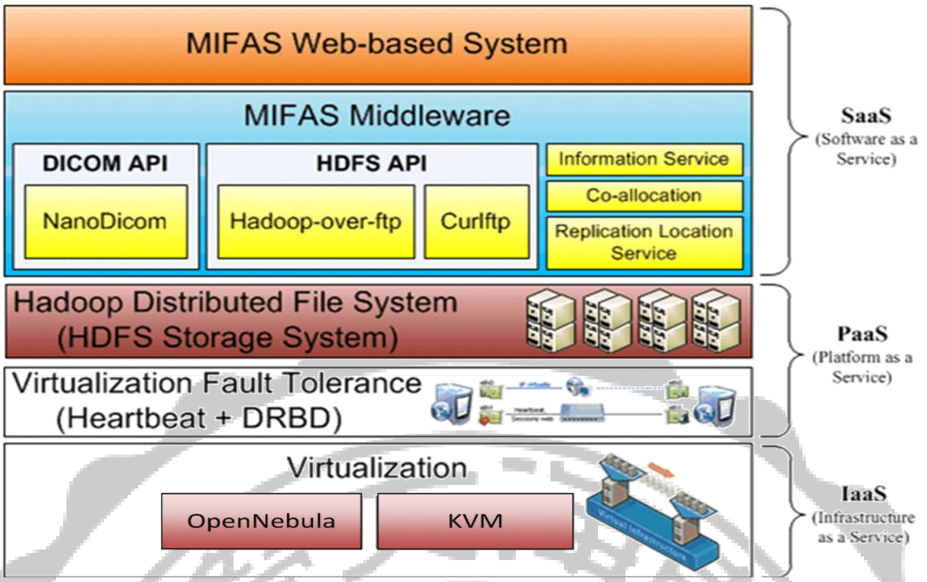

Figure 3-10. The System Components of MIFAS

We can see the detail description of MIFAS components in Figure 3-10. It can briefly introduce the MIFAS with dividing into three service models.

- SaaS: From the top level a web-based system was provided a GUI interface that users or administrators could manage patient's data on it also including the quick view of medical images [Figure 3-11]. And if need, another function can check the images for detail view [Figure 3-12]. We also deploy the Apache and MySQL services in the MIFAS system. Apace help us to provide the web service. MySQL is the database for management of MIAS user accounts.

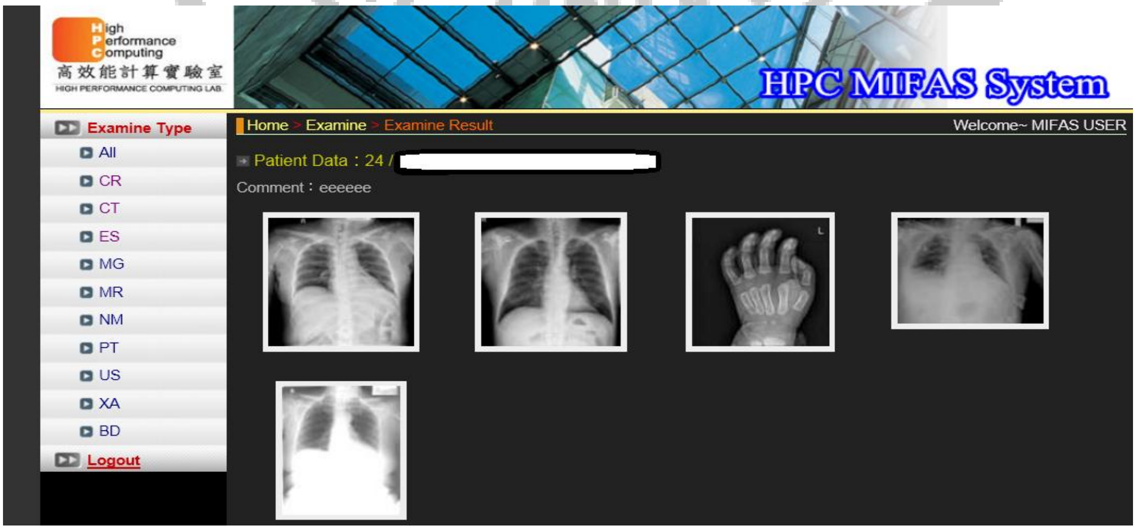

Figure 3-11. Quick view of medical images

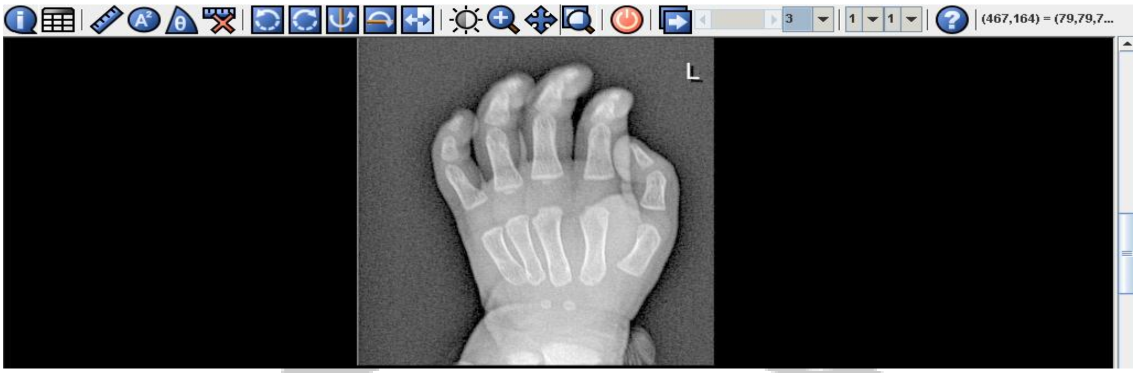

Figure 3-12. The view of medical image.

- PaaS: In order to solve a lot of accessing images, we use the Hadoop to be the distributed file system in the MIFAS. So the medical images are ranged by Hadoop mechanism. This Hadoop platform could be described as PaaS (Platform as a Service).
- IaaS: In our previous introduction, we used OpenNebula to manage our VMs. So the MIFAS is operated on the VMs by the OpenNebula. We can make the live migration of MIFAS system between different hosts to avoid high loading or other purpose.

With MIFAS, Hadoop file system and OpenNebula platform working in coordination, we really implement the combination of the three service models(SaaS, PaaS and IaaS) in our cloud platform.

Chapter 4
Experimental Results

This thesis focuses on testing of live migration features and efficiency of KVM virtual machines, including computing performance between physical machine and VM. The experiment issues are listed as below:

- We try to ping the VM during live migration process. In order to comprehend whether the VM keeps working (service) during live migration, it is a success to get good result at the first experiment tested the ping action.
- Does different memory sizes of the VM initially deployed require different time for live migration? Every VM is deployed by different memory sizes. We would like to know whether this will affect time of live migration or not. In this task, we use such as 512MB, 1GB, 2GB, and 4GB etc., different sizes of memory to deploy the settings of the virtual machine. After VMs deploying conducted, we start to test the live migration.
- HPCC performance testing -We implement HPCC performance measurement in the same hardware with 4 different platforms.

  (1). PM (Physical Machine)

  (2). KVM

  (3). VM Ware 8.0 in Linux

  (4). VM Ware 8.0 in Win7 Professional

The host OS of (1), (2) and (3) is CentOS6.2 x64 (Desktop version). We can compare the differences of physical machine, KVM and VMware. And we choose the VM Ware because VMWare and KVM are the technique of full-virtualization. Moreover, we can compare the pop technique of full-virtualization in Linux and Windows.

- While downloading a file, we try to start the live migration to move VM. Download file becomes a normal situation for daily operation. We would like to knoe the results between WAN and LAN. Is live migration performance the same in WAN and LAN while downloading files?

4.1 Experimental Environment

The host list of the test environment as below.

Table 4-1. Environment Specification

<table>
<tr><th colspan="7">Environment Spec.</th></tr>
<tr><th></th><th>CPU</th><th>Memory</th><th>Disk</th><th>Network</th><th>OS</th><th>Software</th></tr>
<tr><td>Node01</td><td>I7-860 2.8GHz</td><td>4GB</td><td>1TB</td><td rowspan="3">Gigabits</td><td rowspan="3">CentOS6 x64</td><td>OpenNebula-Front End/Node</td></tr>
<tr><td>Node02</td><td>I7-990 3.47GHz</td><td>12GB</td><td>2TB</td><td rowspan="2">OpenNebula-Node</td></tr>
<tr><td>Node03</td><td>I7-3960X 3.3GHz</td><td>16GB</td><td>6TB</td></tr>
</table>

4.2 Results

4.2.1 Experiment 1: VM has very short download time during the live migration.

In the experiment, this thesis tested the action of ping during the VM in live migration. First, we confirm the initial status of environment. The VM of Node02 will be live-migrated to Node03. Before live migration, we start the ping action from Node02 until the VM migration finished. From Figures 4-1 to 4-5, we inspect that VM continues to respond the ping message during live migration process. All live migration process of ping testing is a success without interruption yet only slight fluctuations (Figure 4-6).

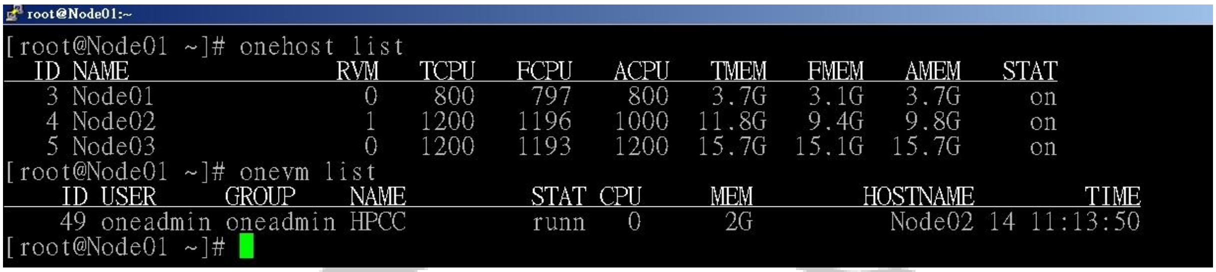

Figure 4-1. The initiation of live migration

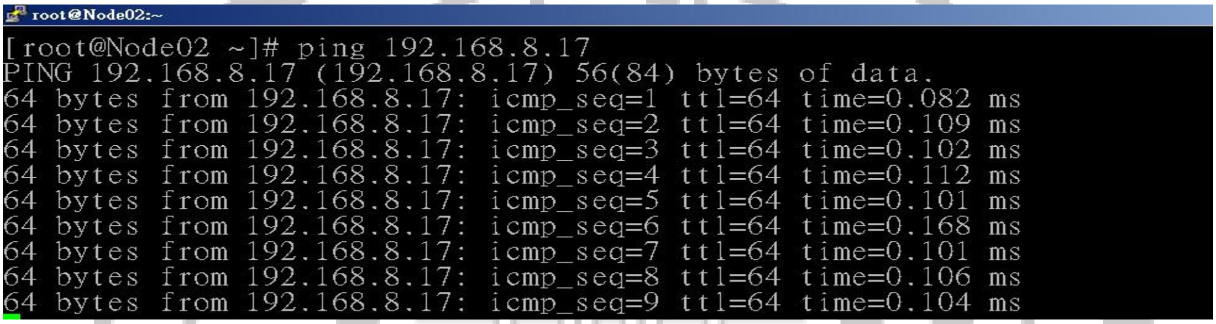

Figure 4-2. The ping of live migration

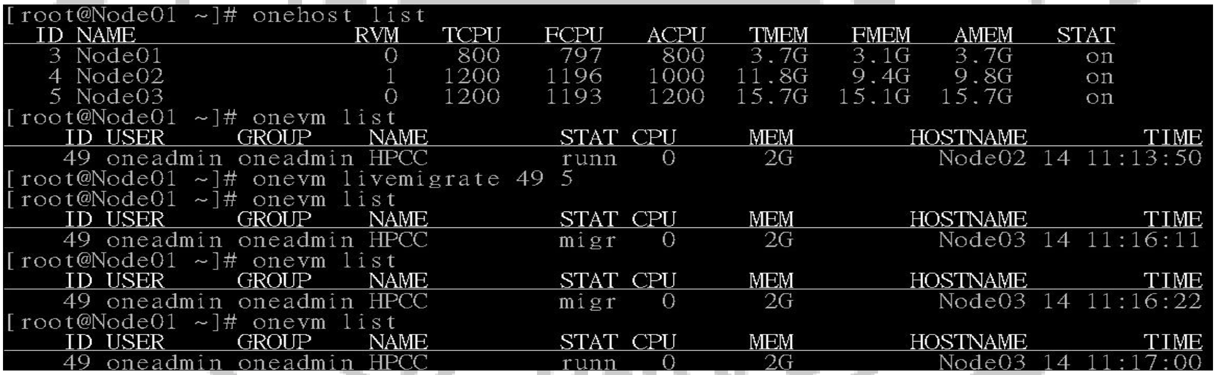

Figure 4-3. The end of live migration

```
root@Node02:~
64 bytes from 192.168.8.17: icmp_seq=37 ttl=64 time=0.101 ms
64 bytes from 192.168.8.17: icmp_seq=38 ttl=64 time=0.107 ms
64 bytes from 192.168.8.17: icmp_seq=39 ttl=64 time=0.104 ms
64 bytes from 192.168.8.17: icmp_seq=40 ttl=64 time=0.109 ms
64 bytes from 192.168.8.17: icmp_seq=41 ttl=64 time=0.102 ms
64 bytes from 192.168.8.17: icmp_seq=42 ttl=64 time=0.107 ms
64 bytes from 192.168.8.17: icmp_seq=43 ttl=64 time=0.100 ms
64 bytes from 192.168.8.17: icmp_seq=44 ttl=64 time=0.105 ms
64 bytes from 192.168.8.17: icmp_seq=45 ttl=64 time=0.104 ms
64 bytes from 192.168.8.17: icmp_seq=46 ttl=64 time=0.105 ms
64 bytes from 192.168.8.17: icmp_seq=47 ttl=64 time=0.100 ms
64 bytes from 192.168.8.17: icmp_seq=48 ttl=64 time=0.107 ms
64 bytes from 192.168.8.17: icmp_seq=49 ttl=64 time=0.101 ms
64 bytes from 192.168.8.17: icmp_seq=50 ttl=64 time=0.102 ms
64 bytes from 192.168.8.17: icmp_seq=51 ttl=64 time=0.104 ms
64 bytes from 192.168.8.17: icmp_seq=52 ttl=64 time=0.111 ms
64 bytes from 192.168.8.17: icmp_seq=53 ttl=64 time=1.01 ms
64 bytes from 192.168.8.17: icmp_seq=54 ttl=64 time=0.179 ms
64 bytes from 192.168.8.17: icmp_seq=55 ttl=64 time=0.211 ms
64 bytes from 192.168.8.17: icmp_seq=56 ttl=64 time=0.223 ms
64 bytes from 192.168.8.17: icmp_seq=57 ttl=64 time=0.250 ms
64 bytes from 192.168.8.17: icmp_seq=58 ttl=64 time=0.114 ms
64 bytes from 192.168.8.17: icmp_seq=59 ttl=64 time=0.217 ms
64 bytes from 192.168.8.17: icmp_seq=60 ttl=64 time=0.196 ms
```

Figure 4-4. The ping during live migration

```
root@Node02:~
64 bytes from 192.168.8.17: icmp_seq=151 ttl=64 time=0.434 ms
64 bytes from 192.168.8.17: icmp_seq=152 ttl=64 time=0.429 ms
64 bytes from 192.168.8.17: icmp_seq=153 ttl=64 time=0.472 ms
64 bytes from 192.168.8.17: icmp_seq=154 ttl=64 time=0.518 ms
64 bytes from 192.168.8.17: icmp_seq=155 ttl=64 time=0.415 ms
64 bytes from 192.168.8.17: icmp_seq=156 ttl=64 time=0.450 ms
64 bytes from 192.168.8.17: icmp_seq=157 ttl=64 time=0.402 ms
64 bytes from 192.168.8.17: icmp_seq=158 ttl=64 time=0.443 ms
64 bytes from 192.168.8.17: icmp_seq=159 ttl=64 time=0.339 ms
64 bytes from 192.168.8.17: icmp_seq=160 ttl=64 time=0.428 ms
64 bytes from 192.168.8.17: icmp_seq=161 ttl=64 time=0.425 ms
64 bytes from 192.168.8.17: icmp_seq=162 ttl=64 time=0.420 ms
64 bytes from 192.168.8.17: icmp_seq=163 ttl=64 time=0.413 ms
64 bytes from 192.168.8.17: icmp_seq=164 ttl=64 time=0.411 ms
64 bytes from 192.168.8.17: icmp_seq=165 ttl=64 time=0.497 ms
64 bytes from 192.168.8.17: icmp_seq=166 ttl=64 time=0.446 ms
64 bytes from 192.168.8.17: icmp_seq=167 ttl=64 time=0.440 ms
64 bytes from 192.168.8.17: icmp_seq=168 ttl=64 time=0.431 ms
64 bytes from 192.168.8.17: icmp_seq=169 ttl=64 time=0.430 ms
64 bytes from 192.168.8.17: icmp_seq=170 ttl=64 time=0.420 ms
^C
--- 192.168.8.17 ping statistics ---
170 packets transmitted, 169 received, 0% packet loss, time 169608ms
rtt min/avg/max/mdev = 0.080/0.318/1.146/0.175 ms
[root@Node02 ~]#
```

Figure 4-5. The ping result after live migration finish

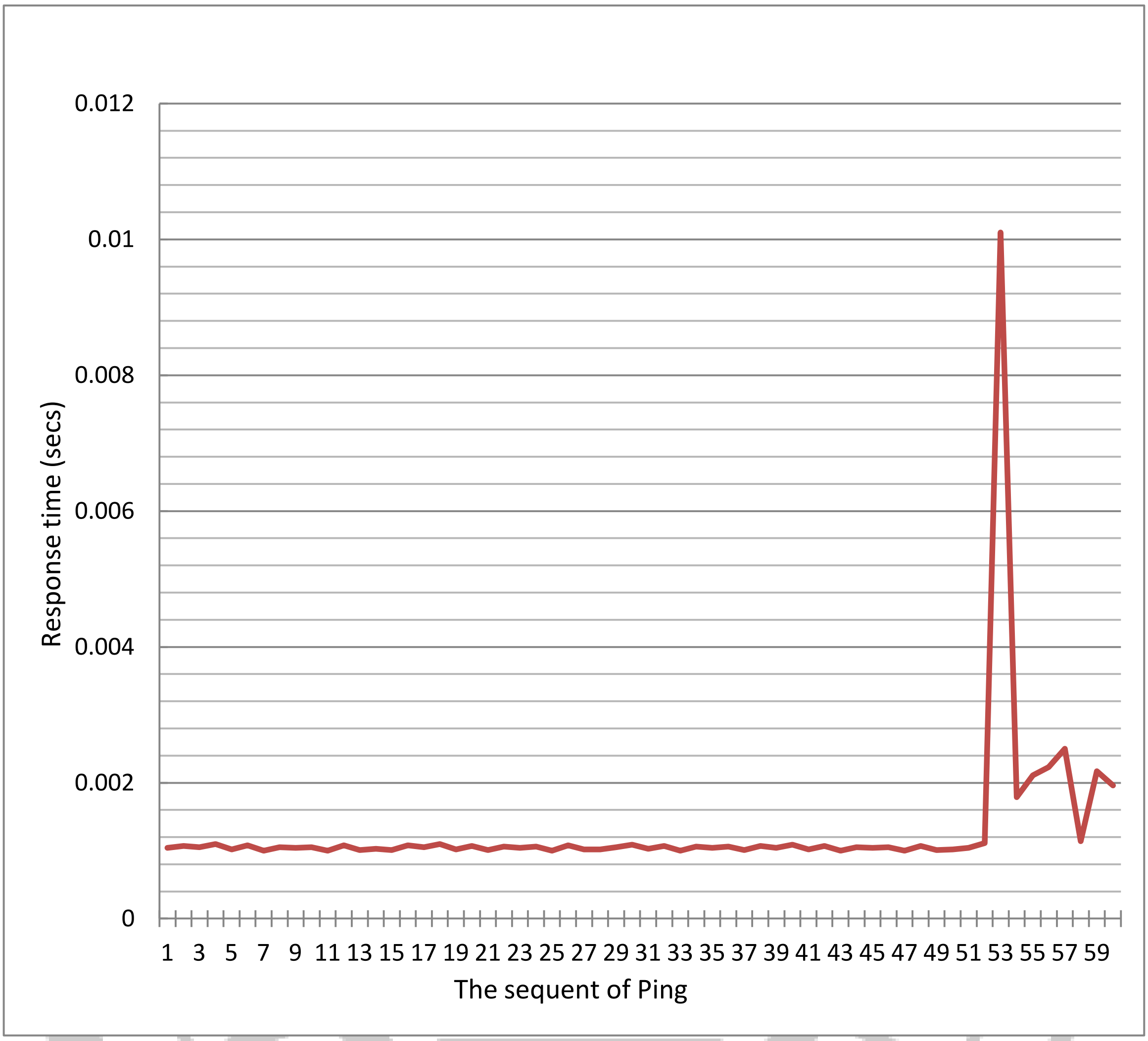

Figure 4-6. Collect ping reply time

4.2.2 Experiment 2: The initial VMs with different system memory sizes have similar live migration time.

We use different memory sizes such as 512MB, 1GB, 2GB, and 4GB so on to deploy the settings of virtual machine. After VMs finishing deploying, we test the live migration in clear OS status without installing other services. We found that the difference is slight (Figure 4-7), and are likely to exist within magnitude of human errors.

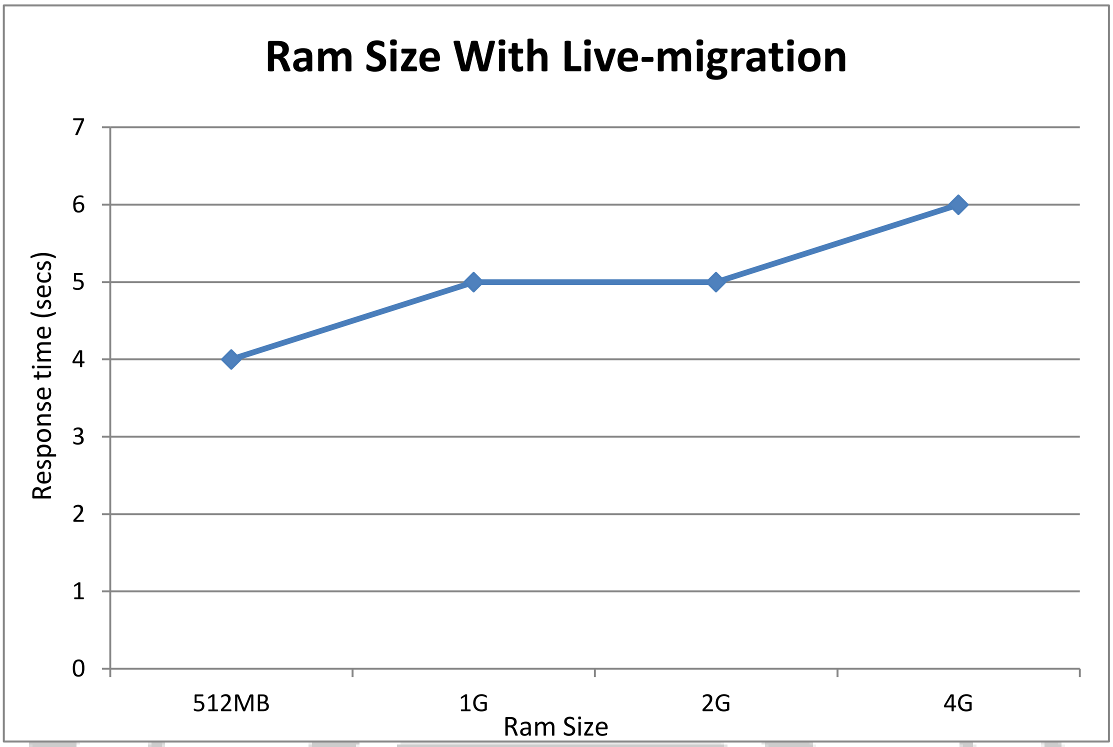

Figure 4-7. Time of live migration

4.2.3 Experiment 3: the comparison of physical machine- KVM and VMware.

In order to avoid running service on testing environments which affects machine performance, we do not pick up the host from cloud platform. The HPCC testing is compared to physical machine, KVM and VMW which are on the same with independent hardware VMs. We compared the differenced of performance and computation by controlling the number of CPU. We list the hardware and software specification in the following:

Table 4-2. Environment of Experiment 3 Specification

	Host OS	Guest Os	H/W SPEC		
Physical Machine	CentOS6.2 x64 (Desktop)	N/A	Intel E3400 2.6GHz dual cores	Ram 8GB	HDD 500GB
KVM	CentOS6.2 x64 (Desktop)	CentOS6.2 x64 (Desktop)			
VM Ware 8.0	CentOS6.2 x64 (Desktop)	CentOS6.2 x64 (Desktop)			

VM Ware 8.0	Win7 Professional	CentOS6.2 x64 (Desktop)			

Shown in Figure 4-8 for single core case, we find that the gap of these four platforms is small in low amount of computation (Ns =5000). By increasing the amount of computation, the tested value of KVM is closed to PM. However, the tested value of VMware lags far behind the KVM and PM. The result of the order is PM, KVM, VMware in WIN7, VMware in Linux. On the other hand, the gap of computation in different VMs is small other than the physical machine in Figure 4-9.

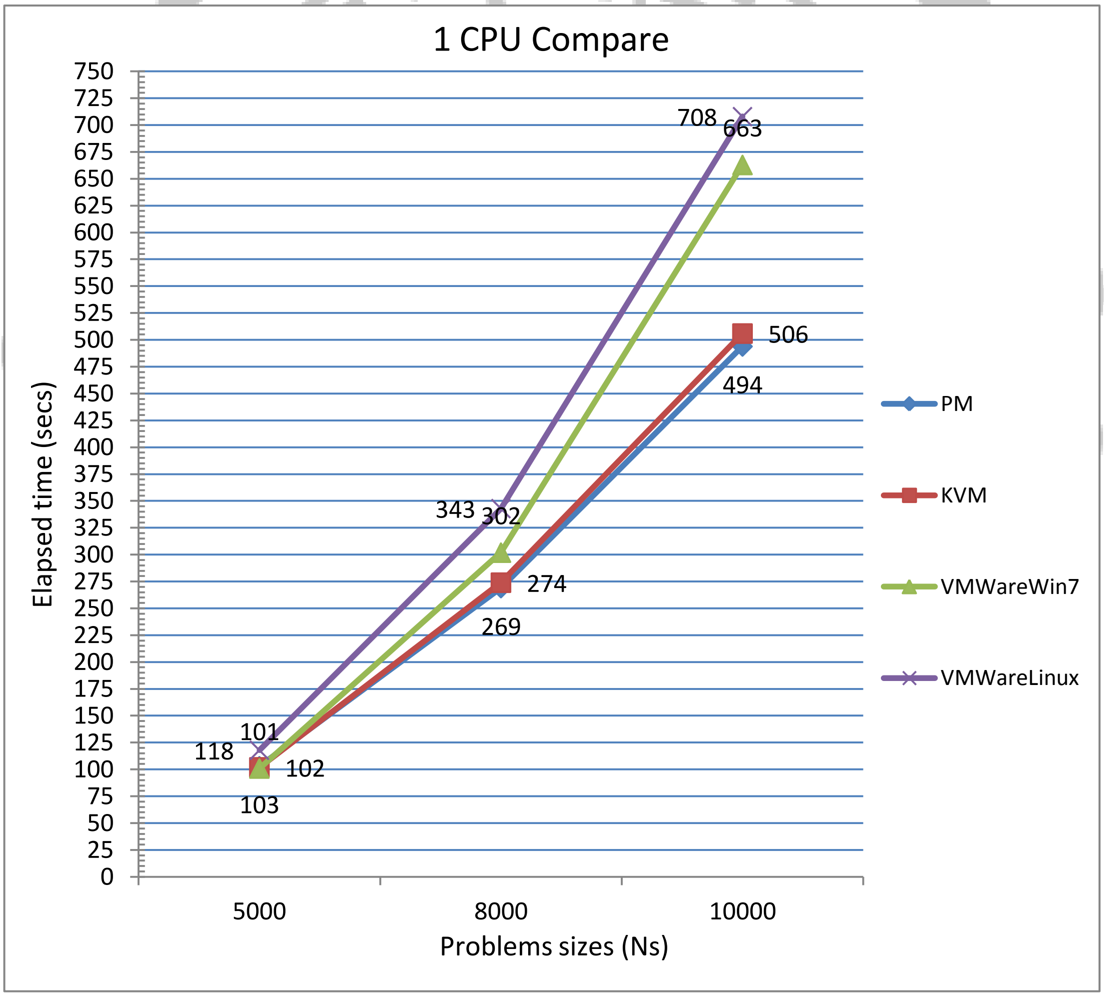

Figure 4-8. The performance results of HPCC by one CPU in different platform.

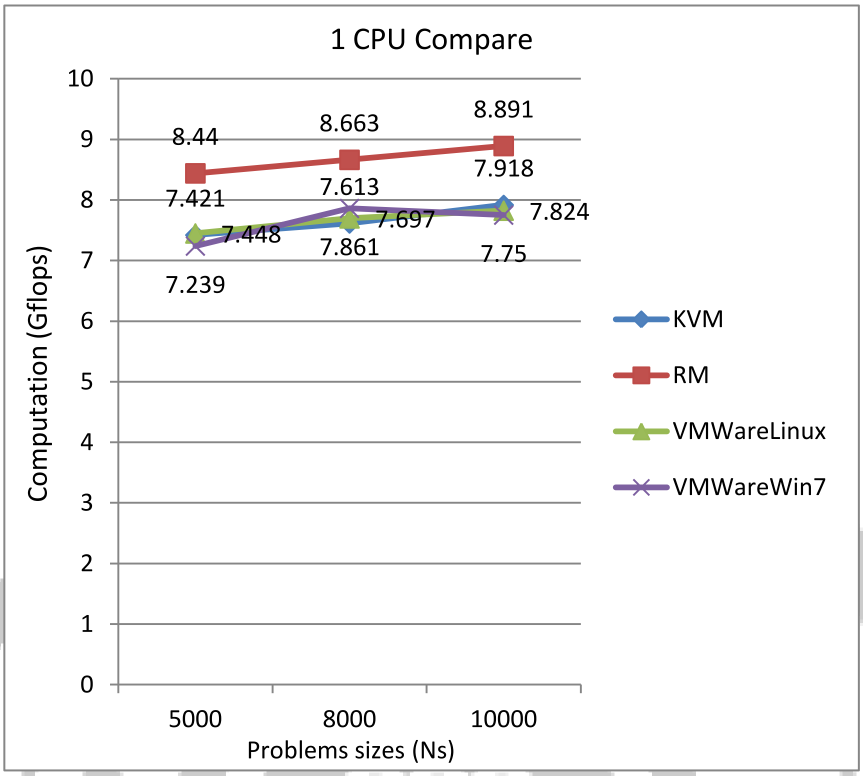

Figure 4-9. The computation results of HPCC by one CPU is in different platforms.

As shown in Figure 4-10, we compare dual cores to single core; the performance of dual cores on each platform has more enhancements. However, the order is the similar but with a slight different, as PM, KVM and VMware in Win7 (approximately equal to), VMware in Linux. The gap time between the physical machine and KVM is larger than one CPU in the "Ns=10000" Level. In addition, the performance of VMware in Windows is better than VMware in Linux. With the significant gap, we are curious whether the VMware in Linux is not optimized or not. The gap of VMware with other two platforms is close to single core case. Another important issue is that the computation of VMware in Win7 is better than KVM and VMware in Linux. Perhaps windows 7 has the implementation of optimization for multi core CPUs.

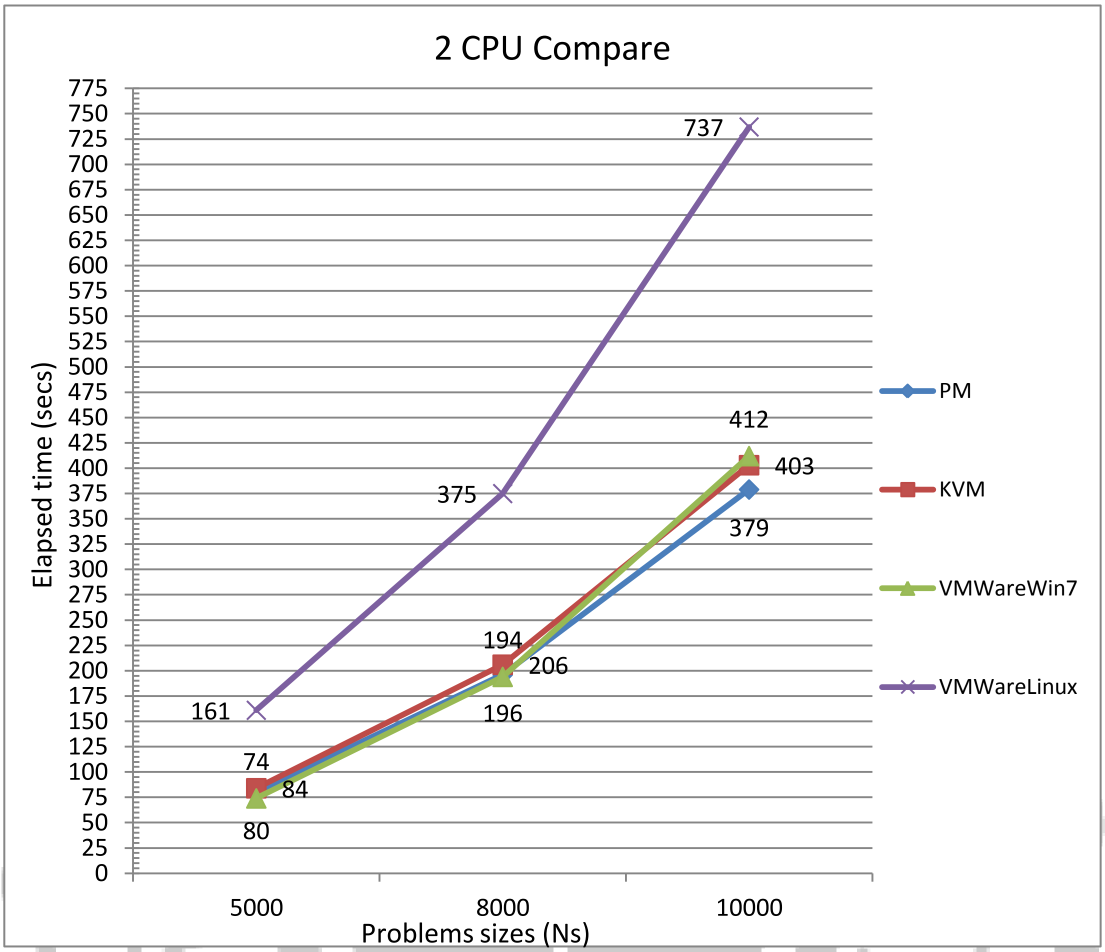

Figure 4-10. The performance results of HPCC by two CPUs in different platform.

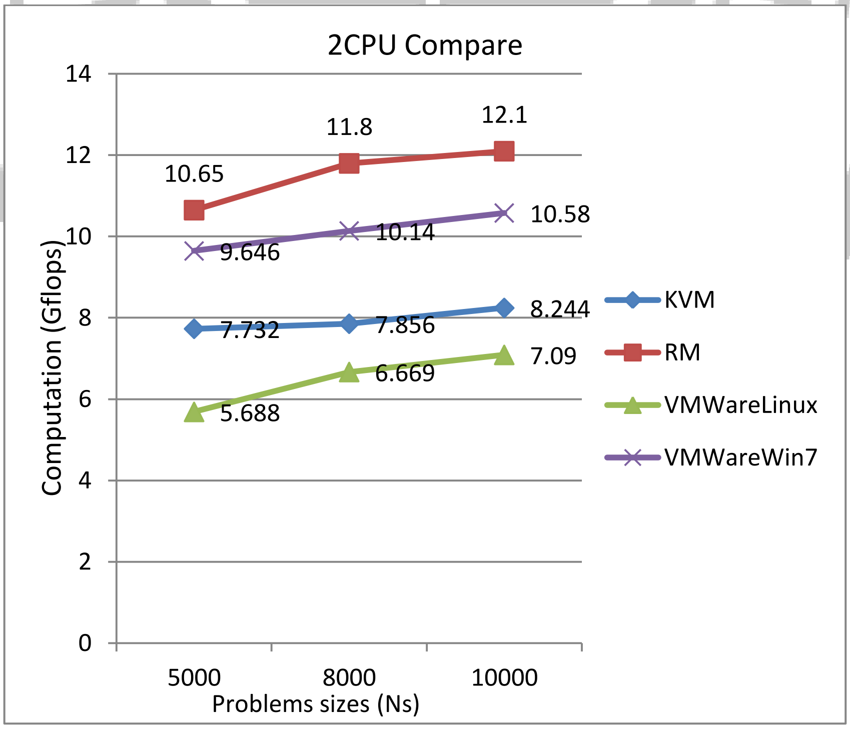

Figure 4-11. The computation results of HPCC by two CPUs in different platform.

4.2.4 Experiment 4: Sessions from WAN or LAN is a basis to judge about live migration.

Table 4-3. Environment Specification

Environment Spec.						
	CPU	Memory	Disk	Network	OS	Software
Node02	I7-990 3.47GHz	12GB	2TB	Gigabits	CentOS6 x64	OpenNebula-Node
Node03	I7-3960X 3.3GHz	16GB	6TB			
Node04	I7-2600 3.4GHZ	4GB	500GB			OpenNebula-Front End/Node

In this experiment, we test the live migration of VM during downloading. It is a normal situation in actual operation. We download a file in 2 situations. One is from internet and the other is intranet.

Situation1: Internet (http://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso).

```
root@HANN:~
[root@HANN ~]# wget ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
--2012-06-25 16:05:45--  ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
           => 'CentOS-6.2-i386-minimal.iso'
正在查找主機 ftp.twaren.net... 140.110.123.9, 2001:e10:5c00:5::9
正在連接 ftp.twaren.net|140.110.123.9|:21... 連上了。
以 anonymous 的身分登入... 登入完成！
==> SYST ... 完成。    ==> PWD ... 完成。
==> TYPE I ... 完成。  ==> CWD (1) /Linux/CentOS/6.2/isos/i386 ... 完成。
==> SIZE CentOS-6.2-i386-minimal.iso ... 298553344
==> PASV ... 完成。    ==> RETR CentOS-6.2-i386-minimal.iso ... 完成。

    [                   <=>                                         ] 298,553,344 6.17M/s   in 52s

2012-06-25 16:06:37 (5.52 MB/s) - 'CentOS-6.2-i386-minimal.iso' saved [298553344]

[root@HANN ~]#
```

Figure 4-12. Download a file from Internet

```
root@HANN:~
[root@HANN ~]# wget ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
--2012-06-25 16:10:06--  ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
           => 'CentOS-6.2-i386-minimal.iso'
正在查找主機 ftp.twaren.net... 140.110.123.9, 2001:e10:5c00:5::9
正在連接 ftp.twaren.net|140.110.123.9|:21... 連上了。
以 anonymous 的身分登入... 登入完成！
==> SYST ... 完成。    ==> PWD ... 完成。
==> TYPE I ... 完成。  ==> CWD (1) /Linux/CentOS/6.2/isos/i386 ... 完成。
==> SIZE CentOS-6.2-i386-minimal.iso ... 298553344
==> PASV ... 完成。    ==> RETR CentOS-6.2-i386-minimal.iso ... 完成。

    [       <=>                                                     ] 7,470,232    5.02M/s
```

Figure 4-13. Download a file from Internet with live migration

```
root@NODE04 ~]# onehost list
  ID NAME                     RVM    TCPU    FCPU    ACPU    TMEM    FMEM    AMEM   STAT
   0 Node02                     1    1200    1197    1100   11.8G   10.7G   10.8G     on
   1 Node03                     3    1200    1197     900   15.7G   13.5G   12.7G     on
   2 Node04                     0     800     796     800    3.6G    3.1G    3.6G     on
root@NODE04 ~]# onevm list
    ID USER     GROUP    NAME         STAT CPU     MEM        HOSTNAME        TIME
     5 oneadmin oneadmin HADN01       runn   0   1024M          Node03 15 19:28:43
     6 oneadmin oneadmin HADN02       runn   0   1024M          Node03 15 19:19:34
     7 oneadmin oneadmin MIFAS        runn   0   1024M          Node02 15 17:28:33
     8 oneadmin oneadmin HANN         runn   0   1024M          Node03 15 17:26:47
root@NODE04 ~]# onevm livemigrate 8 0
root@NODE04 ~]# onevm list
    ID USER     GROUP    NAME         STAT CPU     MEM        HOSTNAME        TIME
     5 oneadmin oneadmin HADN01       runn   0   1024M          Node03 15 19:31:17
     6 oneadmin oneadmin HADN02       runn   0   1024M          Node03 15 19:22:08
     7 oneadmin oneadmin MIFAS        runn   0   1024M          Node02 15 17:31:07
     8 oneadmin oneadmin HANN         migr   0   1024M          Node02 15 17:29:21
root@NODE04 ~]# onevm list
    ID USER     GROUP    NAME         STAT CPU     MEM        HOSTNAME        TIME
     5 oneadmin oneadmin HADN01       runn   0   1024M          Node03 15 19:31:59
     6 oneadmin oneadmin HADN02       runn   0   1024M          Node03 15 19:22:50
     7 oneadmin oneadmin MIFAS        runn   0   1024M          Node02 15 17:31:49
     8 oneadmin oneadmin HANN         runn   0   1024M          Node02 15 17:30:03
```

Figure 4-14. Live migration is success

```
root@HANN:~
[root@HANN ~]# wget ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
--2012-06-25 16:11:31--  ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
           => 'CentOS-6.2-i386-minimal.iso'
正在查找主機 ftp.twaren.net... 140.110.123.9, 2001:e10:5c00:5::9
正在連接 ftp.twaren.net|140.110.123.9|:21... 連上了。
以 anonymous 的身分登入... 登入完成！
==> SYST ... 完成。    ==> PWD ... 完成。
==> TYPE I ... 完成。  ==> CWD (1) /Linux/CentOS/6.2/isos/i386 ... 完成。
==> SIZE CentOS-6.2-i386-minimal.iso ... 298553344
==> PASV ... 完成。    ==> RETR CentOS-6.2-i386-minimal.iso ... 完成。

    [                        <=>                                  ] 60,600,248    747K/s
```

Figure 4-15. When live migration beginning, the transfer rate is down.

```
root@HANN:~
[root@HANN ~]# wget ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
--2012-06-25 16:11:31--  ftp://ftp.twaren.net/Linux/CentOS/6.2/isos/i386/CentOS-6.2-i386-minimal.iso
           => 'CentOS-6.2-i386-minimal.iso'
正在查找主機 ftp.twaren.net... 140.110.123.9, 2001:e10:5c00:5::9
正在連接 ftp.twaren.net|140.110.123.9|:21... 連上了。
以 anonymous 的身分登入... 登入完成！
==> SYST ... 完成。    ==> PWD ... 完成。
==> TYPE I ... 完成。  ==> CWD (1) /Linux/CentOS/6.2/isos/i386 ... 完成。
==> SIZE CentOS-6.2-i386-minimal.iso ... 298553344
==> PASV ... 完成。    ==> RETR CentOS-6.2-i386-minimal.iso ... 完成。

    [                                         <=>               ] 298,553,344 4.27M/s   in 1m 45s

2012-06-25 16:13:16 (2.72 MB/s) - 'CentOS-6.2-i386-minimal.iso' saved [298553344]

[root@HANN ~]#
```

Figure 4-16. With live migration, it takes more time to download.

Situation2: Intranet

```
root@HANN:~
[root@HANN ~]# scp root@192.168.1.232:~/CentOS-6.2-i386-LiveCD.iso .
root@192.168.1.232's password:
CentOS-6.2-i386-LiveCD.iso                                        100%  696MB  11.4MB/s   01:01
[root@HANN ~]#
```

Figure 4-17. Download a file from Intranet(CentOS-6.2-i386-LiveCD.iso)

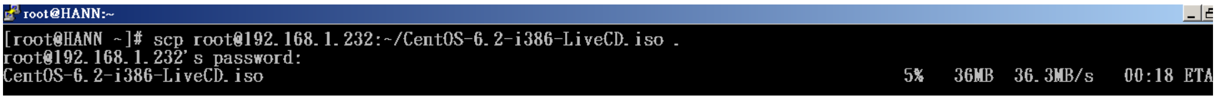

Figure 4-18. Download a file from Intranet with live migration

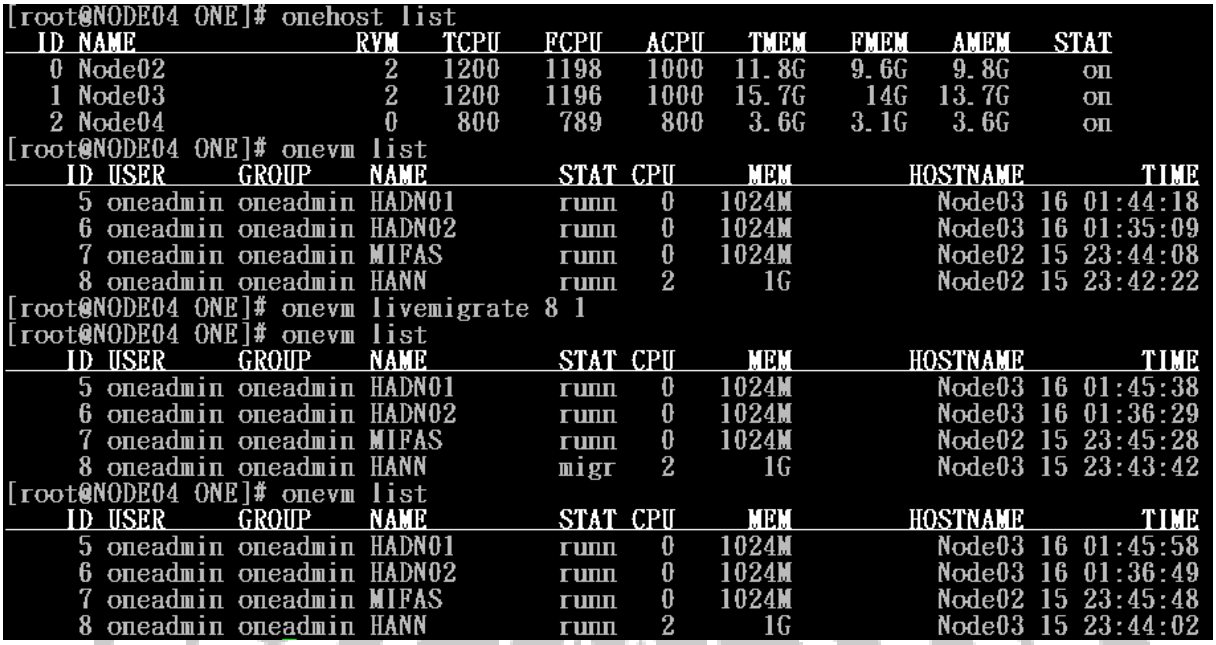

Figure 4-19. Live migration is success

Figure 4-20. When live migration beginning, the transfer rate is down.

Figure 4-21. With live migration, it take more time to download.

We can get the results as the table 4-4.

Table 4-4. The two situations results.

	Internet(285MB)	Intranet(696MB)
Download	52s	61s
Download with live migration	105s	81s

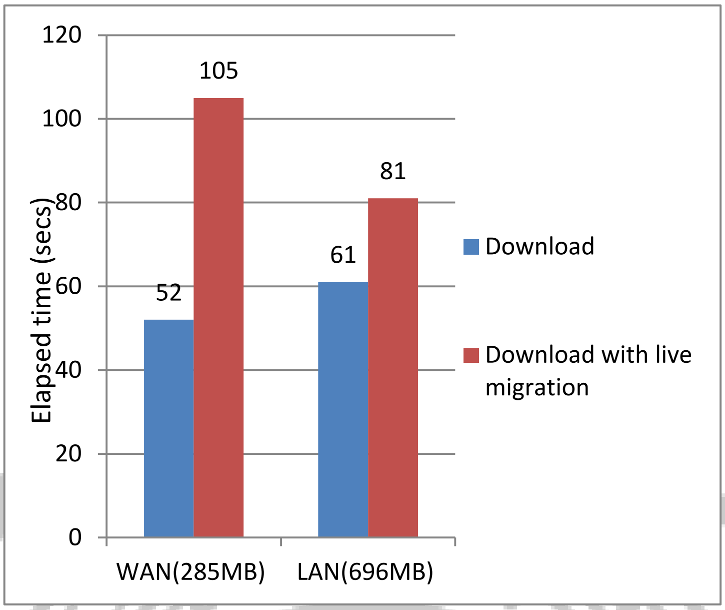

Figure 4-22. The compare between WAN(Internet) and LAN(Intranet).

4.3 Discussions

With points of view in virtualization manger, we want to know the application and features of live migration. Then, we later discuss the four different experiments in four parts.

- In the experiment 1, we understand "Non-stop Service during the live migration." is a commercial advertisement. Although in the experiment 1, the action of ping during the live migration is a success, it is not smooth to collect data from the reply time. We can find obvious crest in the Figure 4-6. Exchanging the physical machine of VM occurs.
- We try different sizes of OS memory to test the length of live migration in the experiment 2; however, we found time during live migration in different memory sizes are almost the same. The real subject in live migration is "dirty pages" and non-allocated memory does not need to be sent. When we begin initial deployment of VM, the used memory of all tested VMs should be the same because of

experimental results.

- We compare the performance of physical machine and VM on Linux and also compare the performance of full virtualization product on Linux and Windows. First we found the performance of full virtualization with KVM technology (hardware-assisted virtualization) is close to physical machine. It helps the popularity of virtualization. In addition, we test the performance of KVM and VMware. They are full virtualization symbol of open source and business. The results tell us that the performance of KVM is better than VMware in Windows7 or Linux. In the experiment, we prove the open source solution is another excellent choice for enterprise or organization.
- The results of downloading file with live migration in different network situations are as below. Although 2 tasks are completed, the result of intranet is better than internet. We receive an important message that the connections of VM from WAN or LAN is another flag when we begin the VM live migration. We believe that if there are 2 VMs in the same loading on the host, the VM with less connection from WAN get higher priority to live migration. With the better hardware (ex: 10 G Network), the connections influence of live migration in LAN are small and smooth.

Chapter 5
Conclusions and Future Work

5.1 Conclusions

In this thesis, we successfully provide private cloud solutions in open source and examine many features of live migration. We explore many detailed tests of live migration in this thesis. It helps us to do live migration smoothly in daily operation. Live migration is an integral part of the virtualization in organization (enterprise).

We test the performance between physical machine and the VM with KVM. We obtain results that performance of KVM is close to physical machine. Performance of full virtualization is improved by hardware-assisted virtualization. Although there is still a gap from the best performance, the final results were very satisfactory.

On the other hand, we build the MIFAS operated on the VMs in our environment. It represents the implementation of three Cloud service models and proves the feasibility of open source technology in cloud computing. The system can be viewed as a successful private cloud solution.

5.2 Future Work

Although the down time of live migration is reduced to the millisecond level, it is interesting to know that the down time cause different kinds of impacts in actual work such as lots of insert action in database or high traffic on the network. In addition, the management of accounts and groups lack of OpenNebula platform because the system is not designed to test big scale account management. However, if it is used in large organization, the account management should be re-designed.

Another important issue is the interchange of hybrid cloud. There are strong and flexible adaptability for OpenNebula to operate VM with varied virtualization. The adaptability of OpenNebula can accept many different VMs from other clouds so the implementation of VM migration between different clouds is another necessary project for the future.

Bibliography

[1] Arun Babu Nagarajan , Frank Mueller , Christian Engelmann , Stephen L. Scott, “Proactive fault tolerance for HPC with Xen virtualization”, Proceedings of the 21st annual international conference on Supercomputing, pp. 23-32, June 17-21, 2007, Seattle, Washington.

[2] Binbin Zhang, Xiaolin Wang, Rongfeng Lai, Liang Yang, Zhenlin Wang, Yingwei Luo, and Xiaoming Li, “Evaluating and Optimizing I/O Virtualization in Kernel-based Virtual Machine(KVM)”, NPC'10 Proceedings of the 2010 IFIP international conference on Network and parallel computing, pp. 220-231, September 13-15, 2010.

[3] Chris Matthews, Yvonne Coady, “Virtualized Recomposition: Cloudy or Clear?” ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 38-44, May 23 2009.

[4] Chien-Hsiang Tseng Chao-Tung Yang, Keng-Yi Chou and Shyh-Chang Tsaur, "Design and Implementation of a Virtualized Cluster Computing Environment on Xen,” presented at the The second International Conference on High Performance Computing and Applications, HPCA, 2009.

[5] C. A. Waldspurger, “Memory Resource Management in VMware ESX Server,” SIGOPS Oper. Rev., vol. 36, no. SI, pp. 181-194, 2002.

[6] F. Bellard, “Qemu, a fast and portable dynamic translator,” in Proceedings of the USENIX 2005 Annual Technical Conference, FREENIX Track, pp. 41-41, 2005.

[7] Gabor Kecskemeti, Gabor Terstyanszky, Peter Kacsuk, Zsolt Nemetha, “An approach for virtual appliance distribution for service deployment,” Future Generation Computer Systems, Volume 27 Issue 3, pp. 280-289, March, 2011.

[8] H. Raj and K. Schwan, “High Performance and Scalable I/O Virtualization via Self-Virtualized Devices,” in the proceedings of HPDC 2007, pp. 179-188, 2007.

[9] Hien Nguyen Van, Fr´ed´eric Dang Tran, Jean-Marc Menaud, “Autonomic virtual resource management for service hosting platforms,” ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 1-8, May 23 2009.

[10] Hitoshi Oi and Fumio Nakajima, “Performance Analysis of Large Receive Offload in a Xen Virtualized System,” in Proceedings of 2009 International Conference on Computer Engineering and Technology (ICCET 2009), Vol. 1, pp475–480, Singapore, January 2009.

[11] J. E. Smith and R. Nair, “The Architecture of Virtual Machines,” Computer, vol. 38, no.5, pp. 32-38, 2005.

[12] J. S. Paul Willmann, David Carr, Aravind Menon, Scott Rixner, Alan L. Cox and Willy Zwaenepoel, "Concurrent Direct Network Access for Virtual Machine Monitors, "The second International Conference on High Performance Computing and Applications, HPCA, pp. 306-317, 2007.

[13] Kertesz. A., Kacsuk. P., “Grid Interoperability Solutions in Grid Resource Management,” Systems Journal, IEEE, Volume 3, Issue:1, pp.131-141, March 2009.

[14] K. Adams and O. Agesen, “A Comparison of Software and Hardware Techniques for x86 Virtualization,” in ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems. New York, NY, USA: ACM Press, pp. 2–13, 2006.

[15] Luis Rodero-Merino, Luis M. Vaquero, Victor Gil, Fermín Galán, Javier Fontán, Rubén S. Montero, Ignacio M. Llorente, “From infrastructure delivery to service management in clouds,” Future Generation Computer Systems, Volume 26 Issue 8, pp. 1226-1240, October, 2010.

[16] Milojičić, Dejan, Llorente, Ignacio M., Montero, Ruben S., “OpenNebula: A Cloud Management Tool”, Internet Computing, IEEE, volume 15, issue 2, pp. 11-14, 2011.

[17] Michael R. Hines, Umesh Deshpande, and Kartik Gopalan, "Post-Copy Live Migration of Virtual Machines", Computer Science, Binghamton University (SUNY)

[18] Patrícia Takako Endo, Glauco Estácio Gonçalves, Judith Kelner, Djamel Sadok, “A Survey on Open-source Cloud Computing Solutions”, VIII Workshop em Clouds, Grids e Aplicações, pp.3-16, 2011.

[19] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris,A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization,” in SOSP ’03: Proceedings of the nineteenth ACM symposium on Operating systems principles. New York, NY, USA: ACM Press, pp. 164–177, 2003.

[20] Qumranet, “White Paper: KVM Kernel-based Virtualization Driver,” Qumranet, Tech. Rep.,2006.

[21] R. S. M. Borja Sotomayor, Ignacio M. Llorente, Ian Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds,” IEEE Internet Computing, vol. 13, pp. 16-23, 2009.

[22] S. Soltesz, H. Potzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, “Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors,” in EuroSys 2007, pp. 275-287, 2007.

[23] W. v. Hagen, Professional Xen Virtualization: Wrox Press Ltd. Birmingham, UK, UK, 2008.

[24] W. Emeneker and D. Stanzione, “HPC Cluster Readiness of Xen and User Mode Linux,” in 2006 IEEE International Conference on Cluster Computing, pp. 1-8, 2006.

[25] Yajun Li, Yuhang Yang, Naode Ma, Liang Zhou, “A hybrid load balancing strategy of sequential tasks for grid computing environments,” Future Generation Computer Systems 2009, pp. 819-828, 2009.

[26] Xiantao Zhang, Yaozu Dong, "Optimizing Xen VMM Based on Intel Virtualization Technology," 2008 International Conference on Internet Computing in Science and Engineering (ICICSE 2008), pp.367-374, 2008.

[27] Yaozu Dong, Shaofan Li, Asit Mallick, Jun Nakajima, Kun Tian, Xuefei Xu, Fred Yang, and Wilfred Yu, “Extending Xen with Intel Virtualization Technology,” Journal, ISSN, Core Software Division, Intel Corporation, pp. 1-14, August 10, 2006.

[28] Z. Hai, et al., "An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems," in ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual, pp. 124-129, 2010.

[29] Amazon, http://aws.amazon.com/ec2/

[30] Cloud computing, http://en.wikipedia.org/wiki/Cloud_computing

[31] OpenStack, http://openstack.org

[32] OpenNebula, http://www.opennebula.org

[33] KVM, http://www.linux-kvm.org/page/Main_Page

[34] Xen, http://www.xen.org/

[35] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny,Eric July, Christian Limpach, Ian Pratt, Andrew Warfield, "Live Migration of Virtual Machines", University of Cambridge Computer Laboratory y Department of Computer Science 15 JJ Thomson Avenue, Cambridge, UK University of Copenhagen, Denmark, 2005.

[36] Hsuan-Hao Wang, ”Improving the Live migration of Virtual Machines with Hybrid Copy

Architecture ", TATUNG UNIVERSITY, 2012.

Appendix – Installation Guide

- Node Define

vi /etc/hosts

192.168.1.232 Node02

192.168.1.231 Node01

192.168.1.233 Node03

192.168.1.234 Node04

- Bridge Setup of Network

EX: ifcfg-eth0/ ifcfg-br0

DEVICE="eth0"

NM_CONTROLLED="yes"

ONBOOT=yes

TYPE=Ethernet

BRIDGE="br0"

BOOTPROTO=dhcp

DEFROUTE=yes

IPV4_FAILURE_FATAL=yes

IPV6INIT=no

NAME="System eth0"

UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03

HWADDR=54:04:A6:DC:45:3D

PEERDNS=yes

PEERROUTES=yes

br0

DEVICE="br0"

ONBOOT=yes

TYPE=Bridge

BOOTPROTO=dhcp

● SELinux Setup

vi /etc/sysconfig/selinux

SELINUX=disabled

1. KVM Install

(1). yum -y install glibc glibc-common glibcdevel cpp glibc-headers kernel-headers libgomp libstdc++-devel nscd gcc-c++ rpm-build yum-utils pkgconfig

(2). yum -y install libxml2 libxml2-devel expat expat-devel libxslt libxslt-devel openssl openssl-devel curl curl-devel

(3). yum -y install ruby ruby-libs ruby-devel ruby-irb ruby-docs ruby-rdoc ruby-ri

(4). yum install kvm virt-* libvirt

(5). # reboot

(6). ln -s /usr/libexec/qemu-kvm /usr/bin/kvm

2. SSH Setup

(1). #ssh-keygen -t rsa

(2). #ssh-copy-id –I ~/.ssh/id_rsa.pub root@RemoteHost

3. NFS Install & Setup

yum install nfs-utils

- **FrontEnd:**

#vi /etc/exports

/var/lib/one Node02(rw,fsid=0,sync,no_root_squash,no_subtree_check)

exportfs -arv

- **Node:**

mount –t nfs node01:/var/lib/one /var/lib/one

4. OpenNebula Install(some RPMs need to download and only operated for

FireFox)

(1). rpm -ivh http://download.fedora.redhat.com/pub/epel/6/i386/epel-release-6-5.noarch.rpm

(2). yum install rubygems

(3). rpm -ivh xmlrpc-c-c++-1.16.24-1200.1840.el6.x8_64_3.rpm

(4). rpm -ivh xmlrpc-c-client++-1.16.24-1200.1840.el6.x86_64.rpm

(5). rpm -ivh opennebula-3.0.0-1.x86_64.rpm

(6). mkdir ~/.one

(7). echo "oneadmin:password" > ~/.one/one_auth

5. The ID of all nodes must be the same with FrontEnd's oneadmin

FrontEnd: (get ID)

id oneadmin (check oneadmin's ID)

uid=1001(oneadmin) gid=1001(cloud) groups=1001(cloud)

NODE: (The same ID)

groupadd --gid 1001 oneadmin

useradd --uid 1001 -g oneadmin -d /var/lib/one oneadmin

6. OpenNebula Environment Setup

vi /etc/one/oned.conf

1. VM_DIR=/var/lib/one

2.remove #

TM_MAD = [

name = "tm_ssh",

executable = "one_tm",

arguments = "tm_ssh/tm_ssh.conf"]

7. ONEHOST Setup

onehost create Node01 im_kvm vmm_kvm tm_ssh

onehost create Node02 im_kvm vmm_kvm tm_ssh

onehost create Node03 im_kvm vmm_kvm tm_ssh

onehost create Node04 im_kvm vmm_kvm tm_ssh

8. ONEVNET Setup

- EX:: PCloud.net

vi PCloud.net

NAME = "PCloud“

TYPE = RANGED

BRIDGE = br0

NETWORK_SIZE=C

NETWORK_ADDRESS =192.168.1.0

GATEWAY=192.168.1.1

#onevnet create PCloud

9. ONEVM Setup

- EX: xp01.one:

vi xp01.one

NAME = WinXP01

CPU = 1

MEMORY = 512

OS = [boot = hd]

DISK = [source = "/var/lib/one/images/XP01.img",

clone = "no",

target = "hda",

readonly = "no"]

GRAPHICS = [type ="vnc",

listen ="0.0.0.0",

port = "5911"]

FEATURES = [acpi="yes"]

NIC = [NETWORK_ID=0]

#onevm create xp01.one

10. SunStone Install & Setup

 # gem install json

 # gem install rack

 # gem install sinatra

 # gem install Thin

- **Upgrade Ruby (Install Ruby 1.92)**
- # wget ftp://ftp.ruby-lang.org//pub/ruby/1.9/ruby-1.9.2-p0.tar.gz
- # tar -zxvf ruby-1.9.2-p0.tar.gz
- # cd ruby-1.9.2-p0
- # ./configure
- # make && make install
- # export PATH=/usr/local/bin:$PATH
- # ruby –v (Confirm ruby version)

gem install sinatra

cd /usr/share/one

oneacctd start

#sunstone-server start