私立東海大學

資訊工程學系研究所

碩士論文

指導教授：楊朝棟　博士

在雲端環境中提供一個動態資源分配模型管理虛擬機器

A Dynamic Resource Allocation Model for Virtual Machine Management on Cloud

研 究 生：鄭翔耀

中華民國一〇〇年七月

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 鄭 翔 耀 所提之論文

在雲端環境中提供一個動態資源分配模型管理虛擬機器

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	楊武	簽章
委 員	時文中	
	[illegible]	
	[illegible]	
指 導 教 授	楊朝棟	簽章

中華民國 100 年 6 月 27 日

摘要

現今的 x86 硬體大多設計為執行單一作業系統與執行單一或多個應用程式，當該服務硬體資源占用太低，就顯得整體資源利用率不高。虛擬化技術可以在一台實體機器上運行多台虛擬機器，每台虛擬機器可在一台實體機上共享資源。要達到多台實體機器的管理、虛擬機器在實體機器中的不中斷遷移都是目前研究的議題。其中，如何確保多台虛擬機器在多台實體機器上的負載可達到平衡點，則是我們所要面臨及探討的。本研究中，我們基於虛擬化技術現有的線上遷移的方式，設計了一套方法為動態資源分配的方式，達到虛擬機器在實體機器上的負載平衡。本文裡設計了一個虛擬化平台的管理系統，並將本方法實際運用在此管理系統，結果證實了虛擬機器在負載變高時，可以透過線上遷移服務，在不停止的狀態下，將虛擬機器搬移置不同的實體機器，達到負載平衡。

關鍵字：虛擬化、線上遷移、動態資源分配、雲端運算

Abstract

Today's x86 computer hardware was designed to run a single operating system and a single application, leaving most machines vastly underutilized. Virtualization Technology lets you run multiple virtual machines on a single physical machine, with each virtual machine sharing the resources of that one physical computer across multiple environments. It is current issue to achieve the goal of management multiple virtualization platform and multiple virtual machine migration across physical machine without disruption. We have to face and discuss that ensure load balance when multiple virtual machine run on multiple physical machine. In this thesis we present a system which is implementation of optimization with Dynamic Resource Allocation(DRA) dealing with virtualization machines on physical machines. And practice DRA method in this system. The results confirmed that the virtual machine which loading becomes too high, it will automatically migrated to another low loading physical machine without service interrupting. And let total physical machine loading reaching balance.

Keywords- Virtualization, Live Migration, Dynamic Resource Allocation (DRA), Cloud Computing

Acknowledgements

回想當初推薦甄試進入東海資工研究所時，自己對要研究哪種議題都尚未沒有很明確的方向，入學後隨著系上安排的課程進行學習，當年雲端運算的議題正漸漸的明朗化，故自己就決定要針對此技術做進一步探討，自行找了楊朝棟老師，表明希望能加入他的團隊，很榮幸的獲得楊老師的首肯，才能進入高效能計算實驗室，與同學進行互相討論及研究。

學業即將完成，首先我要感謝指導教授－楊朝棟 博士，在學期間給予我明確的指導方向與支持鼓勵，無論是參與國際研討會的論文發表或者是在課堂之中及研究的過程或學習的過程中都充滿挑戰且富有成就感，我很榮幸能成為楊老師實驗室的一員，在東海高效能計算實驗室培養之下，造就出的抗壓性及知識，以應付未來更多的挑戰。

接著我也要感謝研究所合作的學長、同學們威利、政達、冠傑、智霖等，在研究所的兩年內提供協助，還要特別感謝公司同事這陣子的體諒與支持我將無法順利完成論文，沒有你們無私的付出，更遑論畢業取得碩士學位。

最後，我要感謝我的家人，沒有家人的支持與體諒，我將不可能完成我的學業，謝謝你們大家，並且再一次深深感謝楊老師的指導。

Tables of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Research Background

A virtual machine was originally defined by Popek and Goldberg as —an efficient, isolated duplicate of a real machine. A virtual machine provides the interface identical to underlying bare hardware, i.e., all devices, interrupts, memory, page tables, etc.[1].

Virtual machines are separated into two major categories, based on their use and degree of correspondence to any real machine. A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS). In contrast, a process virtual machine is designed to run a single program, which means that it supports a single process. An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual world.

Virtual machine operates in a virtual platform can be considered an independent operating system. In addition, due to the independence of this individual, the gust operating system can operate from the original virtual platform, and to maintain the operation of the original schedule. And this move action, generally referred to as Migration [2-23]. The migration is dynamic: if a move action, cause the system to pause time is extremely small enough, it cannot be aware of user using the system services.

1.2 Motivation and purpose

Setting up virtual machine cluster environment on physical machine can provide stable service, but this environment often includes unpredictable workloads. Currently most systems of virtual machine are loading balanced statically, as the systems where the load changes dynamically over time run it is inevitable that some physical hosts with higher load, so for throughput and response time of a system to be maximized it is necessary for load to be distributed to each part of the system in proportion to their computing/IO capacity[3, 20, 24-27].

In this thesis, we develop an adaptive resource control system on a virtual machine cluster that dynamically adjusts the resource shares to individual tiers in order to meet DRA (Dynamic Resource Allocation) goals. There are three efforts in this thesis: (1) supporting DRA mechanism; (2) implementation OpenNebula management tool on a web-based interface; and (3) efficiently isolating the cluster workloads.

1.3 Thesis Organization

This thesis structure is as follows: the second chapter discusses related research, describes the development of virtual technology, the dynamic migration of relevant thesis and research reports, and current use of the operating system migration and production procedures related thesis checkpoint; third chapter represents thesis work out solutions to issues of content; fourth chapter performance evaluation, the use of the systems do the actual number of operation, record all aspects of operation efficiency, and make the analysis and discussion; the fifth chapter is the conclusion and future directions.

Chapter 2
Background Review

2.1 Virtualization

Virtualization is simply the logical separation of the request for some service from the physical resources that actually provide that service. In practical terms, virtualization provides the ability to run applications, operating systems, or system services in a logically distinct system environment that is independent of a specific physical computer system. Obviously, all of these have to be running 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 rather than physical ones, 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 resources of computing with other computers[4].

It's not a new technique, IBM had implemented on 360/67 and 370 on 60, 70 eras. Virtualization gets hardware to imitate much hardware via Virtual Machine Monitor, and each one of virtual machines can be seemed as a complete individual unit. For a virtual machine, there are memories, CPUs, unique complete hardware equipment, etc... It can run any operating systems, called Guest Os, and do not affect other virtual machines. In recent years, many research proposed the application of virtual machine technology [5-7] to solve many system problems. For examples, fault tolerance, migration, intrusion detection and debugging.

Virtual machine can divide into two categories by client service: system virtual machine and process virtual machine[28] in Figure 2-1. The difference is the range of virtualization. System virtual machine provides an integrated system platform where guest OS can operate self-reliant. Process virtual machine only served only single process. This thesis will use system virtual machine technology to investigate the issue of platform switching.

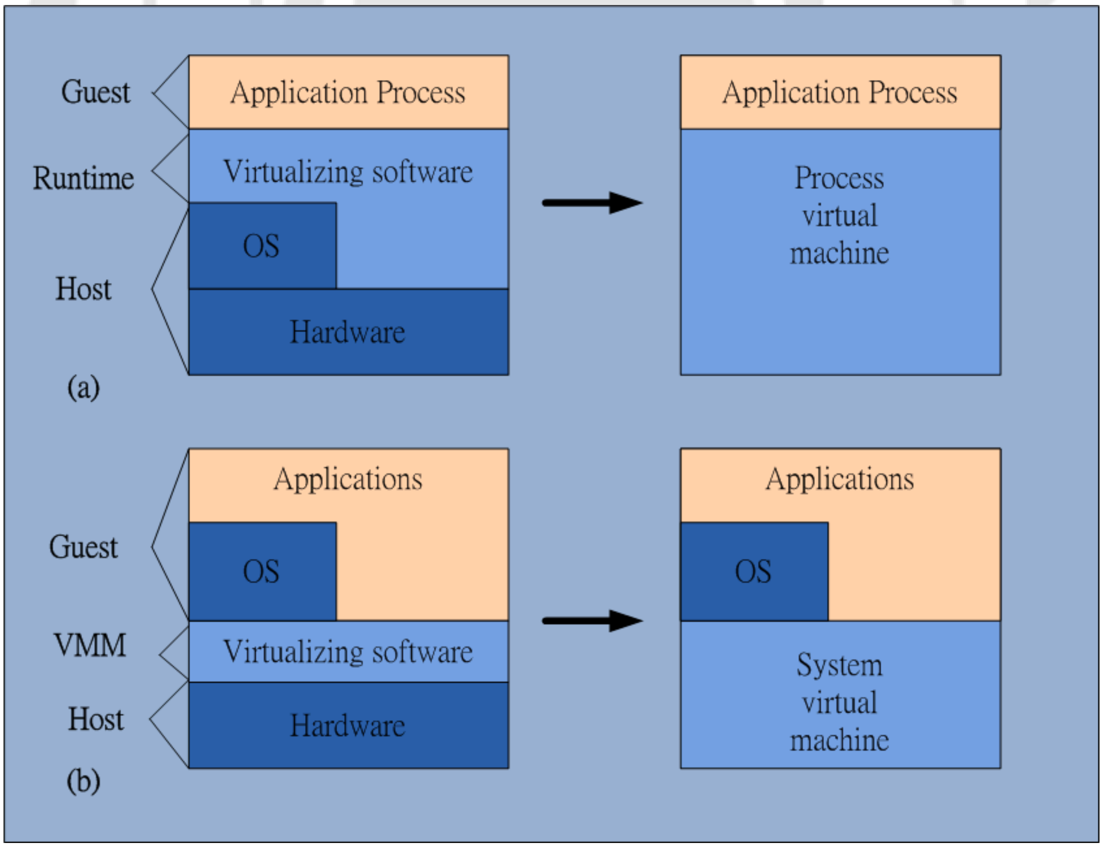


Figure 2-1. Process and System VMs

In general, most virtualization strategies fall into one of three major categories:

Full virtualization: it is similar to emulation. As in emulation, unmodified operating systems and applications run inside a virtual machine. Full virtualization differs from emulation in that operating systems and applications are designed to run on the same architecture as the underlying physical machine. This allows a full virtualization system to run many instructions directly on the 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 must use software to simulate a different basic architecture as shown in Figure 2-2.

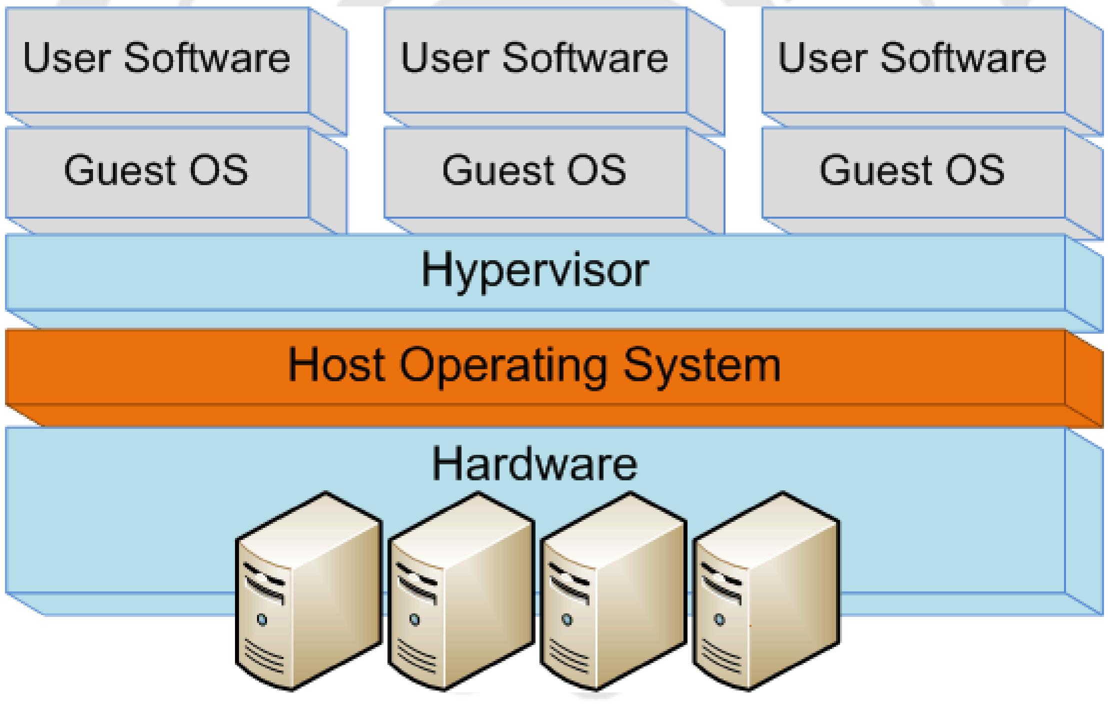


Figure 2-2. Full Virtualization

Hardware Assisted Virtualization: Recognizing the importance of virtualization, hardware vendors Intel and AMD have added extensions to the x86 architecture that make virtualization much easier[8]. While these extensions are incompatible with each other, they are essentially similar, consisting of

- A new guest operating mode: the processor can switch into a guest mode, which has all the regular privilege levels of the normal operating modes, except that system software can selectively request that certain instructions, or certain register accesses, be trapped. VMM(Virtual Machine Monitor) will make necessary processing.

- Hardware state switch: when switching to guest mode and back, the hardware switches the control registers that affect processor operation modes, as well as the segment registers that are difficult to switch, and the instruction pointer so that a control transfer can take effect.

- Exit reason reporting: when a switch from guest mode back to host mode occurs, the hardware reports the reason for the switch so that software can take the appropriate action.

Para-virtualization: In some instances this technique is also referred to as enlightenment. In Para-virtualization[9], VMM will provide Hypercall[10] to allow domains to perform a synchronous software trap into the hypervisor to perform a privileged operation, analogous to the use of system calls in conventional operating systems. The hypervisor exports a modified version of the underlying physical hardware. The exported virtual machine is of the same architecture, which is not necessarily the case in emulation. Instead, targeted modifications are introduced to 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 call application binary interface (ABI) instead of using certain architectural features that would normally be used. This means that only small changes are typically required in the guest operating systems, but any such changes make it difficult to support closed-source operating systems that are distributed in binary form only, such as Microsoft Windows. As in full virtualization, applications are typically still run without modifications.

Para-virtualization, like full virtualization, Para-virtualization also uses a hypervisor, and also uses the term virtual machine to refer to its virtualized operating systems. However, unlike full virtualization, Para-virtualization requires changes to the virtualized operating system. This allows the VM to coordinate with the hypervisor, and reduce the use of the privileged instructions that are typically responsible for the major performance penalties in full virtualization.

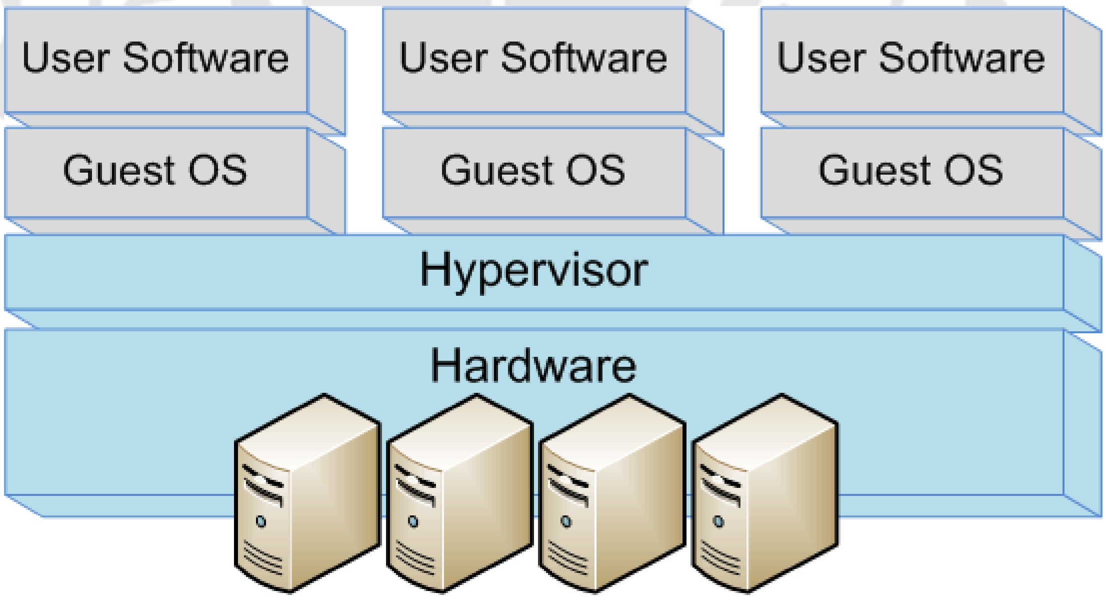


Figure 2-3. Para-virtualization

The advantage is that Para-virtualized virtual machines typically outperform fully virtualized virtual machines. The disadvantage, however, is the need to modify the Para-virtualized virtual machine or operating system to be hypervisor-aware. The framework of Para-virtualization is shown in Figure 2-3.

In order to evaluate the viability of the difference between virtualization and non-virtualization, the virtualization software we used in this thesis is Xen. Xen is a virtual machine monitor (hypervisor) that allows you to use one physical computer to run many virtual computers — for example, running a production Web server and a test server on the same physical machine or running Linux and Windows simultaneously. Although not the only virtualization system available, Xen has a combination of features that make it uniquely well suited for many important applications. Xen runs on commodity hardware platforms and is open source. Xen is fast, scalable, and provides server-class features such as live migration.

Xen is chosen to be our system's virtual machine monitor because it provides better efficiency, supports different operating system work simultaneously, and gives each operating system an independent system environment.

This free software is mainly divided into two kinds of simulate types, Para-virtualization and Full virtualization, as mentioned before. Para-virtualization implements virtualization technology, mostly via the modified kernel of Linux.

The characteristic of Para-virtualization is as follows:

- Virtual machine quite like real machine on operating efficacy
- At most supporting more than 32 cores of computer structures
- Supporting x86/32, with PAE technique and x86/64 hardware platform

- Good hardware driver support, almost for any Linux device driver

There are restricts with full virtualization, and it can be only executed when the hardware satisfy these conditions in the following:

- Intel VT technique (Virtualization Technology, Intel-VT)
- AMD SVM technique (Secure Virtual Machine, AMD-SVM or, AMD-V)

Besides, PAE is the Intel Physical Addressing Extensions technique, and this method enables 4 gigabytes physical memory of 32 bits hardware platform to support the platform that is only supported by 64 gigabytes memory. Then Xen could almost execute on all P-II or more high level hardware platform.

Currently, there are two major software series for platform virtualization: VMWare and Xen, each of them has its own advantages. VMWare is a commercial software series comprised of lots of products including vCloud, vSphere[11], VMWare ESX Server and so on. It has numerous functions and supports kinds of hardware and protocols excellently. There is also a commercial software series based on Xen called XenServer[12].

As a result of the widespread of virtual machine software in recently years, two best x86 CPU manufacturers Intel/AMD, with efficiency of x86 computers and increasing of compute core of CPU, both have published the new integrated virtualization on CPU, one for Intel Vander pool and another for AMD Pacifica. These technologies also support Xen, and make efficiency step up more than initial stages[1].

2.2 OpenNebula

Cloud computing systems fundamentally provide access to large pools of data and computational resources through a variety of interfaces similar in spirit to existing grid and HPC resource management and programming systems. These systems provide a wide variety of interfaces and abstractions ranging from the ability to dynamically provision entire virtual machine (i.e., Infrastructure-as-a-Service systems such as Amazon EC2 and others[29, 30]) to flexible access to hosted software services (i.e., Software-as-a-Service systems such as salesforce.com and others[24, 25, 31]). We have focused our efforts on the "lowest" layer of cloud computing systems (IaaS) because here we can provide a solid foundation on top of which language-, service-, and application-level cloud computing systems can be explored and developed.

In this work, we present OpenNebula: an open-source cloud computing framework. The OpenNebula is a virtual infrastructure engine that enables the dynamic deployment and re-allocation of virtual machines in a pool of physical resources. The OpenNebula system extends the benefits of virtualization platforms from a single physical resource to a pool of resources, decoupling the server, not only from the physical infrastructure but also from the physical location[4]. The OpenNebula contains one frontend and multiple backend. The front-end provides users with access interfaces and management functions. The back-ends are installed on Xen servers, where Xen hypervisors are started and virtual machines could be backed. Communications between frontend and backend employ SSH. The OpenNebula gives users a single access point to deploy virtual machines on a locally distributed infrastructure.

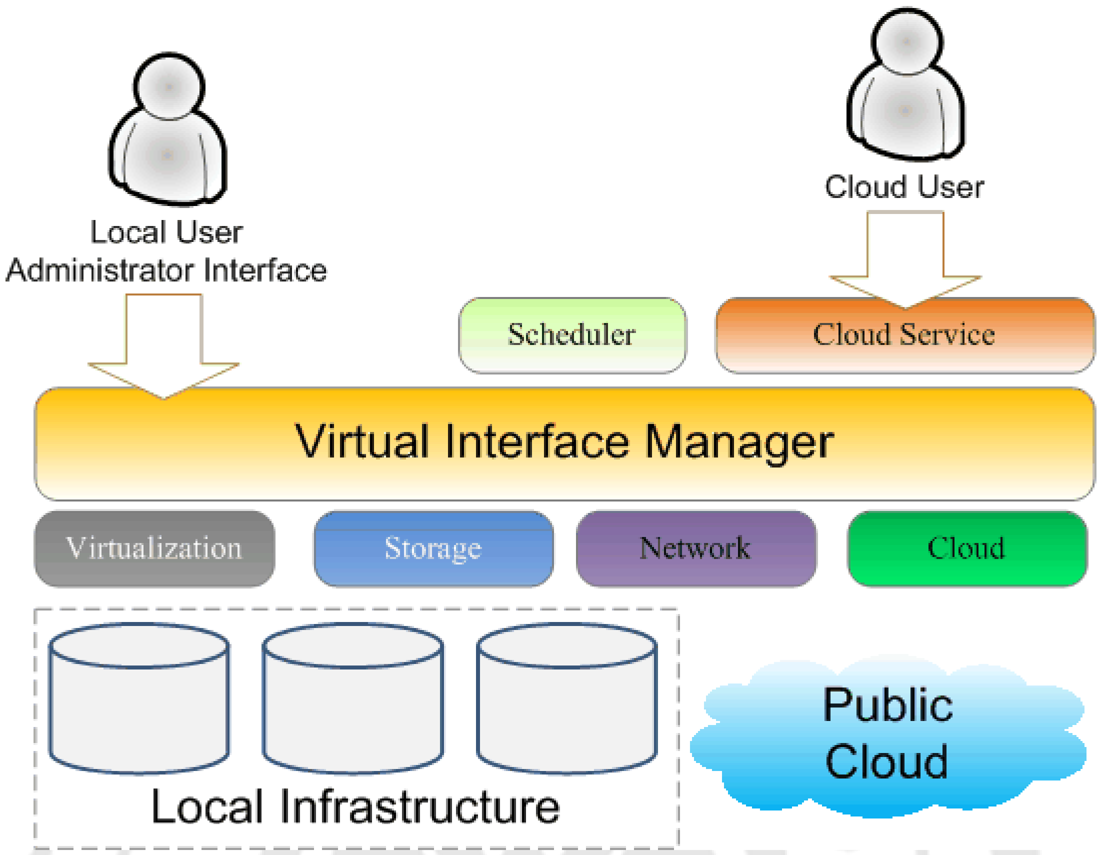


Figure 2-4. OpenNebula Architecture

OpenNebula orchestrates storage, network, virtualization, monitoring, and security technologies to enable the 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]. The architecture of OpenNebula can be described as Figure 2-4.

OpenNebula allow to use multiple storage back ends such as LVM(Logical volume manager)[32], iSCSI(Internet Small Computer System Interface)[33], and different hypervisors, such as VMware, Xen[10] and KVM(kernel-based virtual machine)[8, 13]. Also having a robust and thin core, which use C++ combined with a scripting driver plug-in back end, allows people to modify components to build their own plug-ins, for scheduling or VM placement, for example.

Eucalyptus [14] also belongs to virtual machine management platform. It is an open-source cloud-computing framework that uses computational and storage infrastructure commonly available to academic research group to provide a platform that is modular and open to experimental instrumentation and study[15]. The architecture of the Eucalyptus system is simple, flexible and modular with a hierarchical design reflecting common resource environments found in many academic settings. In essence, the system allows users to start, control, access, and terminate entire virtual machines using an emulation of Amazon EC2's SOAP and "Query" interfaces. That is, users of Eucalyptus interact with the system using the exact same tools and interfaces that they use to interact with Amazon EC2.

As Figure 2-5, Eucalyptus consist of three parts[15]:

- Node Controller: controls the execution, inspection, and terminating of virtual machine instances on the host where it runs.
- Cluster Controller: gathers information about and schedules virtual machine execution on specific node controllers, as well as manages virtual instance network.
- Storage Controller (Walrus): is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.
- 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 controllers.

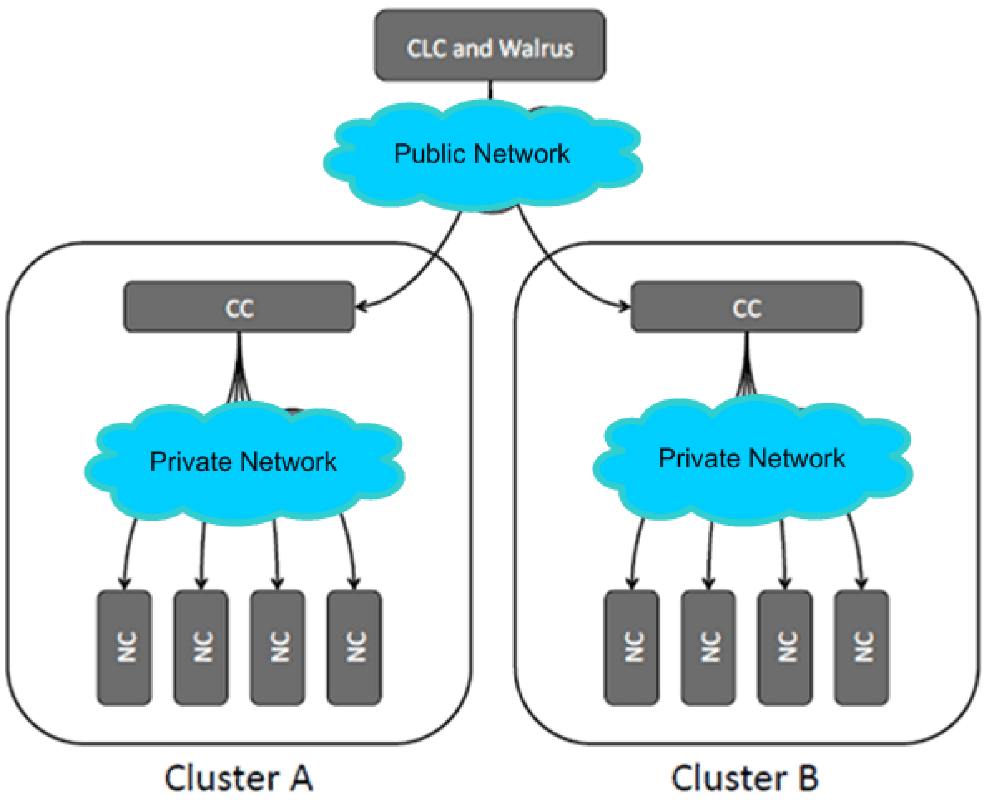


Figure 2-5. Eucalyptus Architecture

The Eucalyptus system is built to allow administrators and researchers the ability to deploy an infrastructure for user-controlled virtual machine creation and control atop existing resources. The system is highly modular, with each module represented by a well-defined API, enabling researchers to replace components for experimentation with new cloud computing solutions.

OpenNebula and Eucalyptus are major open-source cloud computing software platforms. The overall function of these systems is to manage the provisioning of virtual machines for a cloud providing Infrastructure-as-a-Service. These various open-source projects provide an important alternative for those who do not wish to use commercially provided cloud. In table 1 provide a table to compare and analyze these systems[16].

Table 2-1. Cloud Architectures Compared

	Eucalyptus	OpenNebula
Disk Image Options	Options set by admin	In private cloud, Most libvirt options left open
Disk Image Storage	Walrus, which imitates Amazons S3	A shared file system, by default NFS, or SCP
Hypervisors	Xen, KVM	Xen, KVM, VMware
Unique Feature	User management web interface	VM migration supported

However the OpenNebula lack a management tool. In this thesis we were not only implemented OpenNebula but also created a Web-based management system on it. Thus, the system administrator can be easy to monitor and manage the entire OpenNebula System on our project. We solved 3 main challenges: First, How to combine web and OpenNebula, Second, How to monitor virtualization system and physical machines state on the web and Finally, How to arrive at DRA.

2.3 Virtual Machine Migration

System migration technology is an important feature in virtualization. By virtual platform providing system migration technology, virtual guest operating system will be seemed an integrated migrated unit. Migrate an entire operating system and all of its applications as one unit allows us to avoid many of the difficulties faced by process-level migration approaches. In particular the narrow interface between a virtualized operating system and the virtual machine monitor makes it easy avoid the problem of ‘residual dependencies[17]’ in which the original host machine must remain available and network-accessible in order to service certain system calls or even memory accesses on behalf of migrated processes. As Figure 2-6 and Figure 2-7[18]. When migration finish, original system is no more need to provide software or hardware resources to destination system[2].

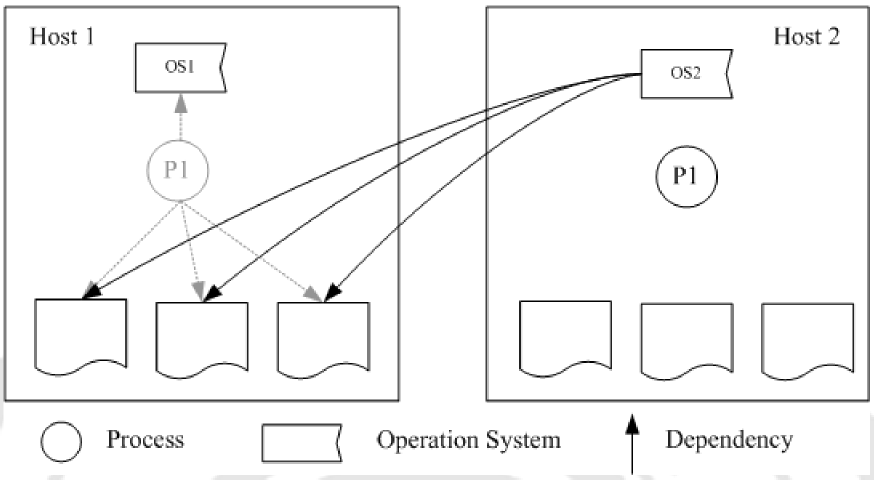


Figure 2-6. Process migration example. *p1* has migrated from *host1* to *host2*.

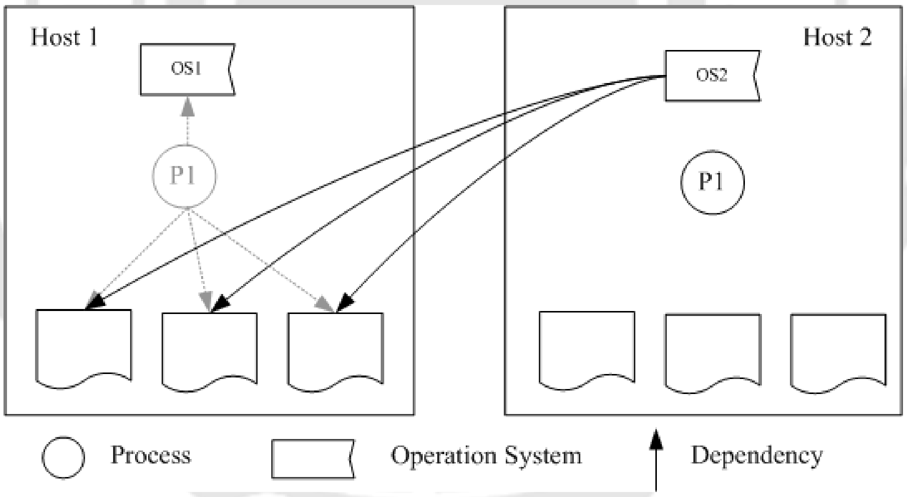


Figure 2-7. Migrating *p1* from host1 to host2 by means of operating system migration.

Migration situation could divided two categories: online migration and offline migration[34]. It represented a typical migration behavior could differentiate three stages[17]:

- Push phase: source virtual system processing and system content migration are in the same time.
- Stop-and-copy phase: source virtual system has already stopped processing and system content migration executes final migration. The end of migration, virtual system which migrated to destination begins to process.
- Pull phase: virtual system which migrated to destination has already operated,

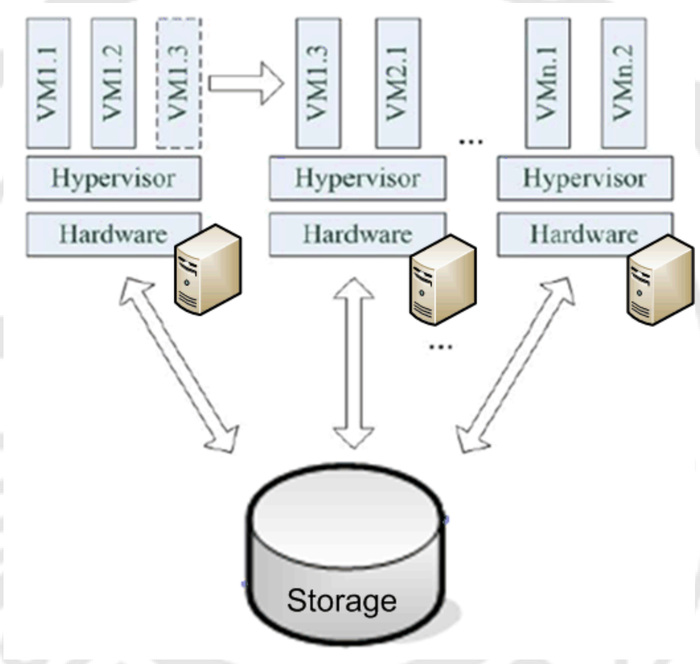
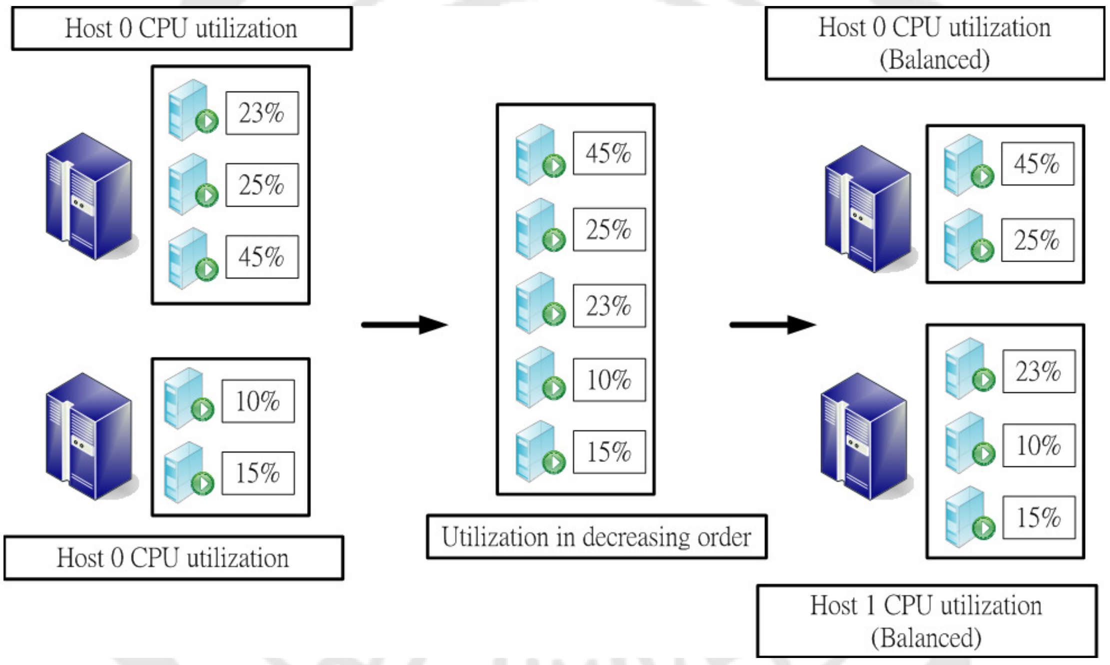


Figure 2-8. virtual machine migration procedure

The live migrating procedure shows as Figure 2-8, there is a shared storage device and plenty of computing nodes which host virtual machines. All computing nodes mount the shared storage device by NFS protocol and access it just as local storage, but impose limitations such as CPU type and generation compatibility. A virtual machine of node1 migrates to node2 to optimize resource allocation for virtual machines of node1 and node2. This procedure can complete in only several seconds. What is more, applications run on the virtual machine will keep continuous running in the migration nearly without any interrupting, which has significant meaning to 7x24 applications or services.

2.4 Dynamic Resource Allocation

Load balancing seeks to improve the performance of a distributed system by allocating the workload amongst a set of cooperating hosts. Such system may attempt to ensure the workload on each host is within a small tolerance of the workload on all other physical hosts, or may attempt to avoid congestion of individual servers. Load balancing can be either centralized or distributed[26].



Host 0 CPU utilization
23%
25%
45%
10%
15%
Host 0 CPU utilization
45%
25%
23%
10%
15%
Utilization in decreasing order
Host 0 CPU utilization (Balanced)
45%
25%
23%
10%
15%
Host 1 CPU utilization (Balanced)

Figure 2-9. Global Load Balance Algorithm concept

The purpose of DRA (Dynamic Resource Allocation) is to reach the best balance between each physical machine. Only the resource can be evenly distributed to achieve maximum efficiency. To achieve this, we designed a method to obtain the best strategy. According to Global Load balancing(GLB) algorithm[27] concept, each virtual machine's CPU Loading value will be added and divided by the sum of physical machine. We can learn the best loading ratio and migrate the suitable virtual machine to physical machine. Resources utilization ratio of physical machine will reach load balancing, as Figure 2-9.

For more detail please refer to section 3.2 in this thesis.

2.5 Related Work

Recently, the dramatic performance improvements in hypervisor technologies have made it possible to experiment with virtual machines (VM) as basic building blocks for flexible computational platforms. Many research efforts have been introduced to reduce the overhead of the networking in virtualized environments. Data transfer between server nodes frequently occurs in parallel and distributed computing systems, the high overhead of networking may induce significant performance loss in the overall system.

Jae-Wan Jang[19] use virtualized parallel and distributed computing systems are rapidly becoming the mainstream due to the significant benefit of high energy-efficiency and low management cost. Processing network operations in a virtual machine incurs a lot of overhead from the arbitration of network devices between virtual machines, inherently by the nature of the virtualized architecture. Wang L et al.[19] propose a new methodology for Grid computing – to use virtual machines as computing resources and provide Virtual Distributed Environments (VDE) for Grid users. Paul Willmann [35] presents hardware and software mechanisms to enable concurrent direct network access by operating systems running within a virtual machine monitor.

It is declared that employing virtual environment for Grid computing can bring various advantages, for instance, computing environment customization, QoS guarantee and easy management. A light weight Grid middleware, Grid Virtualization Engine, is developed accordingly to provide functions of building virtual environment for Grids.

VMware DRS[20] achieves an on-demand resource scheduling scheme for virtual machine cluster via migrating virtual machines among physical machines. In our scheme, the two measures are used simultaneously while reallocating resource of virtual machines within same physical machine is the first choice to get higher efficiency.

Additionally, R. S. Montero[21] proposes a performance model to characterize these variable capacity (elastic) cluster environments. The model can be used to dynamically dimension the cluster using cloud resources, according to a fixed budget, or to estimate the cost of completing a given workload in a target time. This thesis focuses VMs running on physical machines and use DRA technology to Implementation a virtualization environment of HPC.

Chapter 3
System Implementation

3.1 System Architecture

The OpenNebula core orchestrates three different management areas: image and storage technologies (that is, virtual appliance tools or distributed file systems) for preparing disk images for VMs, the network fabric (such as Dynamic Host Configuration Protocol [DHCP] servers, firewalls, or switches) for providing VMs with a virtual network environment, and the underlying hypervisors for creating and controlling VMs. The core performs specific storage, network, or virtualization operations through pluggable drivers. Thus, OpenNebula isn’t tied to any specific environment, providing a uniform management layer regardless of the underlying infrastructure.

Besides managing individual VMs’ life cycle, we also designed the core to support services deployment; such services typically include a set of interrelated components (for example, a Web server and database back end) requiring several VMs. Thus, we can treat a group of related VMs as a first-class entity in OpenNebula. Besides managing the VMs as a unit, the core also handles the delivery of context information (such as the Web server’s IP address, digital certificates, and software licenses) to the VMs[36].

A separate scheduler component makes VM placement decisions. More specifically, the scheduler has access to information on all requests OpenNebula receives and, based on these requests, keeps track of current and future allocations, creating and updating a resource schedule and sending the appropriate deployment commands to the OpenNebula core. The OpenNebula default scheduler provides a rank scheduling policy that places VMs on physical resources according to a ranking algorithm that the administrator can configure. It relies on real-time data from both the running VMs and available physical resources.

In Figure 3-1, it shows the system perspective. According to the previous works we build a cluster system with OpenNebula and also provide a web interface to manage virtual machines and physical machine. Our cluster system was built up with four homogeneous computers; the hardware of these computers is equipped with Intel i7 CPU 2.8 GHz, four gigabytes memory, 500 gigabytes disk, Debian operating system, and the network connected to a gigabit switch.

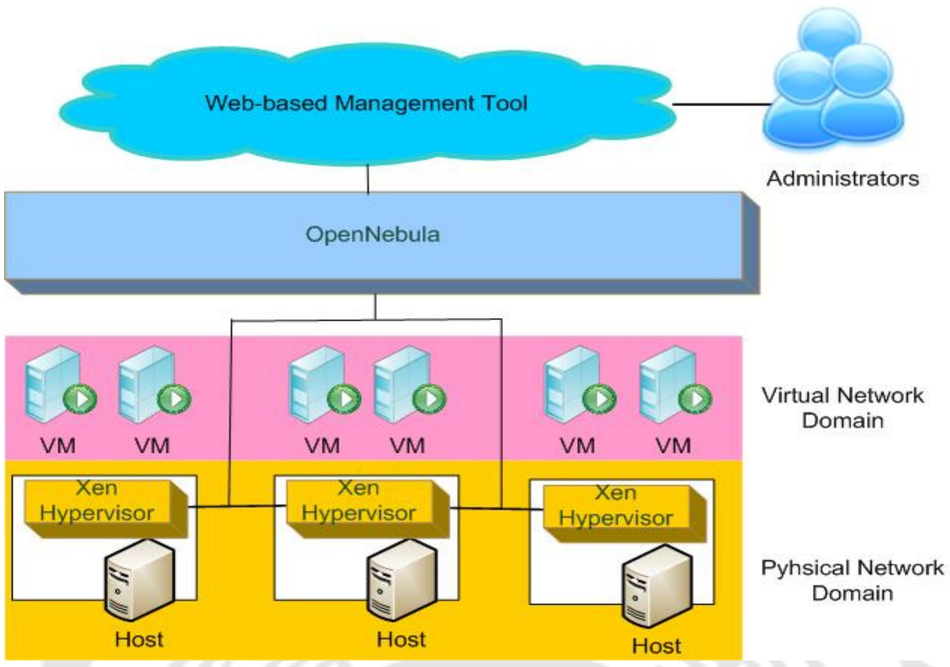


Figure 3-1. System Architecture

3.2 Dynamic Resource Allocation Algorithm

The Dynamic Resource Allocation is an efficient approach to increasing availability of host machine. However, at present open source virtual machine management software merely provide a web interface for user managing virtual machine. Such as Eucalyptus[37] cannot accomplish load balance. When a part of virtual machines load increasing, it will affect all virtual machine on the same host machine. Our Dynamic Resource Allocation algorithm can overcome this obstacle, and improve host machine performance. Dynamic Resource Allocation works by continuously monitoring all virtual machines resource usage to determine which virtual machine have to migrate to another host machine. The goal is to make all host machine CPU and memory loading identically.

The Dynamic Resource Allocation process is as follows. Assuming j host machines are in this pool. Every host machine loading ideal ratio is $\alpha = {}^{1}/_{j}$. And i virtual machines are not running load balancing in these host machine. Each virtual machine resource usage is defined “ ()⁄

$$(\mathbf{VMi\ CPUuse} \mid \mathbf{VMi\ RAM\ allocate}) \Big/ \textstyle\sum_{i=1}^{n} (\mathbf{VMi\ CPUuse} \mid \mathbf{VMi\ RAM\ allocate})$$

(1)

Where denotes virtual machine resource usage percentage in all allocate CPU

and memory physical resource. When D_Dd__________ _

In the next step, virtual machines resource usage ratio has been added up on different host machine. Each host machine current resource usage is defined

“ _()

$$\mathbf{HOST_j}_{\,\mathbf{Rate}} = \sum_{i=1}^{n} \mathbf{VMi\ on\ HOSTj\ \ Rate} \qquad (2)$$

Where 〚 〛 □ 〚 _() 〛

D_Dd__________ʲȽƧƧ________________ , and decide which host machine to be the migrated destination host, such as: . And at last, the migrated virtual machine is defined _()l_() 〖

(3)

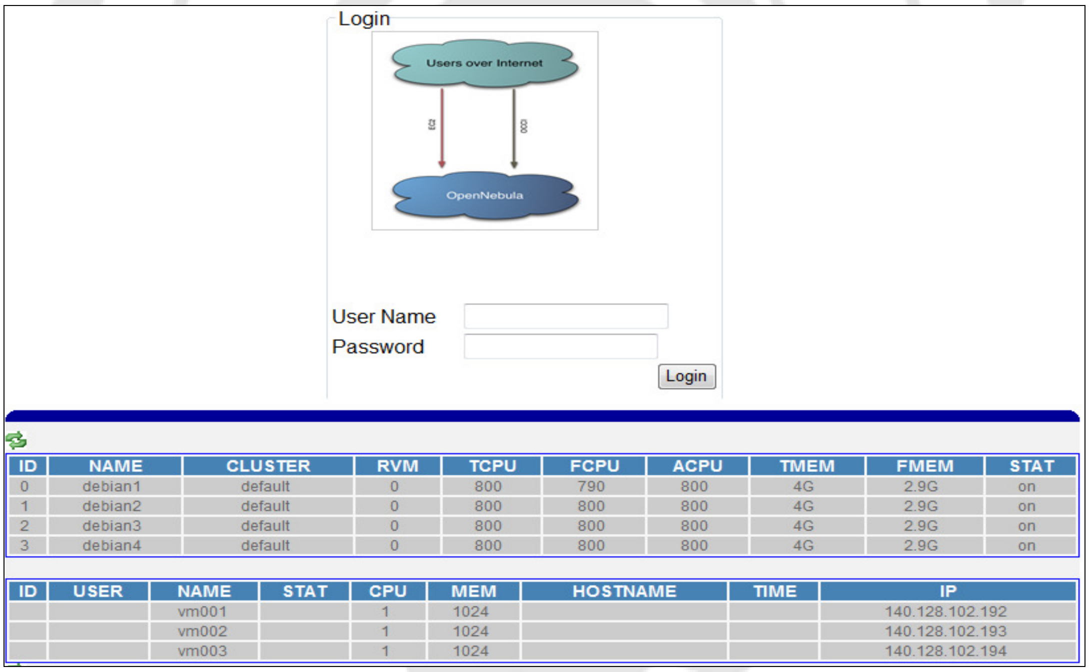
The algorithm performs some calculations for monitoring physical resource information. It is follows:

```
[Initialization]

  Defined virtual machine amount i and host machine amount j

  Calculate ideal ratio $\alpha = {}^{1}/_{j}$ ,virtual machine resource ratio 〖〗 □ 〖_()〗

do

  determine migrate source host machine m

  determine migrate destination host machine n
  determine migrated virtual machine on HOST j
  migrate virtual machine D_Dd__________ZðƧ
```

```
while (    value = 0)
```

3.3 Management Interface

We design a useful web interface for end users fastest and friendly to Implementation virtualization environment. In Figure 3-2, it shows the authorization mechanism, through the core of the web-based management tool, it can control and manage physical machine and VM life-cycle.

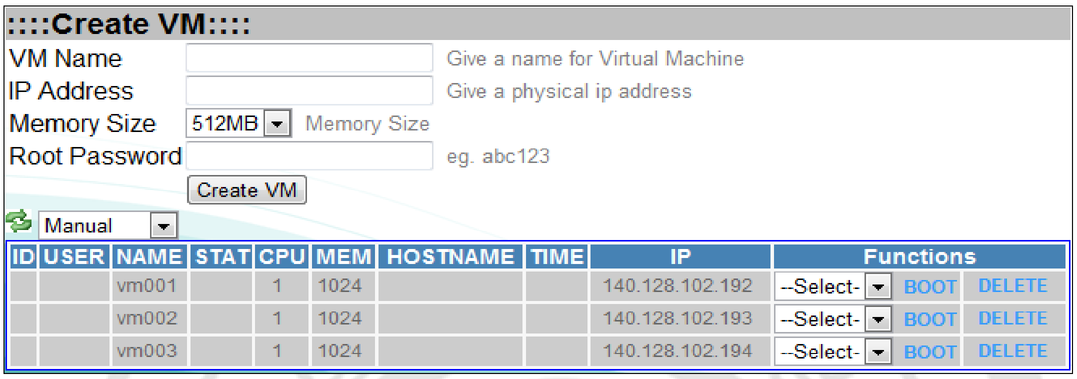


ID	NAME	CLUSTER	RVM	TCPU	FCPU	ACPU	TMEM	FMEM	STAT
0	debian1	default	0	800	790	800	4G	2.9G	on
1	debian2	default	0	800	800	800	4G	2.9G	on
2	debian3	default	0	800	800	800	4G	2.9G	on
3	debian4	default	0	800	800	800	4G	2.9G	on

ID	USER	NAME	STAT	CPU	MEM	HOSTNAME	TIME	IP
		vm001		1	1024			140.128.102.192
		vm002		1	1024			140.128.102.193
		vm003		1	1024			140.128.102.194

Figure 3-2. Web-Based Authorization

The entire web-based management tool including physical machine management, virtual machine management and performance monitor. In Figure 3-3, it can set the VM attributes such as memory size, IP address, root password and VM name etc…, it also including the life migrating function. Life migration means VM can move to any working physical machine without suspend in-service programs. Life Migration is one of the advantages of OpenNebula. Therefore we could migrate any VM what we want under any situation, thus, we have a DRA mechanism to make the migration function more meaningful. For more detail please refer to our experiment in chapter 4.



::::Create VM::::

VM Name		Give a name for Virtual Machine
IP Address		Give a physical ip address
Memory Size	512MB	Memory Size
Root Password		eg. abc123

Create VM

Manual

ID	USER	NAME	STAT	CPU	MEM	HOSTNAME	TIME	IP	Functions		
		vm001		1	1024			140.128.102.192	--Select-	BOOT	DELETE
		vm002		1	1024			140.128.102.193	--Select-	BOOT	DELETE
		vm003		1	1024			140.128.102.194	--Select-	BOOT	DELETE

Figure 3-3. Virtual Machine Manager

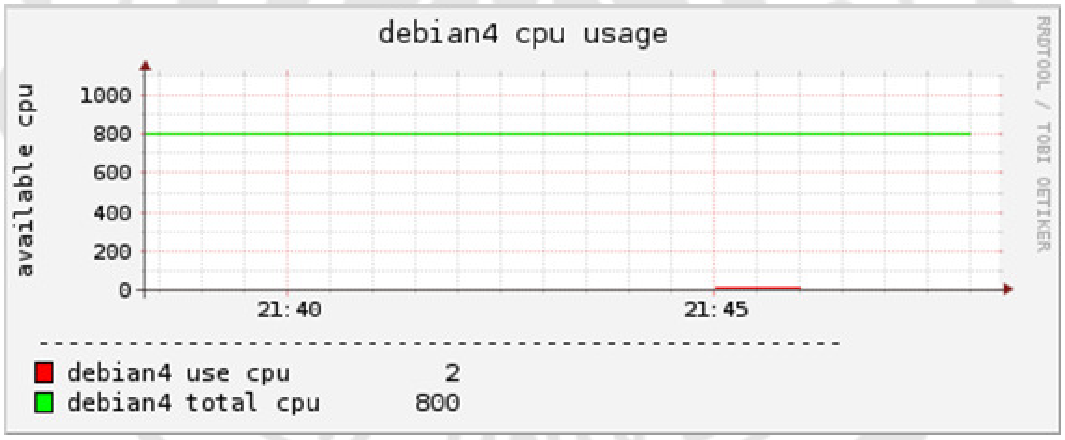
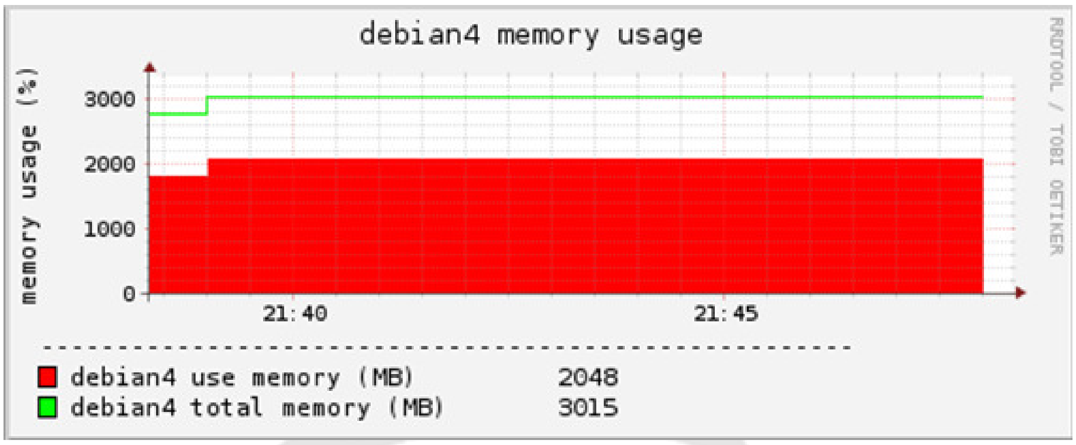


Figure 3-4. CPU Performance

RRDtool is the Open Source industry standard, high performance data logging and graphing system for time series data. Use it to write your custom monitoring shell scripts or create whole applications using its Perl, Python, Ruby, TCL or PHP bindings[38]. In this thesis we use RRDtool to monitor entire system. Figure 3-4 and Figure 3-5 show current physical machines CPU and memory usage.



🟥	debian4 use memory (MB)	2048
🟩	debian4 total memory (MB)	3015

Figure 3-5. Memory Performance

Chapter 4
Experimental Results

4.1 Performance and Capability Testing

In this section, first we measure the performance and network throughput in physical and virtual machine. We use Apache JMeter[39] to measure the testing result. It is a well-known web application measure performance tool of Apache project. Apache JMeter is a 100% pure java desktop application. JMeter can be used to test the performance of static or dynamic resources, it also can simulate the high load of server, network or other objects to test their pressure power of providing the services or analyze the total performance instances of their providing services under different circumstances[40].

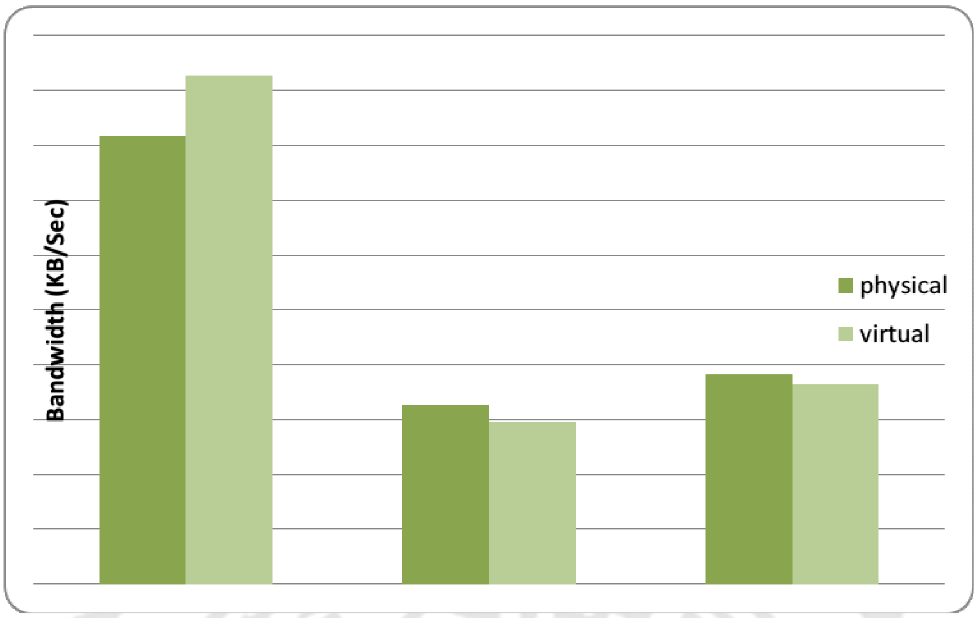


Figure 4-1. Memory Performance

Regarding to the throughput rate as shown Figure 4-1, it can explain for server capabilities or performance difference with physical and virtual machine.

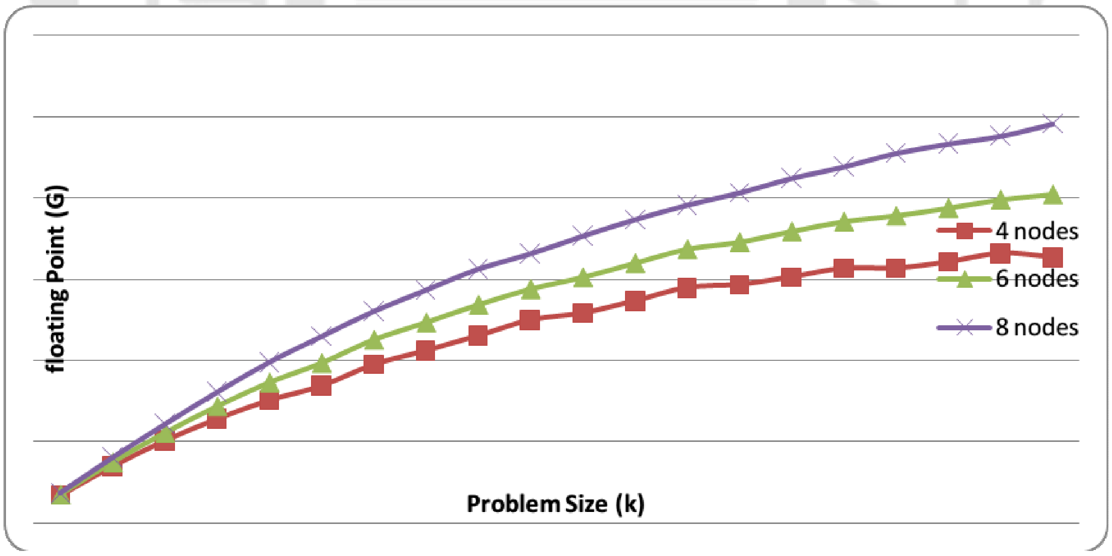


Figure 4-2. HPCC Performance Computing on different nodes graph

In Figure 4-2, it represent the performance by different total nodes which involve in HPCC[41] computing. We set every virtual machine 1 CPU and 1GB memory, and distributed virtual machine on different host to prevent node centralize on the same host. HPCC is an Internet-accessible resource that promotes the exchange of software and information among those involved with high-performance computing and communications[42]. The input parameters of HPCC can be considered with three key elements: P – the number of process rows, could be explained as CPU number, Q – the number of process columns could be explained as total server number, N – the order of the coefficient matrix A, it also called Problem Size in coming article. A formal formula can be described the required memory space of problem size: $matrixSize^2 \times 8\ bits = Required\ Memory$. The output of the HPCC is Gflops which means rate of execution for solving the linear system. $matrixSize^2 \times 8\ bits = Required\ Memory$ In Table 4-1, we can obviously know that when the computing nodes increased, it got better performance value in the same problem size. Furthermore, increasing the problem size, the difference of HPCC performance computing value is more significant.

Table 4-1. HPCC Performance Computing on different nodes value

Problem Size (K)	4 nodes (Gflops)	6 nodes (Gflops)	8 nodes (Gflops)
1000	3.41	3.54	3.68
2000	7.01	7.51	8.05
3000	10.15	11.10	12.14
4000	12.82	14.36	16.09
5000	15.12	17.30	19.80
6000	16.88	19.69	22.97
7000	19.51	22.55	26.07
8000	21.21	24.67	28.69
9000	23.05	26.85	31.28
10000	24.98	28.78	33.16
11000	25.84	30.22	35.34
12000	27.33	31.96	37.37
13000	28.95	33.67	39.16
14000	29.33	34.54	40.67
15000	30.28	35.85	42.45

16000	31.32	37.07	43.87
17000	31.36	37.79	45.54
18000	32.11	38.72	46.69
19000	33.17	39.75	47.64
20000	32.72	40.40	49.16

4.2 DRA Experimental Results

We focus on resource utilization of computing under *DRA* model. Therefore, we also used HPCC software to verify that *DRA* has a good performance and utilization on virtualization cluster. HPCC is an abbreviation of High Performance Computing Challenge, the HPC Challenge Benchmark is a set of benchmarks targeting to test multiple attributes that can contribute substantially to the real-world performance of HPC systems, co-sponsored by the DARPA High Productivity Computing Systems program, the United States Department of Energy and the National Science Foundation[37].

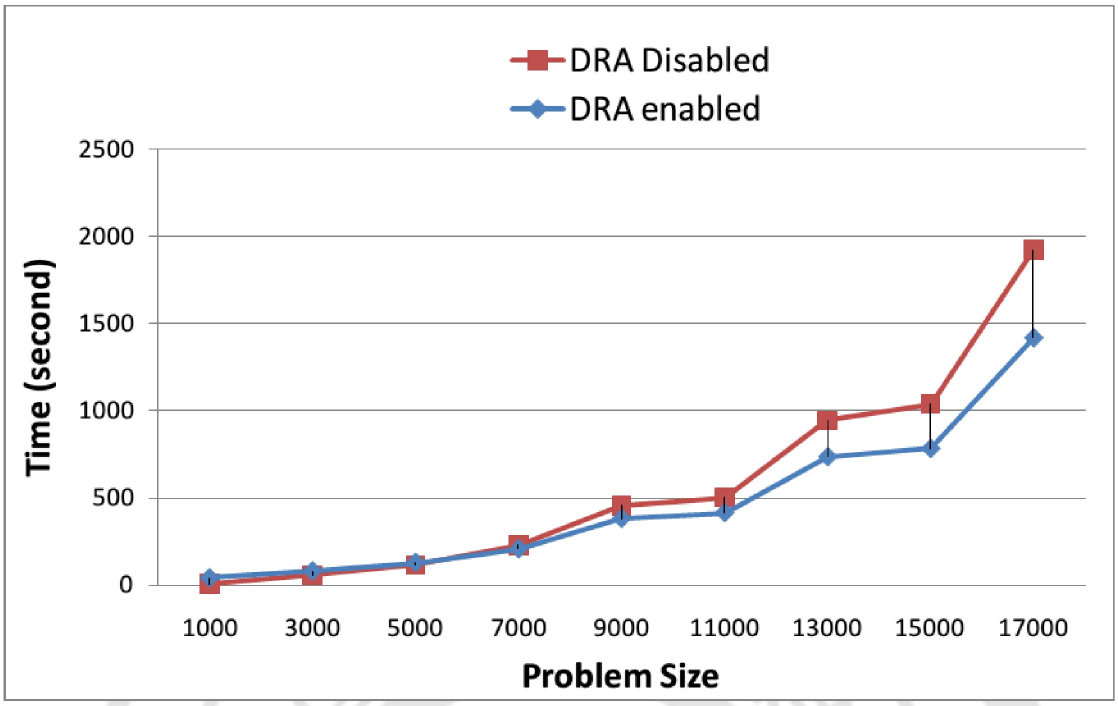


Figure 4-3. Time Cost

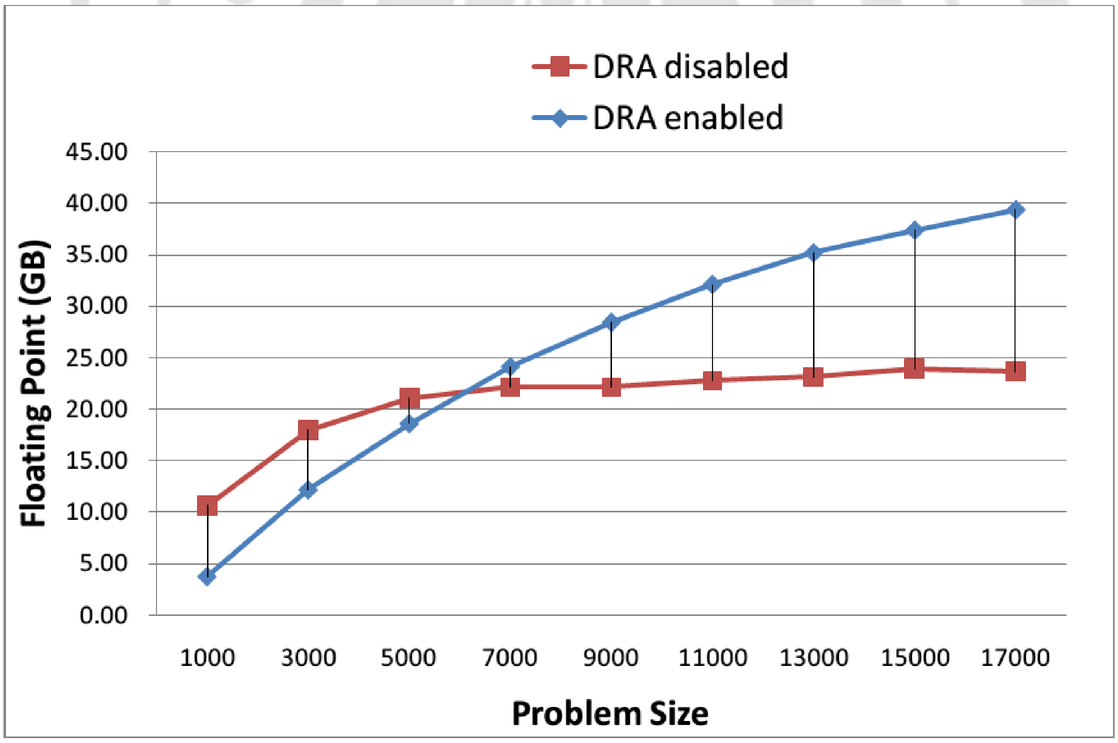


Figure 4-4. Floating Point Count

There were three physical machines in experimental environment. We created six virtual machines and distributed on different host machine. Each virtual machine used one virtual CPU and 512MB virtual memory. High workloads virtual machine on HOST 1 will be migrated to a lower resource cost physical machine when *DRA* function is enabled.

Table 4-2. Floating Point Count Value compared when DRA enabled or not

Problem Size	DRA disabled (GFlops)	DRA enabled (GFlops)
1000	10.65	3.77
3000	17.98	12.19
5000	21.08	18.62
7000	22.14	24.15
9000	22.15	28.46
11000	22.76	32.14
13000	23.15	35.21
15000	23.91	37.40
17000	23.67	39.39

Table 4-3. HPCC finished time when DRA enabled or not

Problem Size	DRA Disabled (Seconds)	DRA enabled (Seconds)
1000	7	42
3000	58	81
5000	116	130
7000	230	208
9000	457	383
11000	503	416
13000	945	737
15000	1039	784
17000	1924	1419

Figure 4-3 and Figure 4-4 are shown experiment results. DRA disable was red line, DRA enable is blue one. In this experiment, we put HPCC programs into six virtual machines and calculate HPCC performance on six virtual machines cluster. It caused virtual machines cluster CPU usage jumped and affect HOST machine CPU usage relatively. When DRA function disable, virtual machines located on same HOST machine and proceeding HPCC computing simultaneously. It caused virtual machines snatch at physical resource each other. When DRA function enable, it will detect all host machine resource usage was balancing or not, therefore, virtual machines on same HOST machine were migrated to others automatically. As Figure 4-4, it shows better performance when virtual machines centralized on the same host than on distributed hosts. Because HPCC performance computing on virtual machines cluster transfer computing data to each virtual machine, so these virtual machines deliver message to each other by the host virtual switch. But we observed that when problem size reach 6000, as Table 4-2 and Table 4-3, DRA enabled virtual machines distributed to different hosts, the HPCC performance is better than DRA disabled virtual machines. Because problem size is too big, so virtual machines cluster on the single host cannot afford the computation. Similarly, as Figure 4-3, it represented that HPCC computing finish earlier when DRA is enabled.

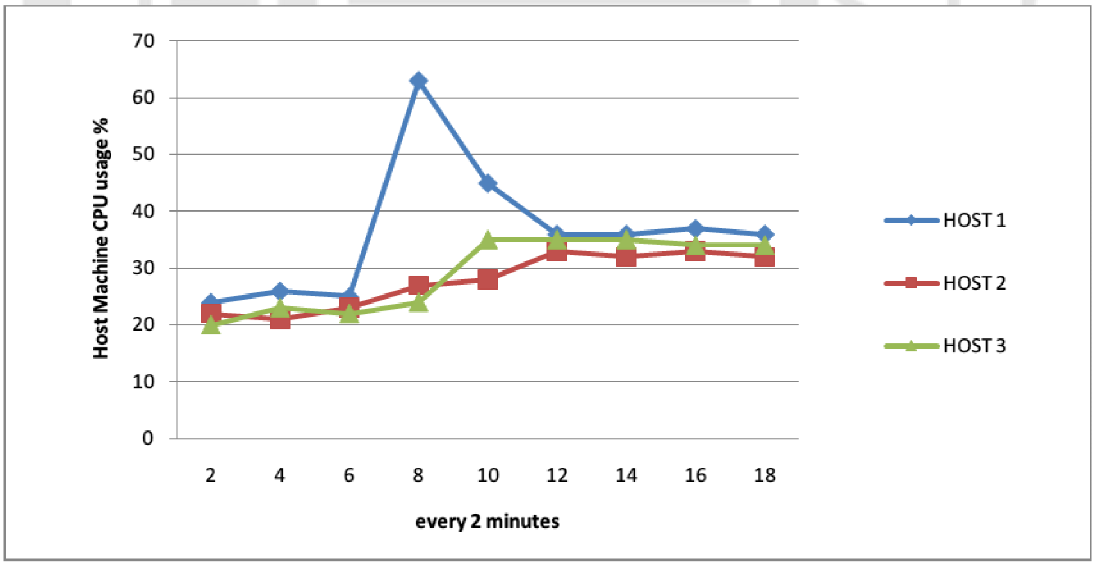


Figure 4-5. Host Machine CPU usage

Figure 4-4 is shown HPCC computing time. The horizontal axis represented HPCC problem size and the vertical axis represented HPCC computing time. We noticed that while HPCC problem size growing up, the difference of HPCC computing finished time when DRA function enable or not will be more obviously. Figure 4-5 is also shown DRA function effectiveness. The vertical axis represented virtual machine floating point calculation performance. With DRA function enabled will obtain good performance. It also proved our thesis is workable under this circumstance.

Chapter 5
Conclusions

In this work we have presented an optimization with dynamic resource allocation model for clusters that allows a flexible management of these computing platforms by: (1) supporting DRA mechanism; (2) implementation OpenNebula management tool on web-based interface; and (3) efficiently isolating the cluster workloads. Moreover, this architecture is able to transparently grow the cluster's capacity using an external cluster provider. Although there is another Open Source virtualization project like, Eucalyptus. But it is difficult to reach DRA goal, because lack life migration function. Therefore, we choose the OpenNebula solutions to hit our goal in this thesis.

Based on this model it is straightforward to plan the capacity of the cluster to, for instance, meet a deadline to complete a given workload. We envision the use of these kinds of models by additional components to dynamically change the cluster capacity according to a given budget, performance policy or in conjunction with a run and queue wait time prediction service. Finally, the architecture presented in this work is compatible with the use of physical resources. These resources can be divided evenly by the mechanism.

Bibliography

[1] R. P. G. Gerald J. Popek. (1974) Formal requirements for virtualizable third generation architectures.

[2] C. Clark*, et al.*, "Live migration of virtual machines," presented at the Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, 2005.

[3] Z. Yi and H. Wenlong, "Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud," presented at the INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, 2009.

[4] W. v. Hagen, *Professional Xen Virtualization*, 2008.

[5] R. N. Uhlig, G. Rodgers, D. Santoni, A. L. Martins, F. C. M. Anderson, A. V. Bennett, S. M. Kagi, A. Leung, F. H. Smith, L., "Intel virtualization technology," *Computer,* vol. 38, pp. 48-56, 2005.

[6] A. Whitaker*, et al.*, "Rethinking the design of virtual machine monitors," *Computer,* vol. 38, pp. 57-62, 2005.

[7] M. Rosenblum and T. Garfinkel, "Virtual machine monitors: current technology and future trends," *Computer,* vol. 38, pp. 39-47, 2005.

[8] A. Kivity*, et al.*, "kvm: the Linux virtual machine monitor," in *OLS '07: Proceedings of the Linux Symposium*, Ottawa, Ontario, Canada, 2007, pp. 225-230.

[9] M. S. Andrew Whitaker, Steven D Gribble, "Denali : Lightweight Virtual Machines for Distributed and Networked Applications," *Technical Report,* vol. 02, p. 10, 02,10 2002.

[10] P. Barham*, et al.*, "Xen and the art of virtualization," presented at the Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing, NY, USA, 2003.

[11] S. Lowe, "Mastering VMware vSphere 4," pp. 26-41, 2009.

[12] K. B. David E.Williams, Juan R. Garcia, and Rami Rosen, "Virtualization with Xen Including XenEnterprise, XenServer, and XenExpress," pp. 23-117, 2007.

[13] R. J. Srodawa and L. A. Bates, "An efficient virtual machine implementation," presented at the Proceedings of the workshop on virtual computer systems, Cambridge, Massachusetts, United States, 1973.

[14] M. H. XU Hui, WANG Xueli, WANG Zhuming "Study on the dynamic model

of leaf area of Eucalyptus camaldulensis," *Yunnan Forestry Science and Technology, 2000,* pp. 20-22.

[15] D. Nurmi*, et al.*, "The Eucalyptus Open-Source Cloud-Computing System," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, 2009, pp. 124-131.

[16] P. Sempolinski and D. Thain, "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 417-426.

[17] F. D. Dejan S. Miloji, Yves Paindaveine, Richard Wheeler, Songnian Zhou, "Process migration," *ACM Comput. Surv.,* vol. 32, pp. 241-299, 2000.

[18] J. G. Hansen and E. Jul, "Self-migration of operating systems," presented at the Proceedings of the 11th workshop on ACM SIGOPS European workshop, Leuven, Belgium, 2004.

[19] C.-H. T. 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.

[20] Resource Management with VMware DRS [Online].

[21] E. S. Jae-Wan Jang, Heeseung Jo, Jin-Soo Kim, "A low-overhead networking mechanism for virtualized high-performance computing systems," *The Journal of Supercomputing,* 2010.

[22] P. Willmann*, et al.*, "Concurrent Direct Network Access for Virtual Machine Monitors," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, 2007, pp. 306-317.

[23] B. Sotomayor*, et al.*, "Virtual Infrastructure Management in Private and Hybrid Clouds," *Internet Computing, IEEE,* vol. 13, pp. 14-22, 2009.

[24] D. Greschler and T. Mangan, "Networking lessons in delivering 'Software as a Service': part I," *Int. J. Netw. Manag.,* vol. 12, pp. 317-321, 2002.

[25] D. Greschler and T. Mangan, "Networking lessons in delivering 'Software as a Service': part II," *Int. J. Netw. Manag.,* vol. 12, pp. 339-345, 2002.

[26] J. H. H. K.Bubendorfer, *A Compositional Classification For Load-Balancing Algorithms*, 1998.

[27] G. Somani and S. Chaudhary, "Load Balancing in Xen Virtual Machine Monitor," in *Contemporary Computing*. vol. 95, S. Ranka*, et al.*, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 62-70.

[28] J. E. Smith and N. Ravi, "The architecture of virtual machines," *Computer,* vol. 38, pp. 32-38, 2005.

[29] *Amazon Web Services home page*. Available: http://aws.amazon.com/

[30] *Enomalism elastic computing infrastructure*. Available: http://www.enomaly.com/

[31] *Salesforce Customer Relationships Management (CRM) system*. Available: http://www.salesforce.com/

[32] *Logical Volume Manager*. Available: http://linuxconfig.org/Linux_lvm_-_Logical_Volume_Manager

[33] K. Z. Meth and J. Satran, "Features of the iSCSI protocol," *Communications Magazine, IEEE,* vol. 41, pp. 72-75, 2003.

[34] E. Anderson*, et al.*, "Hippodrome: Running Circles Around Storage Administration," presented at the Proceedings of the 1st USENIX Conference on File and Storage Technologies, Monterey, CA, 2002.

[35] *OpenNebula*. Available: http://www.opennebula.org

[36] 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,* 2007.

[37] *Eucalyptus*. Available: http://open.eucalyptus.com

[38] R. S. M. Borja Sotomayor, Ignacio M. Llorente, Ian Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing,* vol. 13, 2009.

[39] *Apache JMeter*. Available: http://jakarta.apache.org

[40] J. You*, et al.*, "JMeter-based aging simulation of computing system," in *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, 2010, pp. 282-285.

[41] *HPCC*. Available: http://icl.cs.utk.edu/hpcc/

[42] S. Browne*, et al.*, "The National HPCC Software Exchange," *Computational Science & Engineering, IEEE,* vol. 2, pp. 62-69, 1995.

Appendix – Installation Guide

Appendix A. Prerequisite

System Partition:

/	40GB
Swap	4GB
/data	other space

Operation System:

Debian 5.0 i386

Kernel:

2.6.26-2-xen-686

Appendix B. OpenNebula Server Installation

Install Package:

xen-linux-system-2.6.26-2-xen-686
python-xml
xen-tools
g++ ruby
libsqlite3-0
sqlite3
libsqlite3-dev
libsqlite3-ruby
libxmlrpc-c3-dev
libxmlrpc-c3
libssl-dev
scons
mysql-server-5.0
php5
etherwake
expect
php5-mysql
nfs-kernel-server

Configuration Modified:

```
/etc/hosts
127.0.0.1          localhost
140.128.102.171 debian1.hpc.cs.thu.edu.tw          debian1
140.128.102.172 debian2.hpc.cs.thu.edu.tw          debian2
140.128.102.173 debian3.hpc.cs.thu.edu.tw          debian3
140.128.102.174 debian4.hpc.cs.thu.edu.tw          debian4

/etc/ssh/sshd_config
StrictHostKeyChecking no

/etc/xen/xend-config.sxp
(xend-http-server yes)
(xend-unix-server yes)
(xend-tcp-xmlrpc-server yes)
#(xend-unix-xmlrpc-server yes)
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-address localhost)
(xend-relocation-hosts-allow '')
(network-script 'network-bridge netdev=eth0')
#(network-script network-dummy)

/etc/exports
/data/domains
140.128.102.171(rw,sync,no_subtree_check,no_root_squash)
140.128.102.172(rw,sync,no_subtree_check,no_root_squash)
140.128.102.173(rw,sync,no_subtree_check,no_root_squash)
140.128.102.174(rw,sync,no_subtree_check,no_root_squash)
140.128.102.181(rw,sync,no_subtree_check,no_root_squash)
140.128.98.31(rw,sync,no_subtree_check,no_root_squash)

/data/one
140.128.102.171(rw,sync,no_subtree_check,no_root_squash)
140.128.102.172(rw,sync,no_subtree_check,no_root_squash)
140.128.102.173(rw,sync,no_subtree_check,no_root_squash)
140.128.102.174(rw,sync,no_subtree_check,no_root_squash)
140.128.102.181(rw,sync,no_subtree_check,no_root_squash)
```

```
    /data/xen
    140.128.102.171(rw,sync,no_subtree_check,no_root_squash)
    140.128.102.172(rw,sync,no_subtree_check,no_root_squash)
    140.128.102.173(rw,sync,no_subtree_check,no_root_squash)
    140.128.102.174(rw,sync,no_subtree_check,no_root_squash)
    140.128.102.181(rw,sync,no_subtree_check,no_root_squash)
```

a. Configure Xen environment

```
$ sudo su
# echo xen.independent_wallclock=1 >> /etc/sysctl.conf
# echo loop max_loop=100 >> /etc/modules
# echo xenblktap >> /etc/modules
# ln -s /usr/lib/xen-3.2-1/bin/tapdisk /usr/sbin
```

b. Install Xen-tools and configure

```
$ sudo aptitude -y install xen-tools
$ cd /etc/xen-tools
$ sudo gedit xen-tools.conf
dir = /data/xen
size     = 4Gb
dist     = lenny
gateway     = 192.168.200.254
netmask     = 255.255.255.0
broadcast = 192.168.200.255
passwd = 1
mirror = http://free.nchc.org.tw/debian
serial_device = hvc0
output       = /data/domains
```

c. Create Virtual Machine

virtual machine will create in /data/domains.

```
$ sudo mkdir /home/domains
$ sudo xen-create-image --hostname=vm01 --ip=192.168.200.X
```

d. Virtual Machine basic configuration

Start virtual machine.

```
$ sudo xm console vm01
```

Virtual machine basic configuration.

```
# aptitude -y install udev ntpdate
# cp /usr/share/zoneinfo/Asia/Taipei /etc/localtime
# ntpdate    time.nist.gov
# echo xen.independent_wallclock=1 >> /etc/sysctl.conf
```

e. Install OpenNEbula

```
$ cd
$ wget http://dev.opennebula.org/attachments/download/103/one-1.4.0.tar.gz
$ tar zxvf one-1.4.0.tar.gz
$ cd one-1.4
$ sudo scons
$ sudo mkdir /home/one
$ sudo ./install.sh -d /home/one
$ sudo su
# echo export ONE_LOCATION=/home/one >> ~/.bashrc
# echo export ONE_XMLRPC="http://localhost:2633/RPC2" >> ~/.bashrc
# echo export PATH='$ONE_LOCATION/bin:$PATH' >> ~/.bashrc
# echo export ONE_AUTH=/home/one/.one/one_auth >> ~/.bashrc
# mkdir /home/one/.one
# echo "root:cloud123" >> /home/one/.one/one_auth
```

f. Edit OpenNEbula Setting

```
# cd /home/one
# gedit etc/oned.conf

HOST_MONITORING_INTERVAL = 5
VM_POLLING_INTERVAL          = 10
```

```
IM_MAD = [
        name          = "im_xen",
        executable = "one_im_ssh",
        arguments    = "im_xen/im_xen.conf" ]
#IM_MAD = [
#           name          = "im_kvm",
#           executable = "one_im_ssh",
#           arguments    = "im_kvm/im_kvm.conf" ]
VM_MAD = [
          name          = "vmm_xen",
          executable = "one_vmm_xen",
          default      = "vmm_xen/vmm_xen.conf",
          type          = "xen" ]
#VM_MAD = [
#         name          = "vmm_kvm",
#         executable = "one_vmm_kvm",
#         default      = "vmm_kvm/vmm_kvm.conf",
#         type          = "kvm" ]
#         TM_MAD = [
#         name          = "tm_ssh",
#         executable = "one_tm",
#         arguments    = "tm_ssh/tm_ssh.conf" ]
          TM_MAD = [
          name          = "tm_nfs",
          executable = "one_tm",
          arguments    = "tm_nfs/tm_nfs.conf" ]
```

g. OpenNEbula daemon start

root ssh without password

```
ssh-keygen
cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
scp -r ~/.ssh debian2:~
scp -r ~/.ssh debian3:~
scp -r ~/.ssh debian4:~
```

start oned

```
ssh-keygen
cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
scp -r ~/.ssh debian2:~
scp -r ~/.ssh debian3:~
scp -r ~/.ssh debian4:~
```

h. Add OpenNEbula Client

```
onehost add debian1 im_xen vmm_xen tm_nfs
onehost add debian2 im_xen vmm_xen tm_nfs
onehost add debian3 im_xen vmm_xen tm_nfs
onehost add debian4 im_xen vmm_xen tm_nfs
```

i. Convert Xen VM to OpenNEbula Format

```
NAME       = vm01
CPU        = 1
MEMORY     = 512
OS = [  kernel   =        /boot/vmlinuz-2.6.26-2-xen-686,
        initrd   =        /boot/initrd.img-2.6.26-2-xen-686,
        root     =         sda2 ]
DISK = [ source =         /data/xen/domains/vm01/disk.img,
        clone    =         no,
        target   =        sda2,
        readonly          =         no]
DISK = [ type    = swap,
        size     = 128,
        target   = sda1,
        readonly = no ]
NIC=[IP="192.168.200.X", MAC="00:16:00:00:00:XX"]
```

j. Using One Open VM

```
onevm create vm01.one ; onevm deploy vm01 debian2
```

k. Shutdown VM

```
onevm list
onevm shutdown vm01
```

Appendix C. OpenNEbula Client Installation

a. Configure Xen environment

```
$ sudo su
# echo xen.independent_wallclock=1 >> /etc/sysctl.conf
# echo loop max_loop=100 >> /etc/modules
# echo xenblktap >> /etc/modules
# ln -s /usr/lib/xen-3.2-1/bin/tapdisk /usr/sbin
```

b. NFS mount server partition

```
$ sudo aptitude install nfs-common
$ sudo mkdir /home/domains
$ sudo mount.nfs debian1:/data/domains /data/domains
$ sudo mkdir /home/one
$ sudo mount.nfs debian1:/data/one/ /data/one
$ sudo mkdir /home/xen
$ sudo mount.nfs debian1:/data/xen/ /data/xen
```

c. Migrate VM

```
login debian1
$ cd /home/domains
$ sudo su
# onevm create vm01.one ; onevm deploy vm01 debian1
# onevm livemigrate vm01 debian2
```

Appendix D. OpenNebula Script

a. Host Management

add_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onehost add $1 im_xen vmm_xen tm_nfs
/var/www/script/rrd_graph/src/make_cpu_mem.sh $1
```

delete_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
hostid=`onehost list | grep $1 | awk '{print $1}'`
onehost delete $hostid
```

collect_mac.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
host_server=`onehost list | sed 1d | awk '{print $2}'`
for input_server in $host_server
do
        input_mac=`ssh $input_server ifconfig | grep peth | awk '{print $5}'`
        echo $input_server $input_mac >> host_mac.list
done
```

boot_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
input_mac=`cat /var/www/script/host_mac.list | grep $1 | awk '{print $2}'`
etherwake $input_mac
```

shutdown_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onehost disable $!
ssh $1 shutdown -h now
```

enable_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onehost enable $1/var/www/script/rrd_graph/src/delete_cpu_mem.sh $1
```

disable_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onehost disable $1
```

initial_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
#scp /etc/apt/sources.list $1:/etc/apt/
ssh -n $1 "apt-get update"
ssh -n $1 "apt-get -y install openssh-server ntpdate"
ssh -n $1 "/etc/init.d/network-manager-dispatcher stop"
ssh -n $1 "/etc/init.d/network-manager stop"
ssh -n $1 "apt-get -y remove network-manager"
#install xen deb
ssh -n $1 "apt-get -y install xen-linux-system-2.6.26-2-xen-686 python-xml"
#env setup
ssh -n $1 "echo xen.independent_wallclock=1 >> /etc/sysctl.conf"
ssh -n $1 "echo loop max_loop=100 >> /etc/modules"
ssh -n $1 "echo xenblktap >> /etc/modules"
ssh -n $1 "ln -s /usr/lib/xen-3.2-1/bin/tapdisk /usr/sbin"
#create directory for opennebula
mkdir /data/domains
mkdir /data/one
mkdir /data/xen
#deliver basic config and reboot
scp /var/www/script/config/xend-config.sxp $1:/etc/xen/
scp /var/www/script/config/hosts $1:/etc/
scp /var/www/script/config/rc.local $1:/etc/
scp /var/www/script/config/ssh_config $1:/etc/
ssh -n $1 "/etc/rc.local"
ssh -n $1 "df -h"
#install opennebula package
ssh -n $1 "apt-get -y install ruby libsqlite3-0 sqlite3 libsqlite3-dev libsqlite3-ruby
libxmlrpc-c3-dev libxmlrpc-c3 libssl-dev scons"
ssh -n $1 "reboot"
```

one_start.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
one start
```

one_stop.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
one stop
```

reboot_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onehost disable $!
```

show_host.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onehost show $1
```

b. VM Management

boot_one_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
cd /data/domains
onevm create $1.one
onevm deploy $1 $2
```

convert.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
#xen disk path
host_path="/data/domains"
host_cfg=$1
#xen cfg form
host_name=`cat $host_path/$host_cfg.cfg | grep name | awk '{print $3}' | sed 1d
| cut -d \' -f 2`
host_mem=`cat $host_path/$host_cfg.cfg    | grep memory | awk '{print $3}' | sed
1
d | cut -d \' -f 2`
host_kernel=`cat $host_path/$host_cfg.cfg | grep kernel | awk '{print $3}' | cut
  -d \' -f 2`
host_ramdisk=`cat $host_path/$host_cfg.cfg | grep ramdisk | awk '{print $3}' | c
ut -d \' -f 2`
host_disk=`cat $host_path/$host_cfg.cfg | grep root | awk '{print $3}' | cut -d
\' -f 2 | cut -d \/ -f 3`
host_sourcedisk=`cat $host_path/$host_cfg.cfg | grep file | grep disk | cut -d \' -f 2
| cut -d \/ -f 6 | cut -d \, -f 1`
```

```
host_target=`cat $host_path/$host_cfg.cfg | grep file | grep disk | cut -d \' -f 2 | cut
-d \/ -f 6 | cut -d \, -f 2`
host_swap=`cat $host_path/$host_cfg.cfg | grep file | grep swap | cut -d \' -f 2 | cut
-d \/ -f 6 | cut -d \, -f 2 `
host_ip=`cat $host_path/$host_cfg.cfg | grep vif | awk '{print $4}' | cut -d \' -f 2 |
cut -d \, -f 1 | cut -d \= -f 2`
host_mac=`cat $host_path/$host_cfg.cfg | grep vif | awk '{print $4}' | cut -d \' -f 2
| cut -d \, -f 2 | cut -d \= -f 2`

#create opennebula cfg form
sudo echo "NAME        = $host_name" >    $host_path/$host_cfg.one
sudo echo "VCPU          = 1"                   >> $host_path/$host_cfg.one
sudo echo "MEMORY    = $host_mem"    >> $host_path/$host_cfg.one
sudo echo "OS = [    kernel    =             $host_kernel," >>
$host_path/$host_cfg.one
sudo echo "               initrd    =             $host_ramdisk, " >>
$host_path/$host_cfg.one
sudo echo "               root        =             $host_disk ]" >>
$host_path/$host_cfg.one
sudo echo "DISK = [ source =
/data/xen/domains/$host_name/$host_sourcedisk," >> $host_path/$host_cfg.one
sudo echo "               clone      =             no," >> $host_path/$host_cfg.one
sudo echo "               target    =             $host_target," >>
$host_path/$host_cfg.one
sudo echo "               readonly              =             no]" >>
$host_path/$host_cfg.one
sudo echo "DISK = [ type      = swap," >> $host_path/$host_cfg.one
sudo echo "               size        = 128," >> $host_path/$host_cfg.one
sudo echo "               target    = "$host_swap"," >> $host_path/$host_cfg.one
sudo echo "               readonly = "no" ]" >> $host_path/$host_cfg.one
sudo echo "NIC=[IP=\"$host_ip\", MAC=\"$host_mac\"]" >>
$host_path/$host_cfg.one
```

create_default_vm.sh

```
#!/usr/bin/expect
#set env var
set host_name [lindex $argv 0]
set host_ip [lindex $argv 1]
set host_size [lindex $argv 2]
set host_passwd [lindex $argv 3]
set timeout 2400
#create vm
spawn xen-create-image --hostname=$host_name --ip=$host_ip --size=$host_size
--force
expect "Enter new UNIX password: "
send "$host_passwd\r"
expect "Retype new UNIX password: "
send "$host_passwd\r"
#
interact
```

create_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
sudo /var/www/script/create_default_vm.sh $1 $2 $3 $4
sudo xm create /data/domains/$1.cfg
sudo /var/www/script/install_default_vm.sh $1 $2 $3 $4
sudo xm destroy $1
sudo /var/www/script/convert.sh $1
```

delete_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
rm /data/domains/$1.cfg
rm /data/domains/$1.one
rm -rf /data/xen/domains/$1
```

edit_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
cat /data/domains/"$1".one | sed "s/VCPU..*/VCPU        = "$2"/g" | sed
"s/MEMORY..*/MEMORY    = "$3"/g" > /data/domains/"$1".one
```

install_default_vm.sh

```
#!/usr/bin/expect

#set env var

set host_name [lindex $argv 0]
set host_ip [lindex $argv 1]
set host_size [lindex $argv 2]
set host_passwd [lindex $argv 3]
set timeout 2400

#install vm default package
spawn /usr/sbin/xm console $host_name
expect "$host_name login: "
send "root\r"
```

```
sleep 2
expect "Password: "
send "$host_passwd\r"
sleep 2
send "cp /usr/share/zoneinfo/Asia/Taipei /etc/localtime\r"
send "echo xen.independent_wallclock=1 >> /etc/sysctl.conf\r"
send "ping 168.95.1.1\r"
expect "64 bytes from 168.95.1.1: icmp_seq="
send "\003"
send "ntpdate time.nist.gov\r"
send "aptitude -y install udev ntpdate\r"
expect "Reading task descriptions... Done"
send    "\x1d"
```

list_all_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

#
sudo -s
source /root/.bashrc
#

vm_name=`ls /data/domains/ | grep .one | cut -d . -f 1`

for input_vm in $vm_name
do
        vcpu=`cat /data/domains/$input_vm.one | grep VCPU | cut -d = -f 2`
        vmem=`cat /data/domains/$input_vm.one | grep MEMORY | cut -d = -f
2`
        vm_ip=`cat /data/domains/$input_vm.one | grep IP | cut -d \" -f 2`

        echo "NAME = $input_vm"
        echo "CPU = $vcpu"
        echo "MEM = $vmem"
        echo "IP = $vm_ip"
done
```

list_boot_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
#main code
onevm list
#
interact
```

migrate_one_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onevm livemigrate $1 $2
```

shutdown_one_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
onevm shutdown $1
```

shutdown_vm.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
xm destroy $1
```

ssh_vm.sh

```
#!/usr/bin/expect
#set env var
set host_ip [lindex $argv 0]
set host_passwd [lindex $argv 1]
set timeout 2400
#create vm
spawn ssh root@$host_ip
expect "password: "
send "$host_passwd\r"
#
interact
```

c. RRDTool Graph Script

delete_cpu_mem.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
rrd_path="/var/www/script/rrd_graph"
graph_path="/var/www/graph"
rm "$rrd_path"/rrd/"$1"_cpufree.rrd
rm "$rrd_path"/rrd/"$1"_memuse.rrd
rm "$graph_path"/"$1"_cpufree.png
rm "$graph_path"/"$1"_memuse.png
```

graph_cpu_mem.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
#
while true ;
do
server=`/data/one/bin/onehost list | sed 1d | awk '{print $2}'`

for input_server in $server
do

rrd_path="/var/www/script/rrd_graph"
graph_path="/var/www/graph"
cpu_rrd="$rrd_path"/rrd/"$input_server"_cpufree.rrd
mem_rrd="$rrd_path"/rrd/"$input_server"_memuse.rrd
now=`date +%s`

tcpu=`onehost list | grep "$input_server" | awk '{print $5}'`
fcpu=`onehost list | grep "$input_server" | awk '{print $6}'`
use_cpu=` expr $tcpu - $fcpu `

vm_free_mem=`ssh "$input_server" xm info | grep free_memory | awk '{print
$3}'`
vm_use_mem=`onevm list | grep "$input_server" |grep runn | awk '{print $6}' |
cut -d M -f 1`

initial_vm_mem=0
for plus_vm_mem in $vm_use_mem
do
        initial_vm_mem=`expr "$initial_vm_mem" + "$plus_vm_mem"`
done
vm_use_mem=$initial_vm_mem
```

```
if [ -z "$vm_use_mem" ] ; then
        vm_use_mem=0
fi

vm_total_mem=`expr $vm_free_mem + $vm_use_mem`
#rrdtool update data
rrdtool update $cpu_rrd $now:$use_cpu:$tcpu
rrdtool update $mem_rrd $now:$vm_use_mem:$vm_total_mem
#rrdtool graph rrd data
rrdtool graph $graph_path/"$input_server"_cpufree.png \
--title ""$input_server" cpu usage " \
DEF:t1=$cpu_rrd:"$input_server"-fcpu:AVERAGE \
DEF:t2=$cpu_rrd:"$input_server"-tcpu:AVERAGE \
COMMENT:"-----------------------------------------------------\n" \
AREA:t1#ff0000:""$input_server" use cpu" \
GPRINT:t1:LAST:"    %7.0lf \n" \
LINE1:t2#00ff00:""$input_server" total cpu" \
GPRINT:t2:LAST:"%7.0lf \n" \
-v "available cpu" -M -u 1000 -l 0\
-Y -X b -h 100 -w 400 -s `date --date='10 minute ago' +%s`

rrdtool graph $graph_path/"$input_server"_memuse.png \
--title ""$input_server" memory usage " \
DEF:t1=$mem_rrd:"$input_server"-umem:AVERAGE \
DEF:t2=$mem_rrd:"$input_server"-tmem:AVERAGE \
COMMENT:"-----------------------------------------------------\n" \
AREA:t1#ff0000:""$input_server" use memory (MB)     " \
GPRINT:t1:LAST:"    %7.0lf \n" \
LINE1:t2#00ff00:""$input_server" total memory (MB)" \
GPRINT:t2:LAST:"    %7.0lf \n" \
-v "memory usage (%)" -M -u 100 -l 0\
-Y -X b -h 100 -w 400 -s `date --date='10 minute ago' +%s`
done
sleep 12
done
```

graph_pdu_rrd.sh

```
#!/bin/sh
#-------------------env variable-------------
input_rrd=""
rrd_cmd="/usr/bin/rrdtool"
image_path="/var/www/graph"
pdu_rrd=""
now=`date +%s`
pdu_snmp="public"
pdu_ip="140.128.102.193"
#for paper record
day=`date +%d`
month=`date +%m`
#-------------------------------------------

for temp_ip in $pdu_ip
do
	pdu_total=`/usr/bin/snmpwalk -c $pdu_snmp -v 1 $temp_ip
enterprises.17420.1.1.4.2.1.2.0 | awk '{print $4}'`
	pdu_total_w=`echo "scale=4;$pdu_total * 11" | bc`
	input_rrd="$now:$pdu_total_w"

	#update rrd value
	pdu_rrd="/var/www/script/rrd_graph/rrd/pdu_$temp_ip.rrd"
	$rrd_cmd update $pdu_rrd $input_rrd

	#graph rrd png
	lastip=`echo $temp_ip | cut -d . -f 4`
	$rrd_cmd graph $image_path/pdu_$temp_ip.png \
	--title "HPC.THU PDU Watt Usage" \
	DEF:t1=$pdu_rrd:pdu-$lastip-total:AVERAGE \
COMMENT:"---------------------------------------------CURRENT--------------------
-------------------------------\n" \
            LINE2:t1#FF0000:"pdu $temp_ip $line_1                                  " \
            GPRINT:t1:LAST:"%7.2lf \n" \
            -v "PDU Power Watt" -M -l 0\
            -Y -X b -h 250 -w 600 -s `date --date='1 day ago' +%s`

            #for paper record
            date +%H-%M >> /var/www/script/rrd_graph/tmp/$day-$month
            echo $pdu_total_w >> /var/www/script/rrd_graph/tmp/$day-$month
done
```

make_cpu_mem.sh

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

#
rrd_path="/var/www/script/rrd_graph"
rrdtool create "$rrd_path"/rrd/"$1"_memuse.rrd -s 60 \
DS:"$1"-umem:GAUGE:60:0:10000000 \
DS:"$1"-tmem:GAUGE:60:0:10000000 \
RRA:AVERAGE:0.5:1:603 \
RRA:AVERAGE:0.5:6:603 \
RRA:AVERAGE:0.5:24:603 \
RRA:AVERAGE:0.5:288:800 \
RRA:MAX:0.5:1:603 \
RRA:MAX:0.5:6:603 \
RRA:MAX:0.5:24:603 \
RRA:MAX:0.5:288:800
```

```
rrd_path="/var/www/script/rrd_graph"
rrdtool create "$rrd_path"/rrd/"$1"_cpufree.rrd -s 60 \
DS:"$1"-fcpu:GAUGE:60:0:10000000 \
DS:"$1"-tcpu:GAUGE:60:0:10000000 \
RRA:AVERAGE:0.5:1:603 \
RRA:AVERAGE:0.5:6:603 \
RRA:AVERAGE:0.5:24:603 \
RRA:AVERAGE:0.5:288:800 \
RRA:MAX:0.5:1:603 \
RRA:MAX:0.5:6:603 \
RRA:MAX:0.5:24:603 \
RRA:MAX:0.5:288:800
```

make_pdu_rrd.sh

```
#!/bin/sh

rrd_dir="/var/www/script/rrd_graph/rrd"
input_rrd=""
rrd_cmd="/usr/bin/rrdtool"
pdu_ip="140.128.102.193"

for temp_ip in $pdu_ip
do
lastip=`echo $temp_ip | cut -d . -f 4`

$rrd_cmd create $rrd_dir/pdu_$temp_ip.rrd -s 300 \
DS:pdu-$lastip-total:GAUGE:300:0:10000000 \
RRA:AVERAGE:0.5:1:603 \
RRA:AVERAGE:0.5:6:603 \
RRA:AVERAGE:0.5:24:603 \
RRA:AVERAGE:0.5:288:800 \
RRA:MAX:0.5:1:603 \
RRA:MAX:0.5:6:603 \
RRA:MAX:0.5:24:603 \
RRA:MAX:0.5:288:800

done
```