

私立東海大學
資訊工程研究所

碩士論文

指導教授：楊朝棟 博士

於雲端環境上實作虛擬機器之綠能演算法
Implementation of a Green Power Management
Algorithm for Virtual Machines on Cloud
Computing Environments

研究生：王冠傑

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 王冠傑 所提之論文

於雲端環境上實作虛擬機器之綠能演算法

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

李冠憬

賴冠州

朱正忠

指導教授 楊朝棟 簽章

中華民國 100 年 6 月 28 日

摘要

隨著政府和企業電子化的高度發展，電子化服務的需求正持續、快速地增加，故需要建置更多的資據中心提供服務，這會耗費相當大的電能，特別是當服務運行在低使用率的狀況，例如在 10% 的處理器使用率下，其總功率消耗至少在峰值的 60% 以上，使得資源閒置、電能浪費，這是造成數據中心能源效率低落的主要原因。本篇論文的目的是運用虛擬化技術與電源管理方法達到節能的目標。我們專注在電源管理演算法的發展，並依此實作了一個綠能管理的虛擬雲端平台，並於上建置了應用程式服務並加以測試效能與電能節省的比較實驗。結果証明綠色電源管理(GPM)，在應用程式服務低利用率的狀況下，透過遷移虛擬機器的方式將服務集中，有效地節省電能。

關鍵字：綠能雲端計算，虛擬化，綠色電源管理

Abstract

With the government and the high development of electronic business, electronic services, the demand is sustained, rapid increase, they need to build more data centers to provide services, which will consume considerable power, especially when the service is running at low usage conditions, such as processor utilization of 10%, its total power consumption of at least 60% of the peak, making the idle resources, energy waste, which is caused by low energy efficiency of data centers. The purpose of this paper is to use virtualization technology and power management approach to achieve energy saving targets. We focus on the development of power management algorithms, and implements a virtual cloud of green cloud management platform, and build the web application in the above, and experimentns to test the performance and power saving. We prove that that Green Power Management (GPM), at low utilization state of services, effectly save energy by consolidation service via a method of Migrating VM.

Keywords: Green Cloud Computing, Virtualization, Green Power Management (GPM)

Acknowledgements

回想當初考進入東海資工時的種種情事，就像是才剛發生的一樣，入學後考慮要跟隨那一位教授進行學習，在偶然的機會下，學長兆弘給了我一條明確的道路，在他的推薦下加入高效能計算實驗室，參與一連串的研習。

學業完成在即，首先我要感謝指導教授—楊朝棟 博士，在學期間給予我最多耐心且最大支持與最真誠的鼓勵，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，何其有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，造就出的韌性及知識，以應付未來更多的挑戰。

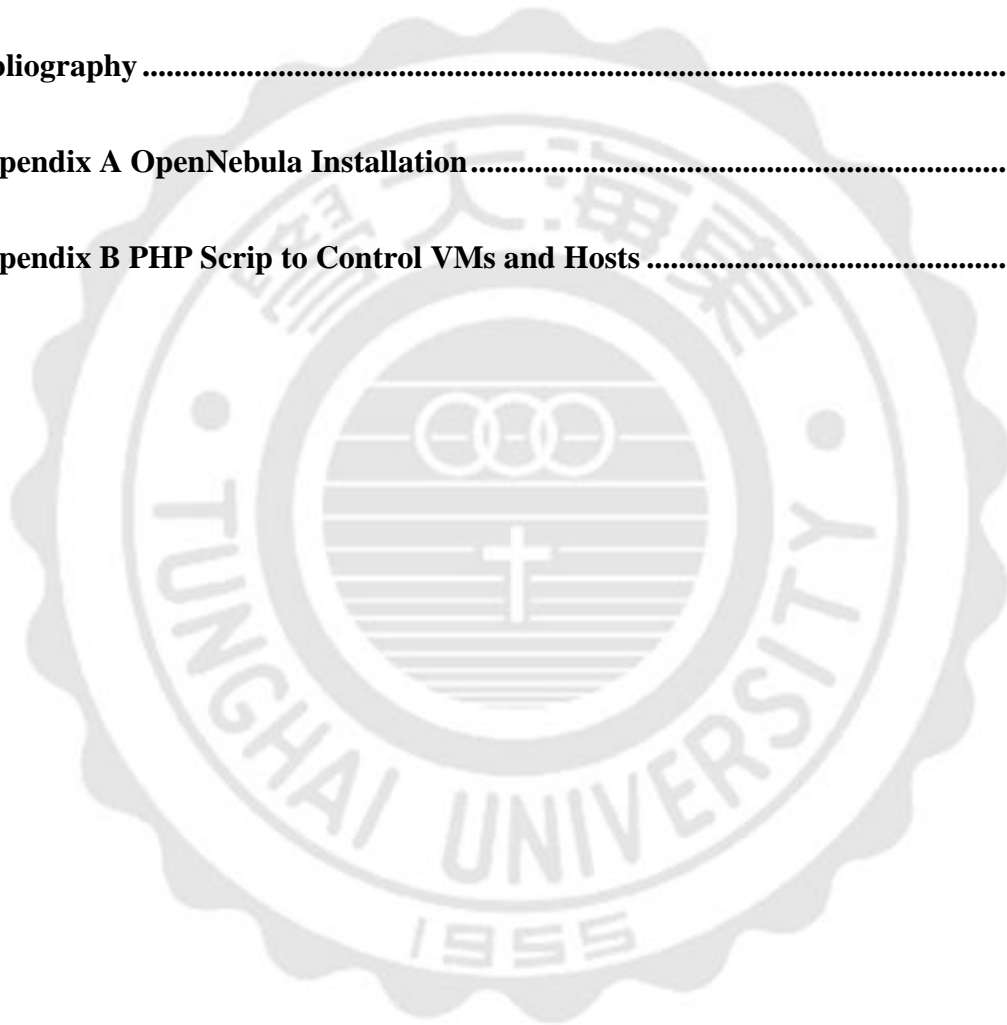
同時，我要感謝研究所合作的學長、同學們、還有翔耀、威利、政達等及提供支持，在數不清的假日裡，相約在 HPC 實驗室或同學們提供的場所，聚在一起做研究。

最後，也要感謝老婆-燕雯，在我就學期間，尤其是寫論文的時候，分擔照顧小孩的責任，沒有她的全力支持與體諒，我將無法順利完成論文，更遑論畢業取得碩士學位，沒有你們無私的付出，我將不可能完成我的研究工作，謝謝你們大家，並且再一次深深感謝楊老師的指導。

Table of Contents

摘要.....	iii
Abstract.....	iv
Acknowledgements	v
Table of Contents	vi
List of Tables	viii
List of Figures.....	ix
Chapter 1 Introduction.....	1
1.1 Motivation	1
1.2 Thesis Contribution	3
1.3 Thesis Organization.....	4
Chapter 2 Background Review	5
2.1. Cloud Computing	5
2.2. Virtualization	6
2.3. OpenNebula.....	11
2.4. Dynamic Resource Allocation Algorithm	13
2.5. Green Power Management	14
2.6. Related Works.....	15
Chapter 3 System Implementation	17
3.1 System Architecture	17
3.2 Performance and Networking Capability	18
3.3 Green Power Manager Algorithm	19

3.4 Management Interface	23
Chapter 4 Experimental Results	27
4.1 Experimental Environments	27
4.2 Experimental Results	28
Chapter 5 Conclusions and Future Work	33
Bibliography	34
Appendix A OpenNebula Installation	39
Appendix B PHP Scrip to Control VMs and Hosts	43



List of Tables

Table 3-1. Response Time Statistic	19
Table 3-2. Throughput Rate	19
Table 4-1. Lab Server Hardware Specification	27



List of Figures

Figure 2-1. Full Virtualization	7
Figure 2-2. Para-virtualization	9
Figure 2-3. OpenNebula Architecture.....	12
Figure 3-1. System Architecture	18
Figure 3-2. The Green Power Manager process	22
Figure 3-3. Web-based Interface.....	24
Figure 3-4. Virtual Machines Manager.....	24
Figure 3-5. CPU Performance.....	25
Figure 3-6. Memory Performance.....	25
Figure 3-7. GPM Setting Page	26
Figure 4-1. Experiment Environment.....	28
Figure 4-2. Shutdown Host and Ping VM.....	29
Figure 4-3. VM CPU Usage of Different Wattage Consumption	30
Figure 4-4. Power Monitor	31
Figure 4-5. VM CPU Usage.....	31
Figure 4-6. Power Consumption.....	32

Chapter 1

Introduction

1.1 Motivation

Cloud is actually refers to network, the name came from engineers in the schematic drawing, often represented by a cloud network. Therefore cloud services are the network services. Ones can be presented as a layered architecture that can be viewed as a collection of IT services referred to Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Among these type SaaS allows users to run applications remotely from the cloud. PaaS includes operating systems with a custom software stack for given application [1, 2]. IaaS refers to computing resources as a service, includes a virtualized computer that is dividing hard ware resources to unit. Each unit means one virtual machine lived on virtualization layer. Virtualization is the key technique to enable elastic computing delivering an infrastructure service.

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.

Current Web applications demand highly flexible hosting and resource provisioning solutions. The rising popularity of social network Web sites, and the current Internet users to store and share increasing amounts of pictures, movies, life-stories have required scalable infrastructure. Benefiting from economies of scale in Web technologies, data centres have as a key model to provision resources to Web applications and deal with their availability and performance requirements.

Cloud and Virtualization to accelerate not only accelerate the data center building, but also brings the huge energy consuming. Recent reports [3] indicate that energy costs are becoming dominant in the Total Cost of Ownership (TCO). In 2006, data centres represented 1.5 percent of the total US electricity consumption. The ever-increasing demand for cloud-based services does raise the alarming concern of data centre energy consumption. By 2011, the current data centre energy consumption could double [4] leading to more carbon emissions.

In the past, the only focus is the processing power of IT equipment and related equipment spending, while infrastructure, including power, cooling and data center space is always assumed available, ready, a given and affordable. Today, the infrastructure is becoming a limiting factor in the driving force behind this change comes from the growing business computing needs, the burden of rapidly rising energy costs, global warming, national energy security awareness [5-12]. Green computing, refer to support business computing needs as little as possible or sustainable computing power, has recently continued to be attention and has been growing [13-17].

Energy consumption of computation, storage and communications, cloud computing has recently received considerable attention, as a promising approach for

delivering ICT services by improving the utilization of data centre resources. When the data centres application based on a generic virtual machine, allowing the application workload combined in a smaller number of virtual machines, which can help more efficient use of resources. If workload size could allocate in different resource depend time and space, it could improve the energy efficiency and avoiding waste resource [18]. At this thesis, we will introduce the Green Power Manager (GPM) algorithm for the virtual machines on cloud computing environments. GPM is based on work requirements, dynamically adjust the available resources, the purpose is to save energy and maintain the appropriate computational efficiency. When the CPU utilization increases, another physical machine will be waked up to join Resource pools. If the CPU usage is at low utilization, then the physical machine will be closed.

Cloud computing data center for maximize profits of the key issues is reducing power consumption. For reach the goal, there are two direction, follow the infrastructure or the application point of view to reduce energy consumption. There are several ongoing research on the development of an integrated management system for the cloud to provide comprehensive on-line monitoring use of resources and enforcement powers conscious policy to reduce energy consumption [19-25].

1.2 Thesis Contribution

We develop the GPM mechanism and implement about loading balance on the virtual machine to concentrate process, it first determines the minimum of physical work load on the server into the way as CPU Usage to meet application performance requirements. Then shutting down unneeded servers in order to reduce server energy consumption, once the application requires more physical resources can automatically

turn on the server. The process is automatic and continuous. To use of GPM will have the opportunity to save electricity costs and reduce carbon footprint to achieve energy conservation purposes.

1.3 Thesis Organization

In this thesis, the Green Power Management (GPM) for loading balance approach was proposed by our thesis and in this thesis it including three main phrases: (1) The Virtualization management. (2) The Dynamic Resource Allocation mechanism. (3) The Green Power Management approach. This thesis is organized as follows. First in Section II we introduce the background and related works, section III describe the system design and more detail of entire system, in section IV we show up the experiment and results, and finally section V outline the main conclusions and the future work.

Chapter 2

Background Review

2.1. Cloud Computing

Energy consumption in hosting Internet services is becoming a pressing issue as these services scale up. Dynamic server provisioning techniques are effective in turning off unnecessary servers to save energy. Such techniques, mostly studied for request-response services, face challenges in the context of connection servers that host a large number of long-lived TCP connections [26]. In this thesis, we show that our algorithms can save a significant amount of energy without sacrificing user experiences. Consolidation of applications in cloud computing environments presents a significant opportunity for energy optimization. The goal of energy aware consolidation is to keep servers well utilized such that the idle power costs are efficiently amortized but without taking an energy penalty due to internal contentions.

There are several different issues; first of all, the merger must be carefully considered combination of different workload on a common physical suitability of the host. Therefore, understanding the nature of the work to determine which components of critical workloads can be packaged together. Second, there is a performance and energy optimization. This is because it can cause performance degradation, leading to increased execution time, which would eat up the energy derived from the lower idle energy savings [26]. In addition, there are many problems affecting the integration, including the behavior of servers and workloads, the performance from the implementation of change and the optimal combination of different applications that

can accept the optimal solution does not interrupt the work load to keep track of changes, which become important the integration of energy efficiency [27].

2.2. 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 [28].

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

Full virtualization (also called native virtualization) 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 2-1.

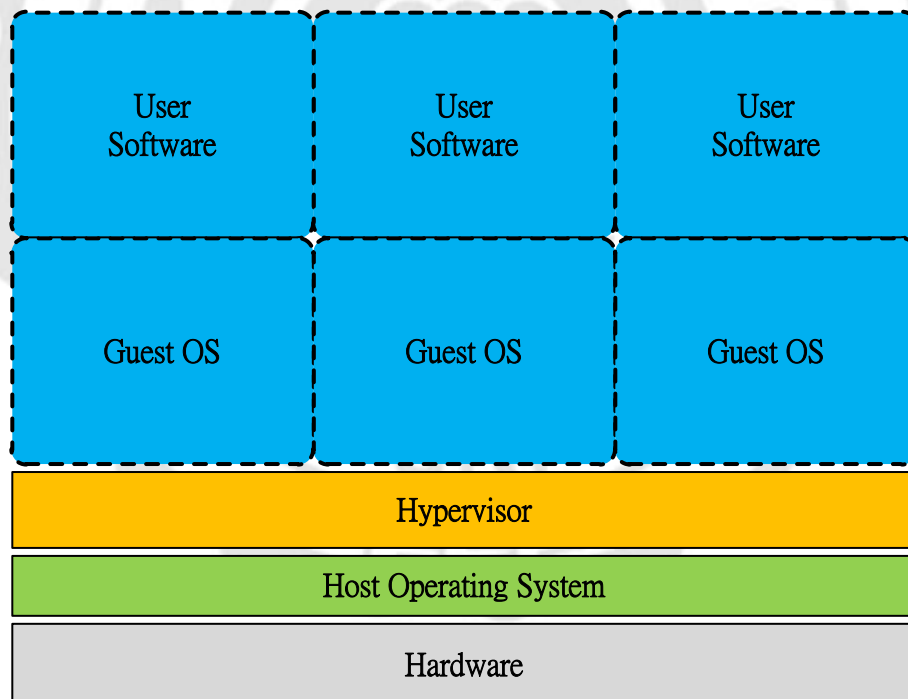


Figure 2-1. Full Virtualization

Para-virtualization: In some instances this technique is also referred to as enlightenment. In Para-virtualization, 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.

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 2-2.

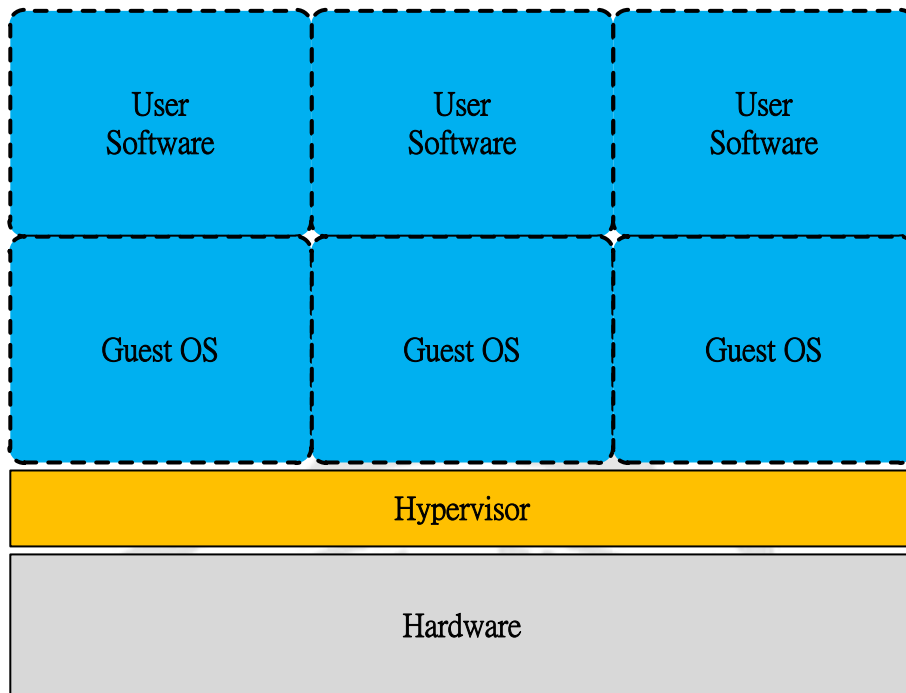


Figure 2-2. Para-virtualization

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.

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 [29].

VMs can run on a single hardware unit (server consolidation). Therefore, less hardware is needed overall, thus reducing energy wasted for cooling, while the deployed hardware utilization increases. Consolidating hardware and reducing redundancy can achieve energy efficiency. Unused server can be turned off (or hibernated) to save energy. Some hardware gets higher load, which reduces the number of physical servers needed.

2.3. OpenNebula

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 [30]. 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.

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 [30]. The architecture of OpenNebula can be described as Figure 2-3.

Live migration is the movement of a virtual machine from one physical host to another while continuously powered-up. When properly carried out, this process takes place without any noticeable effect from the point of view of the end user. Live migration allows an administrator to take a virtual machine offline for maintenance or upgrading without subjecting the system's users to downtime. 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 the fact that it facilitates proactive maintenance. If an imminent failure is suspected, the potential problem can be resolved before disruption of service occurs. Live migration can also be used for load balancing, in which work is shared among computers in order to optimize the utilization of available CPU resources.

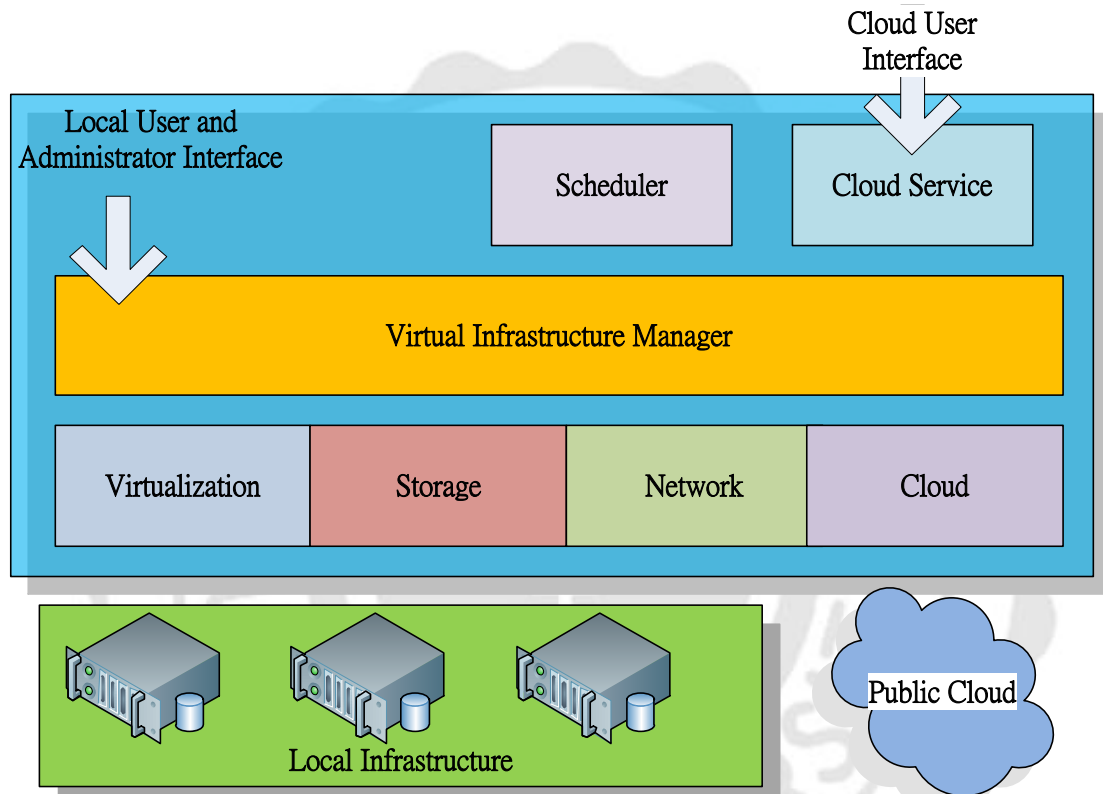


Figure 2-3. OpenNebula Architecture

However the OpenNebula lack a GUI management tool. In pervious works we build virtual machines on OpenNebula and implemented Web-based management tool. Thus, the system administrator can be easy to monitor and manage the entire OpenNebula System on our project. OpenNebula is composed of three main components: (1)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 monitoring interface for the physical hosts (2) the

Capacity Manager governs the functionality provided by the OpenNebula core. The capacity manager adjusts the placement of VMs based on a set of pre-defined policies (3) Virtualizer Access Drivers. In order to provide an abstraction of the underlying virtualization layer, OpenNebula uses pluggable drivers that expose the basic functionality of the hypervisor [31].

2.4. Dynamic Resource Allocation Algorithm

In our previous thesis “A Dynamic Resource Allocation Model for Virtual Machine Management on Cloud” published Dynamic Resource Allocation (DRA) algorithm which has a detail description of DRA [41], and it is one of key components of the this thesis basis. It 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 [32] cannot accomplish load balance. When a part of virtual machines load increasing, it will affect all virtual machine on the same host machine. DRA can overcome this obstacle, and improve host machine performance. DRA 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.

In this thesis we focus on enhance virtualization saving energy, therefore DRA is not described in detail in this thesis; if you are interested in DRA please refer to “A Dynamic Resource Allocation Model for Virtual Machine Management on Clusters” article. However, the purpose of DRA is to reach the best balance between each physical machine. To avoid computing resources centralized on some specify physical

machines, how to balance the resources is most important issue. To achieve the maximum efficiency the resource must be evenly distributed.

2.5. Green Power Management

DRA manages the allocation of resources to a set of virtual machines running on a cluster hosts with the goal of fair and effective use of resources. One makes virtual machine placement and migration recommendations that serve to enforce resource-based service level agreements, user-specified constraints, and maintain load balance across the cluster even as workloads change.

GPM saves power by dynamically right-sizing cluster capacity according to workload demands. One recommends the evacuation and powering of hosts when CPU is lightly utilized. GPM recommends powering hosts back on when either CPU utilization increases appropriately or additional host resources are needed to meet user-specified constraints. GPM executes DRA in a what-if mode to ensure its host power recommendations are consistent with the cluster constraints and objectives being managed by DRA.

Hosts powered off by GPM are marked in standby mode, indicating that they are available to be powered on whenever needed. GPM can be awakened from a powered-off (ACPI S5) state via either wake-on-LAN (WOL) packets. WOL packets are sent by the front-end host in the cluster, so GPM keeps at least one such host powered on at all times.

2.6. Related Works

Pervious works for data centres power saving, use turning servers on and off based on demand [6]. X.Wang and M.Chen use dynamic voltage and frequency scaling (DVFS) based on well-established control theory [33]. K.Rajamani and C.Lefurgy propose spare servers to manage power [34].

Internet-based services have recently expanded to include network-based storage and network-based computing. These new services are being offered both to corporate and individual end users. J. Baliga analysis considered both public and private clouds and included energy consumption in switching and transmission as well as data processing and data storage. And after his analysis, he found the number of users per server is the most significant determinant of the energy efficiency of a cloud software service [35].

In power management area, Z. Wu and J. Wang presented a control framework of tree distribution for power management in cloud computing so that power budget can be better managed based on workload or service types [36].

Additionally, R. S. Montero [37] 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.

R.S Chang and C.M Wu present a routing algorithm with energy awareness. It estimate the energy consumed by the network component to decide the packet forwarding route. On the other hand, it designs a energy aware provision algorithm to decrease the energy consumed by the virtual network [38].

H. Abdelsalam and K. Maly take prior service level agreement request when determining time slots in which changes should take place [39]. S. Srikantaiah and A.

Kansal take consolidation issue as a modified multi-dimensional bin-packing problem of allocating and migrating workloads to achieve energy optimal operation [18].

This thesis focuses power management allocation on physical machines with virtual machines. And we presented a green power management mechanism for this. For more detail please see section III.



Chapter 3

System Implementation

3.1 System Architecture

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. [40]

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 physical infrastructure consists of five hosts (Host0 to Host4), which are interconnected by a Gigabit Ethernet LAN. Each physical host node has a four core 2.8 GHz i7 processor, 4GB of RAM, 500 gigabytes disk, Debian operating system, and the network connected to a gigabit switch.

The Host0 acts as the front-end of the physical pool and is also connected to the Internet. This host runs the Open-Nebula engine, which has the ability to deploy, manage and monitor local VMs on any physical pools (using the XEN hypervisor).

The deployment of VMs by OpenNebula can be controlled manually or can be done automatically by GPM mode. In this case, GPM monitor continuously every CPU core usage per physical host to a given threshold. When this limit is reached and

the VM on the physical host needs migrate to another. The virtual computing cluster consists of a front-end node and a variable set of worker nodes, This cluster front-end acts also as NFS for every worker node in the cluster.

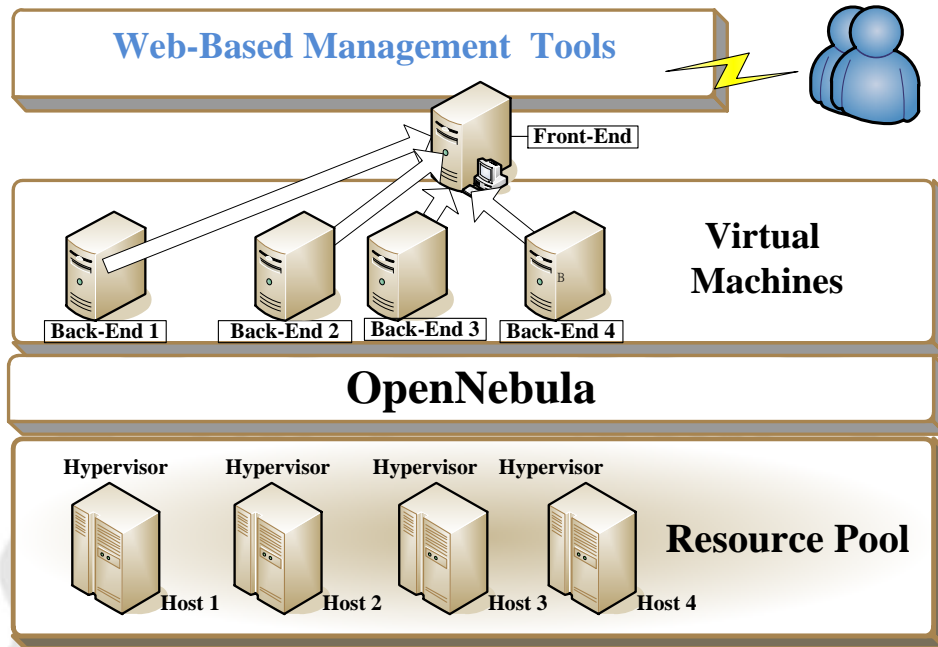


Figure 3-1. System Architecture

3.2 Performance and Networking Capability

JMeter can load and performance test Web server. Here we use to test web application by simulation of concurrent users accessing static HTTP pages, test throughput and response time, as shown in Table 1. According to the measure result, the response time is significant raise after threads increase. JMeter can be simulated multi-threading, and set the number of times the loop to obtained average. We could get a basic numbers of performance and network capabilities as shown in Table 2 for its throughput, which can be interpreted as the server load and performance.

Table 3-1. Response Time Statistic

<i>Request Loop Count</i>		<i>HTML Access Threads (About 20 Bytes)</i>			
		10	50	200	500
WAN	200	624ms	667ms	838ms	1711ms
LAN	200	44 ms	139 ms	490ms	987 ms

Table 3-2. Throughput Rate

<i>Request Loop Count</i>		<i>Throughput Rate</i>			
		10	50	200	500
WAN	200	21.4/sec	113.3/sec	244.5/sec	254.7/sec
LAN	200	369.9/sec	368.1/sec	360.9/sec	355.3 /sec

3.3 Green Power Manager Algorithm

GPM algorithm archives energy saving based on the load balance environments of building on DRA. GPM is based on work demands, do not emphasize the maximum efficiency to the allocation of resources. DRA is based the maximum performance to allocate of resources. Now we descript GPM algorithm as follow.

First it define a Avg of total host loading ratio ($Load_{\theta}$). Total hosts means total available resources for allocation. The purpose of this parameter is to understand the current work requirements have been properly met, or over satisfied, or insufficient to meet. calculation is as follows:

$$Load_{\theta} = \frac{\sum_{i=1}^n (\sum_{p=1}^m \text{Current Core}_p \text{ Usage on Host}_i)}{\sum_{i=1}^n \text{Cpu Capacity on Host}_i} \quad (1)$$

Here, The parameter i is host number, the number of physical machines available for allocation, p is the number of core on the $Host_i$. The molecular of $Load_{\theta}$ is total usage summary of all host, the denominator represents total CPU capacity of all Host. The CPU capacity on $Host_i$ is calculated below:

$$\text{Total Cpu Capacity} = \sum_{p=1}^m \text{Core}_p \text{ on Host}_i * 100 \quad (2)$$

Suppose a host has eight cores, it's computing power total is 800. We only monitor the physical CPU, not only reducing the additional cost of computing but also simplify the implementation and design. In such an environment, each entity machine computing performance should be same as much as possible, because we expressed as a ratio for CPU loading. In additional, each of VM on OpenNebra is single-core.

Parameter $Load_{\theta}$ provides GPM consistent basis of comparison, refers to be compared under the different size of resource. let's define two critical value of loading: λ is maximum tolerance loading ratio, β is minimum critical loading ratio. Take two critical values to compared with $Load_{\theta}$ to determine the system is currently in the three states: The first is supply not enough, means computing resources can not meet the work for demand, which will cause the calculation bottleneck, and work can't complete on the appropriate time, the efficiency is poor. Second is supply appropriate, computing resources match the size for demand, which is the optimal state, energy efficient. Third is oversupply, demand for computing resources far greater than the work, this state resulting in waste of resources and energy efficiency is also poor.

When the $Load_{\theta}$ touch to a maximum threshold, the system will open a new

physical host, increasing computing pool, then automatically loading balance by the DRA, VMs averagely allocate to all hosts, to avoid resource concentrated in a few of the physical machine. When the $Load_{\theta}$ touch to a minimum threshol, GPM would shut down the $Host_k$, as minimum CPU usage host among of available host for allocation. $Host_k$ calculation is as follow

$$Host_k = \min\left(\frac{\sum_{i=0}^n Host_i \text{ Current Cpu Usage}}{\sum_{i=0}^n Host_i \text{ Total Cpu Capacity}}\right) \quad (3)$$

Shutdown $Host_k$ before, the system move the VM on the $Host_k$ to other hosts. Move order is by workload size of VM. Move to host is the second-lowest ones, the moving mechanism by DRA.

There is a example to show flow: Suppose there are n virtual machines and the $Load_{\theta}$ is greater than the λ , it shows the loading on physical machine is too much, and then GPM will awake a new a host and apply the DRA to do load balancing. If the $Load_{\theta}$ is small then β . It expressed resource utilization in most of the time is idle state. So it needs to be turn off the one of the booted hosts. GPM mechanism will decide which one should be shut down. Once target host have determined. The virtual machines on target ones would migrate averagely to the others host, then shut down the target host to attain the purpose of energy saving.

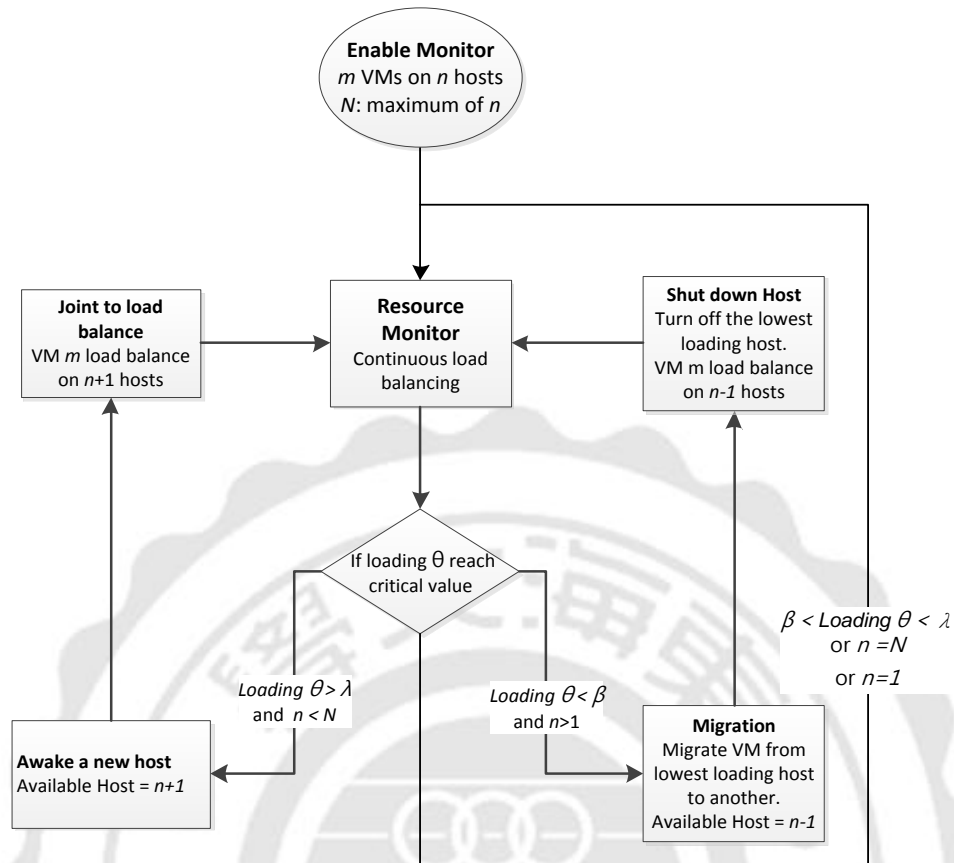


Figure 3-2. The Green Power Manager process

In Figure 3-2, it demonstrates GPM process. At the beginning, there are m virtual machines on n hosts (n as booted hosts, N as maximum of n) and resource monitor detects the loading change continuously. There are three circumstances. The first is loading between β (minimum critical ratio of loading) and λ (maximum tolerance ratio of loading), in this state, m VMs do load balance continuously on n hosts. The Second is loading greater than λ and n less than N (if booted hosts equal N , do nothing), then GPM awake a new host to join resource pool, in this state, there are m VMs on $n+1$ host. The third is loading less than β and n greater than 1 (if booted host is only one, do nothing), then the VM on the lowest loading host will be migrated to others, and then turn off the lowest loading host, in this state, there are m virtual machines on $n-1$ host.

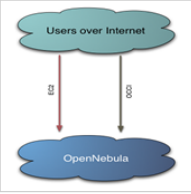
The algorithm is very simple and performs some calculations for monitoring info. Our pseudo code is as follows:

```
[Initialization]
Define new section per min
   $Load_{\theta}$  = Avg of total host loading ratio (avg per 5 min)
  If  $Load_{\theta} > \lambda$  then
    [Open another physical machine]
    Send magic packet to awake host on shutdown status.
    Determine the target VM by DRA mechanism.
    Move the target VM machine to just awaked-host.
     $DRA(Host_1, Host_2,..Host_{n+1})$ 
  Else if  $Load_{\theta} < \beta$ 
    [Shutdown physical machine]
    Determine the minimum load of the host
     $DRA(Host_1, Host_2,..Host_{n-1})$ 
  End if
  [Confirm how many resources available at next loop]
   $N$  = maximum physical host.
End new section.
```

3.4 Management Interface

We design a useful web interface for end users fastest and friendly to Implementation virtualization environment. In Figure 3-3, 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.

Login



User Name

Password

Login

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 3-3. Web-based Interface

The entire web-based management tool including physical machine management, virtual machine management and performance monitor. In Figure 3-4 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 physic 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.

::::Create VM::::

VM Name Give a name for Virtual Machine

IP Address Give a physical ip address

Memory Size 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 3-4. Virtual Machines Manager

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. In this thesis we use RRDtool to monitor entire system. Figure 3-5 and Figure 3-6 show current physical machines CPU and memory usage.

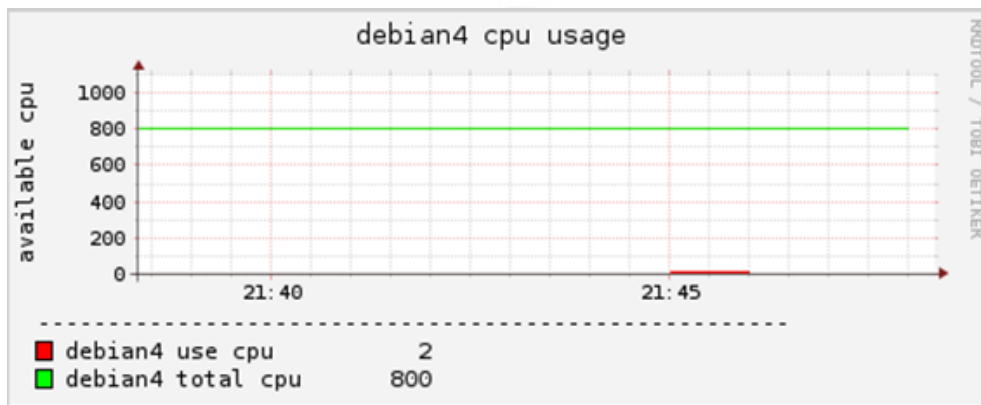


Figure 3-5. CPU Performance

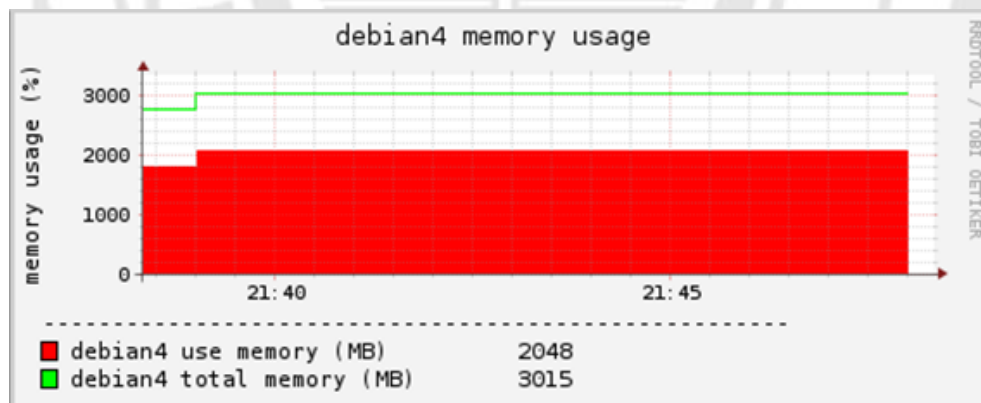


Figure 3-6. Memory Performance

Figure 3-7 is a GMP setting page. it show which hosts controlled by OPENNEBULA currently, which ones have enabled the GPM mechanism, hosts state of GPM. Moreover once the host enables the GPM, the system will control the VM on the HOST automatically and start loading balance



Host Name	Description	GPM State	Enable	Host Status of GPM
debian1	OS: Debian 5.0	On	Yes <input type="radio"/> No <input type="radio"/>	Running
debian2	OS: Debian 5.0	On	Yes <input type="radio"/> No <input type="radio"/>	Sleep
debian3	OS: Debian 5.0	On	Yes <input type="radio"/> No <input type="radio"/>	Sleep
debian4	OS: Debian 5.0	Off	Yes <input type="radio"/> No <input checked="" type="radio"/>	-
debian5	OS: Debian 5.0	Off	Yes <input type="radio"/> No <input checked="" type="radio"/>	-

Figure 3-7. GPM Setting Page



Chapter 4

Experimental Results

In this section, we show the results of our efforts. First, we introduce the experimental environment, next is the service was not interrupted. Finally, the results show the mechanism of GPM.

4.1 Experimental Environments

In our experimental environment each server has same specification. We give a table list as shown [錯誤! 找不到參照來源。](#), it described our servers CPU, Memory and storages capabilities.

Table 4-1. Lab Server Hardware Specification

<i>Hardware Lists</i>						
No	Model	Cores	CPU MHz	Disk (Giga)	Memory (Giga)	Comments
1	Intel(R) Core(TM) i7 CPU 860@2.80GHz	4	2,800	500	4	Front-End
2	Intel(R) Core(TM) i7 CPU 860@2.80GHz	4	2,800	500	4	Back-End
3	Intel(R) Core(TM) i7 CPU 860@2.80GHz	4	2,800	500	4	Back-End
4	Intel(R) Core(TM) i7 CPU 860@2.80GHz	4	2,800	500	4	Back-End

4.2 Experimental Results

Node_01 and Node_02 are the hosts. They act as backend and sharing storage. VM_01 is living on Node_01 and provides service to external user. The whole lab environment is as shown illustration as Figure 4-1. We ping service IP on VM_01 continuously. First we migrate VM_01 from Node_01 to Node_02, then power off Node_01. It doesn't cause the service IP stop providing service. You can see the connection status of service IP in Figure 4-2.

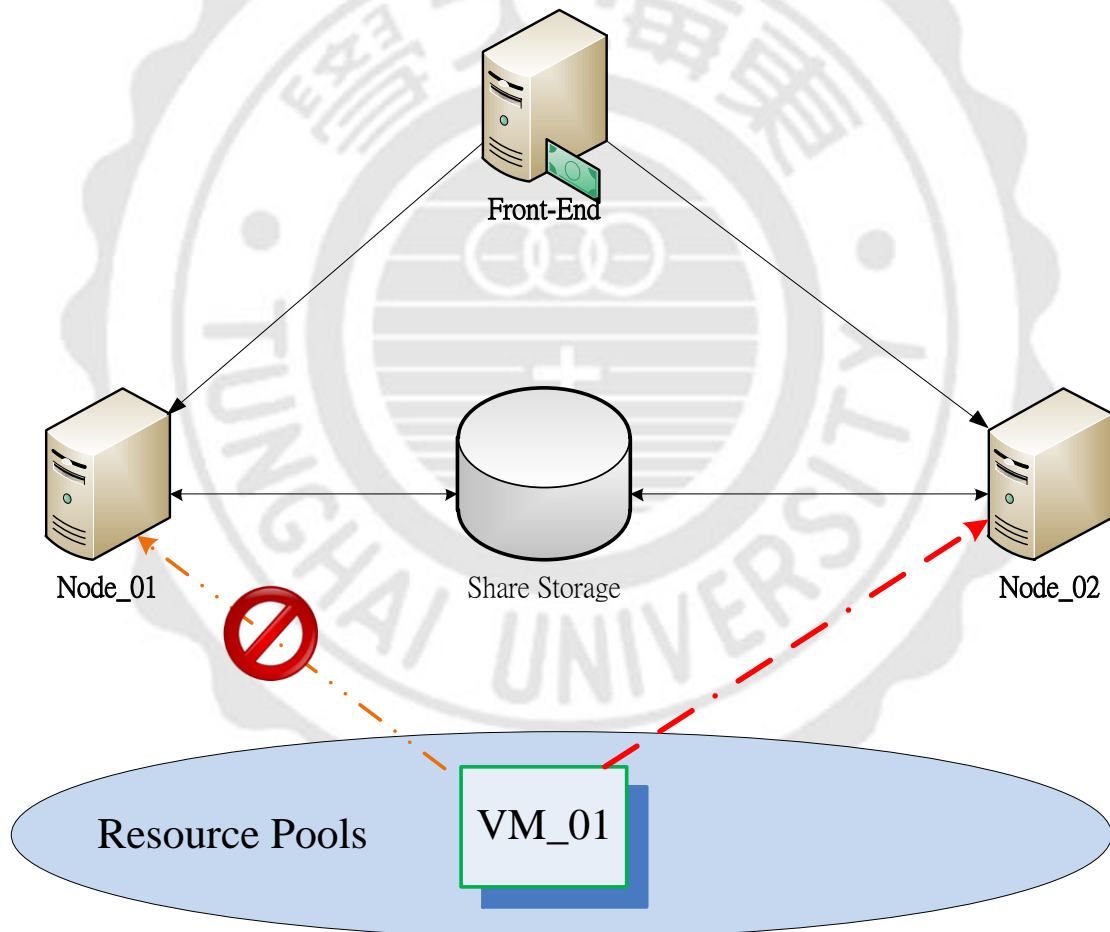


Figure 4-1. Experiment Environment.

```

C:\Windows\system32\cmd.exe - ping 140.128.102.192 -t
回 140.128.102.192: 位元組=32 時間=60ms TTL=56
覆 140.128.102.192: 位元組=32 時間=79ms TTL=56
自 140.128.102.192: 位元組=32 時間=98ms TTL=56
自 140.128.102.192: 位元組=32 時間=121ms TTL=56
自 140.128.102.192: 位元組=32 時間=113ms TTL=56
自 140.128.102.192: 位元組=32 時間=27ms TTL=56
自 140.128.102.192: 位元組=32 時間=31ms TTL=56
自 140.128.102.192: 位元組=32 時間=29ms TTL=56
自 140.128.102.192: 位元組=32 時間=24ms TTL=56
自 140.128.102.192: 位元組=32 時間=23ms TTL=56
自 140.128.102.192: 位元組=32 時間=25ms TTL=56
自 140.128.102.192: 位元組=32 時間=30ms TTL=56
自 140.128.102.192: 位元組=32 時間=32ms TTL=56

```

Figure 4-2. Shutdown Host and Ping VM

Next, we compare the VM CPU usage wattage consumed. Experimental design have four HOST, each open two VM, the configuration of each VM as a single-core, 1G RAM, a total of eight VM, each VM running HPCC, problem size about ten, for purpose is to get energy consuming data of full VM CPU usage. Figure 4-3 as VM CPU usage of different wattage consumption chart. X-axis is the number of VM, Y-axis is the wattage consuming. Our objective is to obtain the VM CPU power consuming, so the number of wattage of Y-axis is deducted the number of wattage of physical host idle (about 580 watt total aggregate consist of 4 hosts) which means the host in the boot status and had deployed virtual environment. The triangle mark of line is energy consuming when the VM turned on and not running any programs. We find at time only deploy the VM and not running any program will increase the number of wattage slightly more than ones not deployment. Rhombus symbol line represents the wattage consumption when VM CPU use fully. Each VM at fully running state will increase about 27(W) energy consumption more than the idle ones.

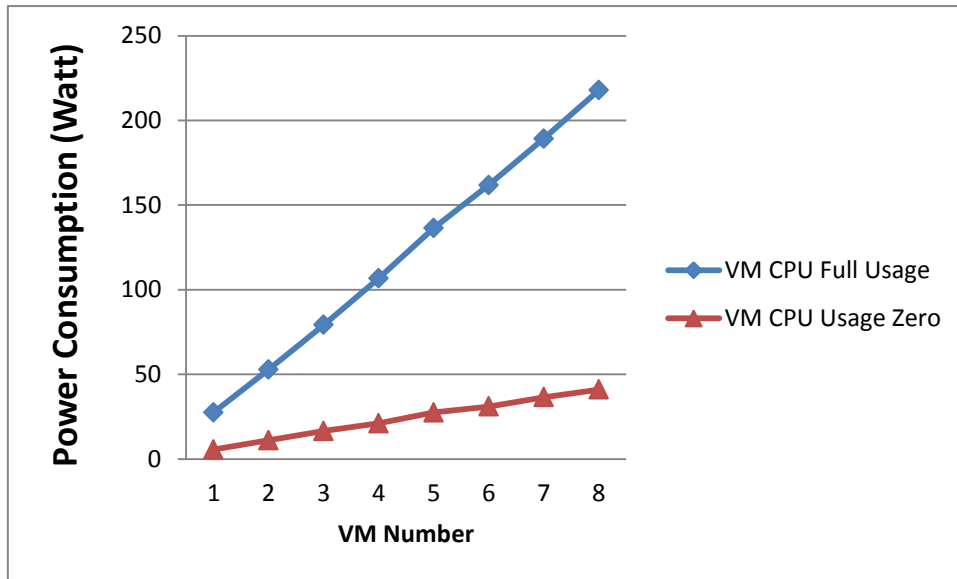


Figure 4-3. VM CPU Usage of Different Wattage Consumption

Furthermore we built up application servers, including the computing service, teaching website, multi-media services contain compression and decompression of media files on virtual environment in HPC lab at Tunghai University. The system architecture has showed in Figure 3-1. All the services are composed of four physical machines. All of them are on power distribution unit (PDU). A PDU is a device fitted with multiple appliance outlets designed to distribute electric power. It continuously monitors instant wattage consumption for four physical machines (Figure 4-4). We could observe that changes in wattage over at least 400. The four physical machines as OpenNebula client opened in the above total of four VM. Each VM provide an application service. In Figure 4-5, within one month, record the four VM CPU averagely total usages per hour. X-axis is time interval (hours), Y-axis is the four VM CPU total usage, here we use the SNMP protocol record the VM CPU usage per hour. We can find between 2 AM to 7 AM, CPU was at lower utilization, 10 AM to 16 PM is relatively high, so VM need more physical resources the interval between 10 AM to 16 PM.

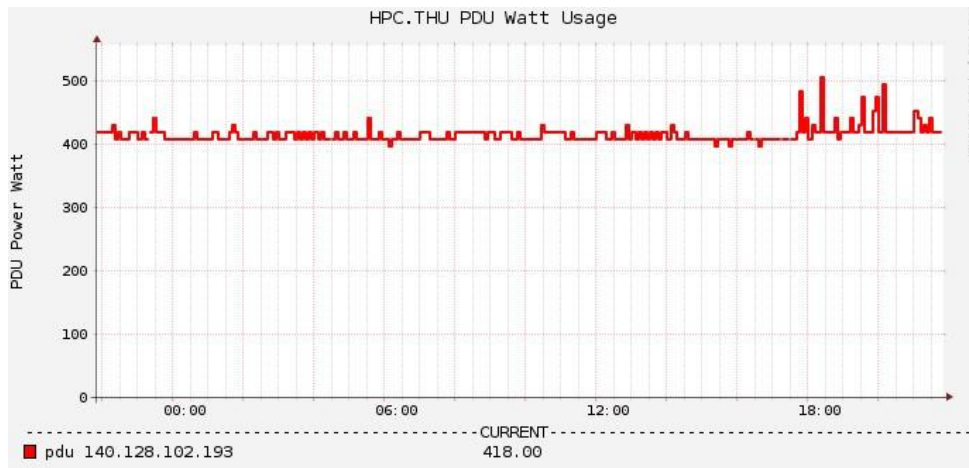


Figure 4-4. Power Monitor

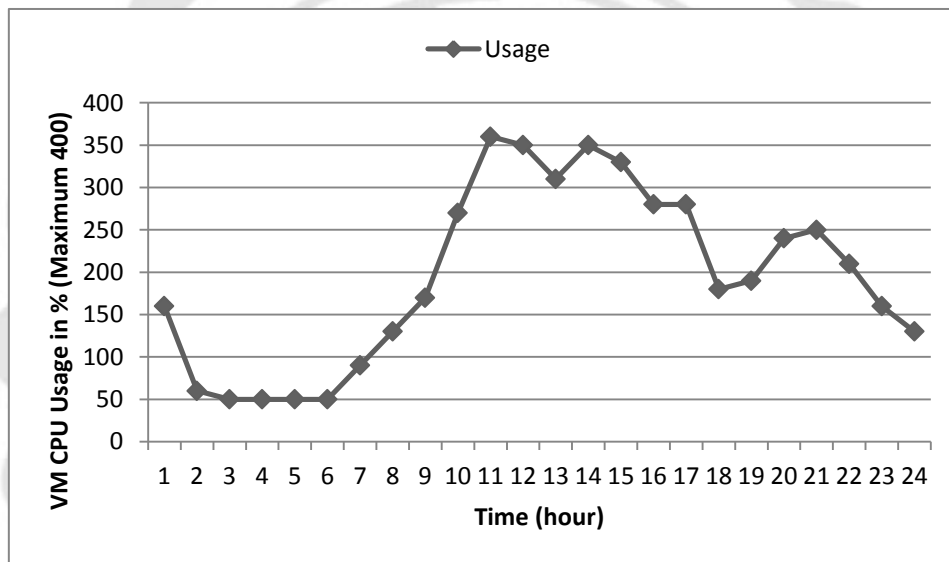


Figure 4-5. VM CPU Usage

Figure 4-6 has same measuring period as Figure 4-5, it records averagely total power consumption per hour. X-axis is time (hours), Y-axis as the power consumption total of the four physical machines (watts). The illustration shows that in the case turn off GPM, four machines are in power always, the power consumption has been over the 400W (Diamond marker), but contrast 2 AM to 7 AM of Figure 4-6, VM CPU demand volume is relatively small, so the decision-making based on GPM, front-end would migrate VM to the same physical machine, and others physical machines shut down to save energy. When the period in 10 AM to 16 PM, GPM was aware of VM

CPU demand increasing to exceed a single physical machine can supply, front-end wake up another machine by WOL technology and load balance automatically. System can power or shut down the physical machine according to computing demand, effectively achieve the purpose of saving energy.

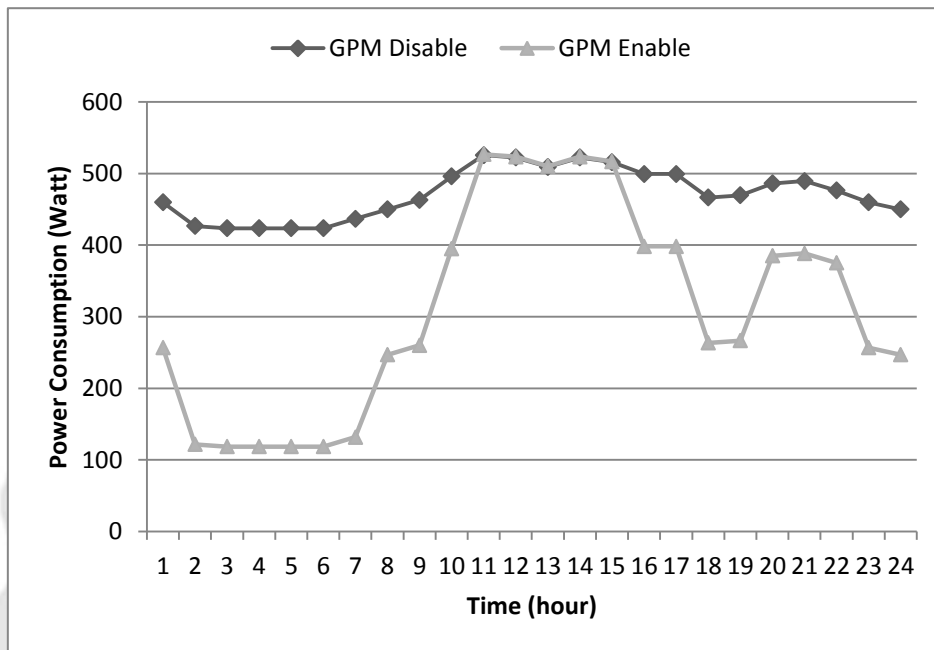


Figure 4-6. Power Consumption

Chapter 5

Conclusions and Future Work

In this work we have presented an optimization with green power management model for virtualization platform that allows a flexible management of these computing platforms including: (1) supporting GPM mechanism; (2) implementation Resource Monitor on OpenNebula web-based interface; and (3) based on DRA and OpenNebula advantage, instead of booting physical machines with schedule traditionally .

Moreover, we expect to improve violent CPU highly loading solution. Because in our thesis, we assume a prefect smooth virtual machines changes not a dramatic changes. For instance, set sensitivity parameters for entire mechanism or etc. However under our GPM approach, it already got a signification energy saving than traditional approach.

Bibliography

- [1] L. Wang, J. T. Kunze, M. Castellanos, A. C. Kramer, and D. Karl, "Scientific Cloud Computing: Early Definition and Experience " presented at the High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference Dalian 2008.
- [2] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," *Internet Computing, IEEE*, vol. 13, pp. 10-13, 2009.
- [3] M. K. Patterson, D. G. Costello, M. Loeffler, and P. F. Grimm, "Data center TCO; a comparison of high-density and low-density spaces," 2007.
- [4] S. V. L. Group, "Data Center Energy Forecast," July 29 2008.
- [5] D. Wang, "Meeting Green Computing Challenges," in *High Density packaging and Microsystem Integration, 2007. HDP '07. International Symposium on*, 2007, pp. 1-4.
- [6] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," presented at the Proceedings of the eighteenth ACM symposium on Operating systems principles, Banff, Alberta, Canada, 2001.
- [7] George and Lawton. (2007) Powering Down the Computing Infrastructure. 16-19. Available: <http://doi.ieeecomputersociety.org/10.1109/MC.2007.69>
- [8] D. Wang, "Cooling challenges and best practices for high density data and telecommunication centers," in *High Density Microsystem Design and Packaging and Component Failure Analysis, 2006. HDP'06. Conference on*, 2006, pp. 49-54.
- [9] M. J. Pawlish and A. S. Varde, "A decision support system for green data

- centers," presented at the Proceedings of the 3rd workshop on Ph.D. students in information and knowledge management, Toronto, ON, Canada, 2010.
- [10] X. Zhang, X.-n. Zhao, Y. Li, and L.-j. Zeng, "Key Technologies for Green Data Center," presented at the Proceedings of the 2010 Third International Symposium on Information Processing, 2010.
- [11] J. B. Carter, "A look inside IBM's green data center research," presented at the Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design, San Francisco, CA, USA, 2009.
- [12] W.-c. Feng, X. Feng, and R. Ge, "Green Supercomputing Comes of Age," *IT Professional*, vol. 10, pp. 17-23, 2008.
- [13] R. R. Harmon and N. Auseklis, "Sustainable IT services: Assessing the impact of green computing practices," in *Management of Engineering & Technology, 2009. PICMET 2009. Portland International Conference on*, 2009, pp. 1707-1717.
- [14] L. Andr, Barroso, U. H, and Izle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, pp. 33-37, 2007.
- [15] C. E. Bash, "Sustainable IT ecosystems and data centers," presented at the Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design, San Francisco, CA, USA, 2009.
- [16] R. Harmon, H. Demirkan, N. Auseklis, and M. Reinoso, "From Green Computing to Sustainable IT: Developing a Sustainable Service Orientation," presented at the Proceedings of the 2010 43rd Hawaii International Conference on System Sciences, 2010.
- [17] R. Harmon and H. Demirkan, "The Next Wave of Sustainable IT," *IT Professional*, vol. 13, pp. 19-25, 2011.
- [18] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud

- computing," presented at the Proceedings of the 2008 conference on Power aware computing and systems, San Diego, California, 2008.
- [19] V. Hien Nguyen, F. D. Tran, and J. M. Menaud, "Performance and Power Management for Cloud Infrastructures," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 329-336.
- [20] G. von Laszewski, W. Lizhe, A. J. Younge, and H. Xi, "Power-aware scheduling of virtual machines in DVFS-enabled clusters," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 2009, pp. 1-10.
- [21] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, 2007, pp. 119-128.
- [22] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application Performance Management in Virtualized Server Environments," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, 2006, pp. 373-381.
- [23] M. N. Bennani and D. A. Menasce, "Resource Allocation for Autonomic Data Centers using Analytic Performance Models," in *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, 2005, pp. 229-240.
- [24] W. XiaoYing, L. DongJun, W. Gang, F. Xing, Y. Meng, C. Ying, and W. QingBo, "Appliance-Based Autonomic Provisioning Framework for Virtualized Outsourcing Data Center," in *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, 2007, pp. 29-29.
- [25] L. Wang, G. v. Laszewski, J. Dayal, and F. Wang, "Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with

- DVFS," presented at the Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010.
- [26] A. Berl, E. Gelenbe, M. d. Girolamo, and G. Giuliani, "Energy-Efficient Cloud Computing," *The Computer Journal*, vol. Vol. 53, No. 7, pp. pp. 1045-1051, 1 September 2010.
- [27] S. Heo, K. Barr, and K. Asanovi, "Reducing power density through activity migration," presented at the Proceedings of the 2003 international symposium on Low power electronics and design, Seoul, Korea, 2003.
- [28] W. v. Hagen, *Professional Xen Virtualization*, 2008.
- [29] C. T. Yang, C. H. Tseng, K. Y. Chou, and S. C. 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.
- [30] OpenNebula. Available: <http://www.opennebula.org>
- [31] J. W. Jang, E. Seo, H. Jo, and J.-S. Kim, "A low-overhead networking mechanism for virtualized high-performance computing systems," *The Journal of Supercomputing*, 2010.
- [32] Eucalyptus. Available: <http://open.eucalyptus.com>
- [33] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, "SHIP: Scalable Hierarchical Power Control for Large-Scale Data Centers," presented at the Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques, 2009.
- [34] K. Rajamani and C. Lefurgy, "On evaluating request-distribution schemes for saving energy in server clusters," presented at the Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software, 2003.

- [35] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," *Proceedings of the IEEE*, vol. 99, pp. 149-167, 2011.
- [36] Z. Wu and J. Wang, "Power Control by Distribution Tree with Classified Power Capping in Cloud Computing," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, 2010, pp. 319-324.
- [37] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Elastic management of web server clusters on distributed virtual infrastructures," *Concurrency and Computation: Practice and Experience*, pp. n/a-n/a, 2011.
- [38] C. Ruay-Shiung and W. Chia-Ming, "Green virtual networks for cloud computing," in *Communications and Networking in China (CHINACOM), 2010 5th International ICST Conference on*, 2010, pp. 1-7.
- [39] H. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, and D. Kaminsky, "Towards Energy Efficient Change Management in a Cloud Computing Environment," presented at the Proceedings of the 3rd International Conference on Autonomous Infrastructure, Management and Security: Scalability of Networks and Services, Enschede, The Netherlands, 2009.
- [40] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing*, vol. 13, 2009.
- [41] C.T. Yang, et al., "A Dynamic Resource Allocation Model for Virtual Machine Management on Cloud," in *Symposium on Cloud and Service Computing 2011*

Appendix A

OpenNebula Installation

A. Installation

We are using the OpenNebula 2.0 (beta1) release, which comes as a single .deb package.

Head-node

Install pre-requisite packages, using the command:

```
$sudo apt-get install libcurl3 libmysqlclient16 libruby1.8 libsqlite3-ruby  
libsqlite3-ruby1.8 libxmlrpc-c3 libxmlrpc-core-c3 mysql-common ruby ruby1.8
```

Next, install OpenNebula with:

```
$ sudo dpkg -i opennebula_2.0-beta1-1_amd64.deb
```

OpenNebula is now installed into /srv/cloud/one/. Set the environment variables that OpenNebula uses, using for example ~/.bashrc:

```
export ONE_LOCATION=/srv/cloud/one  
export ONE_XMLRPC=http://localhost:2633/RPC2  
export ONE_AUTH=$ONE_LOCATION/.one_auth  
export PATH=$ONE_LOCATION/bin:$PATH
```

then create the file \$ONE_AUTH:

```
$echo "username:password" > $ONE_AUTH  
$chown oneadmin:oneadmin $ONE_AUTH
```


Now the OpenNebula daemon can be started with:

```
$ sudo -u oneadmin one start
```

Cluster nodes: In addition to OpenSSH, ruby needs to be installed on the nodes:

```
$ sudo apt-get install ruby
```

B. OpenNebula Configuration

Head-node

The configuration file for OpenNebula is found in `$ONE_LOCATION/etc/oned.conf`. By default, OpenNebula is configured for KVM and shared filesystem for VM images using NFS. Next, we configure NFS sharing on the OpenNebula folder

```
sudo apt-get install nfs-kernel-server
```

and configure the shared folder by adding to `/etc/exports`:

```
/srv/cloud 192.168.2.0/255.255.255.0(rw,sync,no_subtree_check)
```

and then restart the NFS server:

```
$ sudo service nfs-kernel-server restart
```

Lastly, create a folder for shared images:

```
$ sudo -u oneadmin mkdir /srv/cloud/images
```

Cluster nodes On the cluster nodes we mount the OpenNebula shared folder from the head-node, by adding to `/etc/fstab`:

```
192.168.2.1:/srv/cloud /srv/cloud nfs rw,hard,intr 0 0
```

and then:

```
$ sudo apt-get install nfs-client  
$ sudo -u oneadmin mkdir /srv/cloud  
$ sudo mount /srv/cloud
```

EDIT: On the cluster nodes, configure the bridge network interface by modifying `/etc/network/interfaces` to look similar to:

```
# The primary network interface  
auto eth0  
iface eth0 inet manual  
  
auto br0  
iface br0 inet static  
    address 192.168.2.2  
    netmask 255.255.255.0  
    network 192.168.2.0  
    broadcast 192.168.2.255  
    gateway 192.168.2.1  
  
# dns-* options are implemented by the resolvconf package, if installed  
dns-nameservers x.x.x.x  
  
bridge_ports eth0  
  
bridge_fd 9  
  
bridge_hello 2
```

```
bridge_maxage 12
```

```
bridge_stp off
```



Appendix B

PHP Scrip to Control VMs and Hosts

A. host.php

(1) list host

list_host.sh → List all of hosts.

```
#!/bin/sh
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
#
sudo -s
source /root/.bashrc
onehost list
show_host.sh → List detail host info 【Parameter :
host_name(debian1~debian4)】
```

(2) add host

add_host.sh → Add a host 【Parameter: host_name(debian1~debian4)】

```
#!/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
```

(3) delete host

delete_host.sh → Delete a host **【Parameter: host_name(debian1~debian4)】**

```
#!/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

/var/www/script/rrd_graph/src/delete_cpu_mem.sh $1
```

(4) start opennebula

one_start.sh → Start OpenNebula

```
#!/bin/sh

PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

#
```

```
sudo -s

source /root/.bashrc

#

one star
```

(5) stop opennebula

one_stop.sh → Stop OpenNebula

```
#!/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
```

B. vms.php

(1) create vm

create_vm.sh → Build VM 【Parameter: host_name(ct) ip(140.128.102.187)

size(1g) rootpw(abc123)】

- convert.sh
- create_default_vm.sh

- create_default_vm.sh.ok
- install_default_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
```

(2) list vm

list_all_vm.sh → List all of VMs.

```
#!/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->列出目前開機 VM  
#!/bin/sh  
PATH=/data/one/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
  
#load env  
  
sudo -s  
  
source /root/.bashrc  
  
#main code  
  
onevm list
```


(3) boot vm

boot_vm.sh → Start VM

```
#!/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
```

(4) delete vm

delete_vm.sh → Delete VM

```
#!/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
```

(5) shutdown vm

shutdown_one_vm.sh → Shutdown VM

```
#!/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
```

(6) migrate vm

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
```