東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

一個節電方法於雲端虛擬機管理平台之實作
Implementation of a Power Saving Method for Virtual Machine Management Platform in Cloud

研究生: 黃冠龍

中華民國一零二年七月

摘　要

雲端技術的應用和服務與日俱增，無論是政府、企業或是個人都可能有建構雲端系統的需求。隨著雲端運算的盛行，雲端基礎設施的需求也跟著提升。雲端運算環境所帶來的電力花費是相當可觀的，為了提供大量的使用者存取雲端服務，企業經常需要運用大量的雲端伺服器，但服務並非隨時都被大量的存取，在低使用率的期間，若同樣保持大量的雲端伺服器開機，就會造成不必要的浪費。因此本研究的目的是在雲端三種服務模式的基礎建設服務，利用開放原始碼的技術，整合 KVM 和 Libvirt 等開放源始碼軟體，在雲端建立一個虛擬機管理平台，經由 SNMP 偵測雲端伺服器以及虛擬機器運作情形並計算整體使用效率，透過 Live migration 技術分配虛擬機器，並關閉多餘的雲端伺服器，達到節能的效果。透過這個節能方法建置一套具節能效果之虛擬機管理平台，目標為提供企業或個人一個具有節能效果的私有雲解決方案。我們也建置一個讓使用者容易操作的使用者介面，透過此網頁介面來管理、遷移虛擬機以及監控系統的資源使用情況。在實驗環境中以資源使用率監控技術、虛擬機動態遷移與控制本系統之節電方法。在實驗中透過 PDU 記錄用電情形，證明我們能有效的利用硬體資源並根據數據分析結果，達成節能之效果。

關鍵字: 節電，虛擬機管理，動態遷移，虛擬化，服務品質保證

Abstract

With the popularity of cloud computing, the demand of cloud infrastructure also increases; as a result, considerably large amount of electricity is consumed on the cloud computing environment. In order to provide a lot of users with access to cloud services, companies often need to use many cloud servers; but since most services are not accessed in great amount all the time, in the low service usage period, it will cause unnecessary waste of electricity and computing resources if lots of cloud servers maintain at full power. This work uses the open source codes and PHP web programs to implement a virtualization resource management system for power-saving. In this thesis, we propose to adopt system integrated open source software like KVM and libvirt to construct a virtual cloud management platform, which detects the status of resources via SNMP, calculates the operation efficiency of the overall system, allocates virtual machines through the live migration technology and turns off extra machines on the cloud to save energy. According to the power-saving method developed by our research, we have constructed a power efficient virtualization management platform in the cloud. Our objective is to provide enterprises or end users with power-saving private cloud solutions. In this work we also have built a webpage to allow users to easily access the cloud virtualization resources, i.e., users can manage virtual machines and monitor the status of resources via the web interface. From analysis of the experimental results of live migration of virtual machines, this work demonstrates that efficient use of hardware resources is realized by the power-saving method, and the aim of power-saving is achieved.

Keywords: Power Saving, VM Management, Live migration, Virtualization, SLA

致謝詞

能夠順利完成這篇論文，我要感謝我的指導教授楊朝棟老師，謝謝老師六年來的教導，尤其是研究所這兩年的帶領，讓我在一件件的工作中學習與成長，沒有這些磨練，我沒有辦法順利完成這篇論文與學業。也謝謝老師這兩年給我這麼多機會到國外去挑戰與接觸新的事務，希望在最後的成果上，沒有讓老師失望。

感謝實驗室的同學，育佐、文鴻、啟瑞、小歐，謝謝你們兩年來的陪伴，與諸多方面的幫忙，無論是架系統、寫程式、做 Power point 還是打 LOL，沒有你們，我一定無法順利畢業。雖然我重色輕友常常不跟你們一起行動，又常常因為總管的職務拿一堆雜事煩你們，但我一直都是最愛你們的。拼論文的這兩個月，大家為了做實驗、畫圖、研究 Latex，每天躲蟑螂、定速時、看日出、吃炒麵的生活，一輩子都不會忘記。

感謝我的家人，在我表現不如你們理想的時候，依然堅定的支持我相信我，最後終於完成學業，希望這篇論文能讓你們感到驕傲。感謝顯意、Adam、允文、很辣還有其他學長的帶領，讓我能快速融入實驗室的生活，你們是我最好的榜樣。感謝口試委員呂芳懌老師以及陳瑞男老師這幾年的教導還有其他老師的提點，感謝學弟妹們的幫忙。

最後特別感謝女朋友 Mia，謝謝妳這一年多來的陪伴，每天聽我抱怨、訴苦。謝謝你的安慰、鼓勵與支持，讓我能一直堅持到今天。

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud computing, one of today's hottest topics, has been developing very fast. Many large cloud companies, such as Google, Amazon, Yahoo, offer multiple cloud services and have a large number of users. In order to provide these services, the cloud infrastructure grew nearly 27 percent in 2012, and is expected to grow nearly 47.3 percent in 2013. A large cloud infrastructure usually includes cloud servers and cloud storage. The global cloud infrastructure is estimated to consume about 30 million kilowatts of electricity in one hour, but 90 percent of it is wasted due to inefficient use of cloud resources; hence, how to reduce electricity consumption waste on the cloud infrastructure is a topic worth exploring.

1.1 Motivation

Cloud computing is a concept, in which computers over networks are able to cooperate with each other to provide far-reaching network services [1,12,13,26,34, 38]. The basic approach to computing through the Internet terminal operations of cloud computing moves hardware, software, and shared information from users to the server side; in this way, the original waste of redundant resources on personal computer is evaded to improve computing and resource efficiency [3,28,35,37,41, 43,49,52].

For increasingly high demand of the cloud today, a typical application of cloud computing is the large-scale data processing center, which operates with tremendous amount of power consumption [2, 6, 11, 20, 21, 27, 32, 42, 50]. The power consumption of large data processing centers today share about 0.5% of the global carbon emissions; with the deepening of cloud computing in the future, it is predicted to account for 2%, which represents carbon emissions of large data processing centers of some large enterprises, and may even beyond 2% for some countries. Such huge amount of power consumption, especially with today's emphases on power conservation and carbon reduction, is a major problem that cannot be ignored. How to maintain the growth of cloud computing technology, but also take into account of the efficiency of power use, comprises the main research direction of this work.

The power demand of the cloud environment issues cannot be ignored [5, 8, 10, 15–17, 19, 23, 33, 36, 40, 44, 46–48]. Through live migration of the virtual machine technology, the need to interrupt and reopen servers for operations is eliminated and the purpose of power-saving is achieved. However, if one only partially focuses on the virtual machine side, the quality of services in the cloud environment is rendered to reduce, contrary to the intent of the cloud computing. In the commercial cloud computing environment, emphasis is on service level agreements. An effective management tools can help the service provider and the client with services and service level expectations of each other; in other words, it can help enterprises to construct channels of communication with communication plans to establish consistency and reduce conflicts, and methods to measure service performance [9, 14, 22, 24, 25, 29, 30, 39, 51].

1.2 Thesis Goal and Contributions

This article will take into considerations the service level agreements, primarily through measurement of the CPU and memory usage. With quality of service as a precondition, this work designed and implemented an energy efficient cloud virtual machine management system using power-saving method, and live migration of virtual machines.

Finally, the experimental results show the relationship between VM's vCPU and migration time, the relationship between Server's CPU and migration, and live migration due to the extra cost of electricity. The experimental results show that the proposed power-saving method under normal usage scenarios can successfully detect cloud environment resource usage, turn off unnecessary servers to reduce power waste in the case of low resource use, and maintain a certain degree of cloud service quality for rapid deployment of resources use; indeed, it achieves a certain degree of energy saving effect.

1.3 Thesis Organization

Section 2 will describe some background information, including cloud computing, virtualization, live migration, and hardware information measurement method. Section 3 will introduce our experimental environment and methods, and the overall architecture. Section 4 presents and analyses experimental results. Finally, Section 5 summarizes this thesis by pointing out its major contributions and directions for future work.

Chapter 2

Background Review and Related Work

2.1 Cloud Computing

Cloud computing is Internet-based computing, in which shared hardware, software resources, and data can be provided to users according to their demand of computers and other devices. The user does not need to know the details of the cloud infrastructure, or possess appropriate expertise, and is without direct control. Cloud computing allows companies to deploy applications more quickly and reduce the complexity of management and maintenance costs, and allows rapid changes of IT resources reallocation in response to business needs. Cloud computing describes a new Internet-based services to increase IT use and delivery models, usually involving with the Internet, and is easy to provide dynamic and virtual extension of the resource. The cloud is network, the Internet a metaphor. Cloud computing can be considered to include the following levels of service: infrastructure as a service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS):

- SaaS: Consumers use the application, but do not control the operating system, hardware, or the operation of the network infrastructure, for example:

Microsoft CRM and Salesforce.com platform as a service. With the SaaS model, the user can access the services, software, and data. The service providers maintain the normal operation of the infrastructure and platform to maintain service, while users access cloud services through browsers, and desktop or mobile applications.

- PaaS: Consumers use the host operating applications. Consumers control the operation of the application environment (host also has some control over it), but do not control the operating system, hardware, or the operation of the network infrastructure. The platform is usually the application infrastructure, for example: Google App Engine.
- IaaS: Consumers can use the basic computing resources such as processing power, storage space, network components or middle-ware. Consumers can control the operating system, storage, deployed applications and network components (such as firewalls, load balancers, etc.), but do not control the cloud infrastructure. Examples of IaaS include Amazon AWS, and Rackspace.

Cloud computing relies on the sharing of resources in order to achieve economies of scale in similar infrastructure. Service providers integrate a large number of resources for use by multiple users. Users can easily request more resources and adjust usage at any time, without the need to release all resources back to the whole structure. Users do not need to buy a lot of computing resources; they only need to enhance the amount of rent for short-term spikes of resource demands, and release some resources during lower demands. Service providers are able to lease unused resources to other users, and even adjust rents in accordance with the demand of the whole. The National Institute of Standards and Technology Research Institute also defines cloud computing based on the cloud computing deployment model:

- Public cloud: In short, public cloud provides services through the Internet and third party service provider, and is opening up to the customer to use.

The term public does not necessarily mean free, but may also represent very cheap. In the public cloud, the user data is available for anyone to view. Public cloud providers usually suggest interested users to implement some access control mechanisms. The public cloud is a flexible and cost-effective solution.

- Private cloud: Private cloud have many advantages, such as with more flexibility than a public cloud environment and being suitable for the provision of services. The difference between the two is that for private cloud services, data and programs are managed within the organization, while for public cloud services, they are constrained by the network bandwidth, security concerns, and regulatory restrictions. In addition, private cloud services suppliers and users have more control over the cloud infrastructure to improve safety and flexibility, because the user and the Internet are subject to special restrictions.
- Community Cloud: Community cloud is controlled and used by members of organizations with common interests, specific security requirements, and common purpose. Community members can access cloud data and applications.
- Hybrid cloud: The hybrid cloud is a mix of the public cloud and private cloud. In this mode, the user usually has non-business-critical information dealt with in a public cloud, but at the same time, control of business-critical services and information.

2.2 Virtualization

With virtualization, the computer's physical resources, such as servers, network, memory, and storage, are abstractly presented after conversion, so that users can apply those resources in a better way than the original configuration. Virtualization is commonly referred to virtualized resources including computing power and data storage; in this paper virtualization is specifically referred to server virtualization. Server virtualization software technology refers to the use of one or more of the host hardware settings. It has the flexibility to configure the virtual hardware platform and operating system, like real hardware. In this way, a variety of different operating environments (for example, Windows, Linux, etc.) can operate simultaneously on the same physical host, and be independent as being operating in different physical hosts. Virtualization solutions can be broadly divided into three categories: full virtualization, para-virtualization, and hardware-assisted virtualization.

2.2.1 Full Virtualization

In Full virtualization, Hypervisor replaces the traditional core of the operating system in the Ring 0 privilege level, known as Binary Translation technology used in full virtualization. The guest operating system requires direct access to the hardware Ring 0 level instruction by the hypervisor conversion to submit requests further to hardware.

Hypervisor virtualizes all hardware components. The guest operating system works as the individual real host, with a high degree of independence. But Binary Translation technology will make the performance of the virtual machine (Virtual Machine VM) decreased significantly. One example of software using virtualization technology is the VMware Workstation.

The full virtualization is totally dependent on the virtual hardware layer constructed Guest OS, which can use hardware resources, only limited by the virtual hardware resources, and less able to misallocate of the entity's computer hardware. In addition to the central processor, memory, graphics card, etc. can be virtualized if the proprietary driver is installed first.

The benefits of full virtualization are, regardless of how the hardware environment, the Guest OS are able to maintain a more consistent compatibility, and can use different operating systems with the physical machine. The disadvantage of full virtualization is that it is a greater burden of the physical machine.

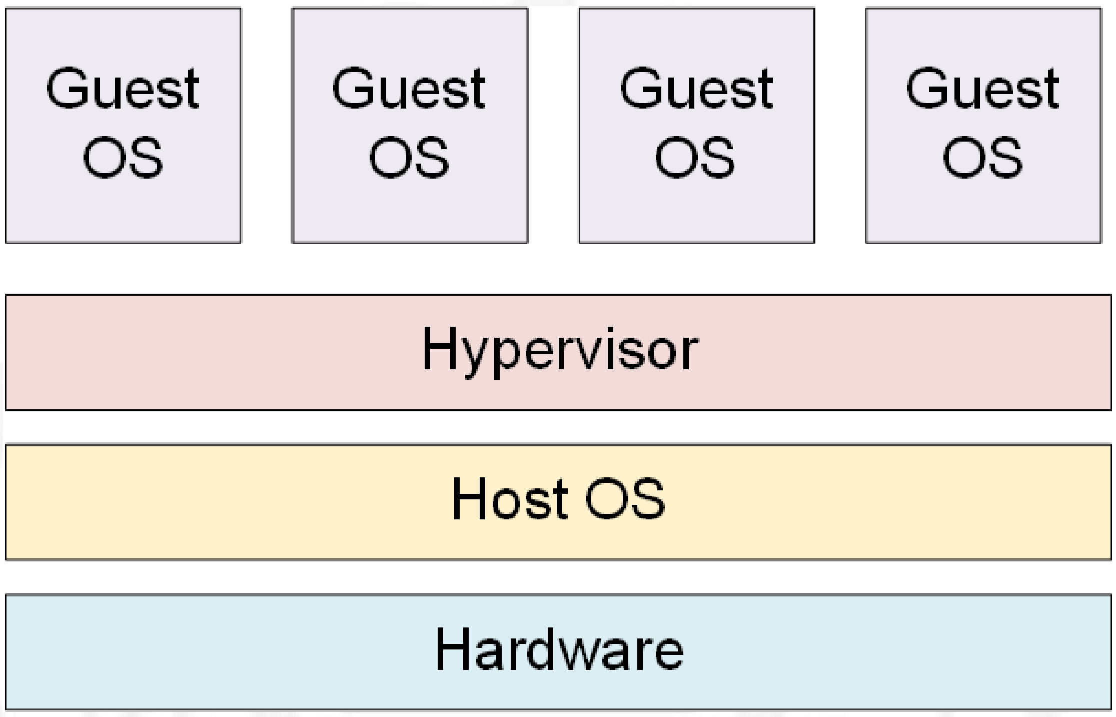


Figure 2.1: Full Virtualization

2.2.2 Para-virtualization

Para-virtualization, also known as parallel virtualization, is not all hardware device virtualization, such as Xen. Compared to the full virtualization virtual machine instruction Binary Translation, para-virtualized virtual machine invokes hardware devices through Dom0, and manages each virtual machine access to physical resources. The virtual machine performance is significantly improved, but the hardware driver binding in dom0, and the core of the operating system in the virtual machine must go through a special modification; thus the independence of the virtual machine is relatively low.

The para-virtualization does not configure the virtual layer on top of the hardware, but use more than one memory location program calls at different times. Para-virtualization allows the Guest OS share hardware resources. The advantage is that the hardware will not need to waste performance on the hardware layer;

however, the drawback is that the operating system in the virtual machine must be consistent with the physical environment.

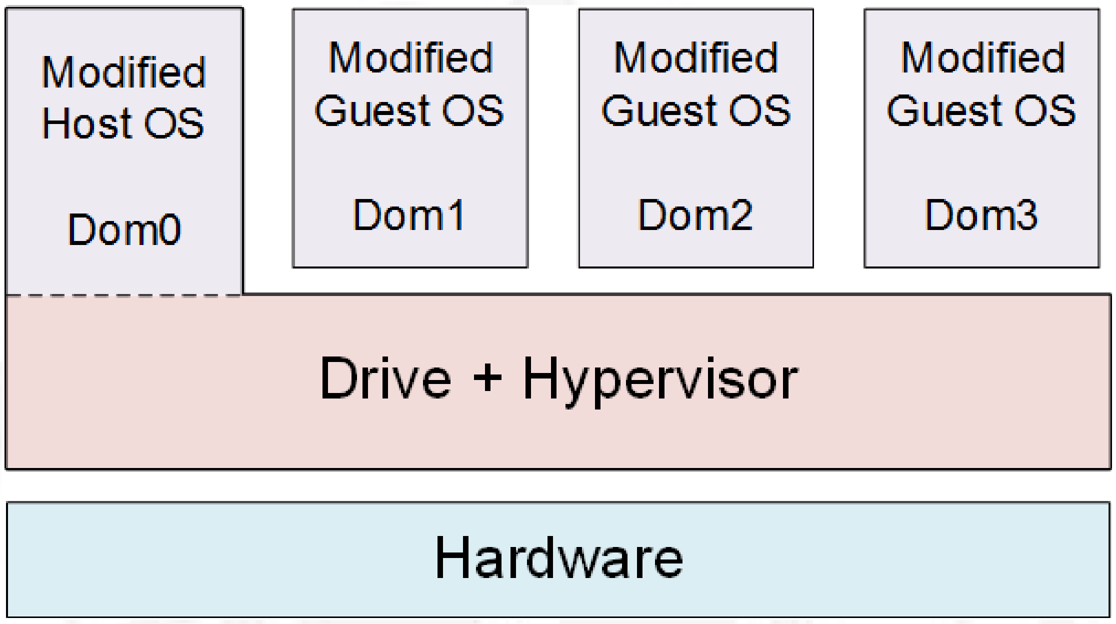


FIGURE 2.2: Para-virtualization

2.2.3 Hardware-assisted Virtualization

Hardware-assisted virtualization, as shown in Figure 2.3, is used to overcome the dilemma that the virtual machine operating system kernel cannot be placed in the processor Ring 0 privilege level. Since both the hypervisor and virtual operating system kernel can be issued in Ring 0, the hypervisor will be automatically intercepted by the need to deal with the instruction of the virtual operating system directly by the hardware processing; therefore the full virtualization binary translation or para-virtualization Hypercall operations are no longer in need. Hardware assisted virtualization basically eliminates the difference between full virtualization and para-virtualization. On the basis of hardware-assisted full virtualization or para-virtualization program virtual machines are high performance and independent. The representative program of the hardware-assisted virtualization VMware is ESXi with open source Kernel-based Virtual Machine (KVM). KVM needs a host operating system, whose core provides virtualization services;

whereas VMware ESXi is the equivalent of a specialized virtualization thin client operating system, installed directly on a physical machine.

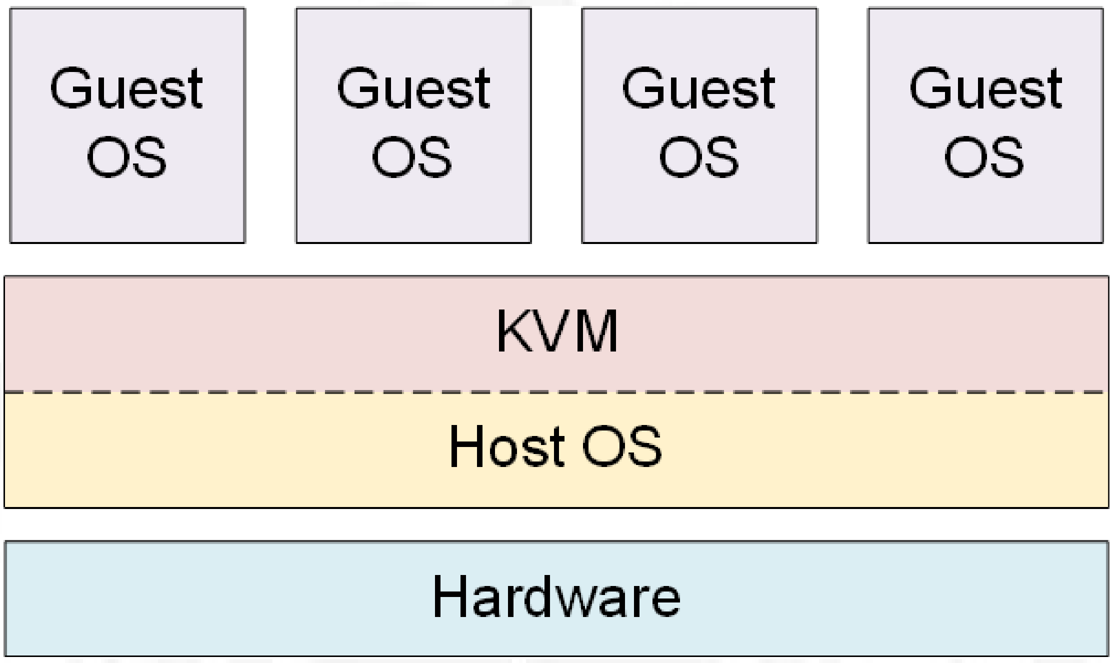


FIGURE 2.3: Hardware-assisted Virtualization

2.3 Hypervisor

The one that builds and manages virtualized environment software is collectively referred to as the hypervisor (or Virtual Machine Monitor, VMM). Depending on the virtualization solution, there are different choices for hypervisor:

2.3.1 KVM

Kernel-based Virtual Machine (KVM), as illustrated in Figure 2.4, is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, which provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU, although work is underway to get the required changes upstream. Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.

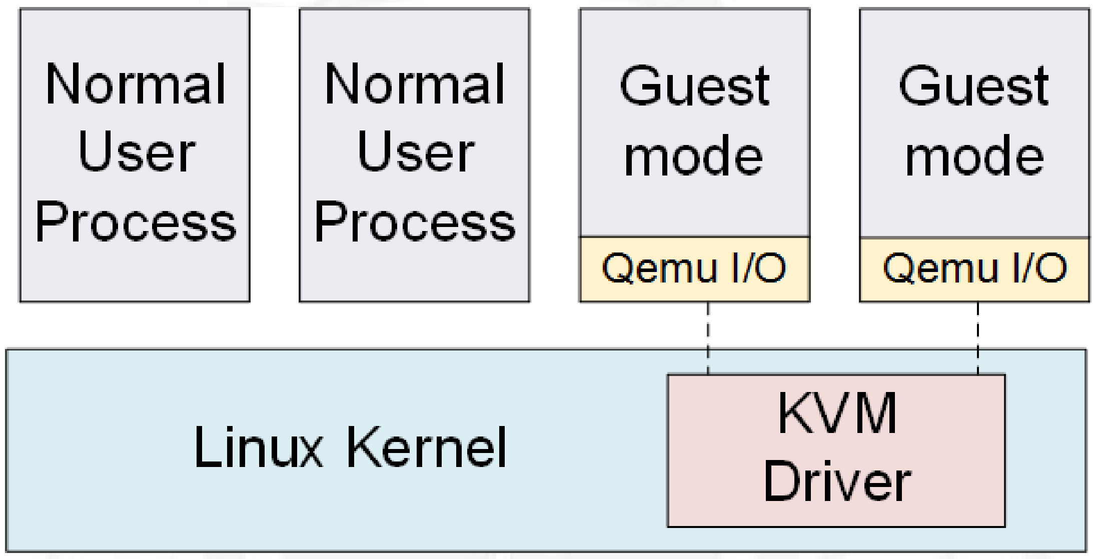


FIGURE 2.4: Kernel-based Virtual Machine

2.3.2 Xen

Xen is a native or bare-metal hypervisor. It runs in a more privileged CPU state than any other software on the machine. Responsibilities of the hypervisor include memory management and CPU scheduling of all virtual machines ("domains"), and for launching the most privileged domain ("dom0") - the only virtual machine which by default has direct access to hardware. From the dom0 the hypervisor can be managed and unprivileged domains ("domU") can be launched.

The dom0 domain is typically a modified version of Linux, NetBSD or Solaris. User domains may either be unmodified open-source or proprietary operating systems, such as Microsoft Windows (if the host processor supports x86 virtualization, e.g., Intel VT-x and AMD-V),[2] or modified, para-virtualized operating system with special drivers that support enhanced Xen features. On x86 Xen with a Linux dom0 runs on Pentium Pro or newer processors. Xen boots from a boot loader such as GNU GRUB, and then usually loads a para-virtualized host operating system into the host domain (dom0).

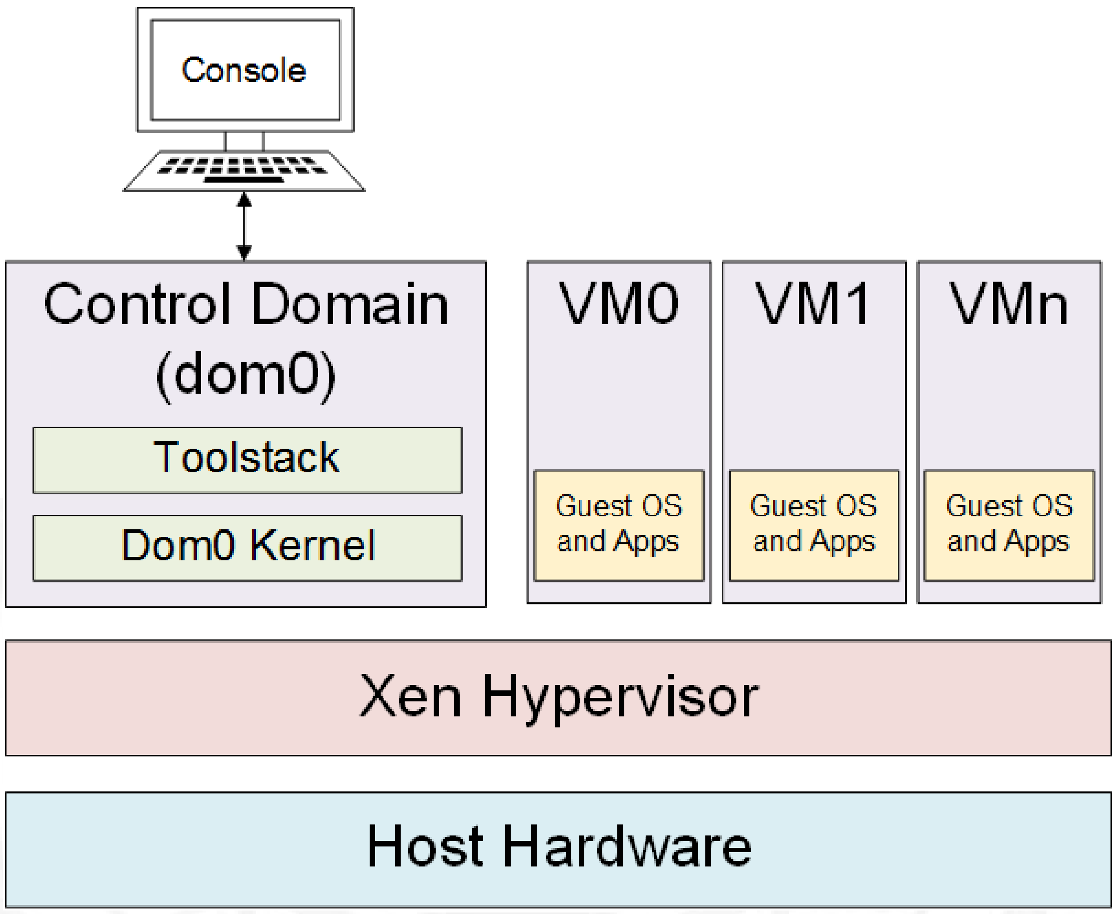


FIGURE 2.5: Xen

2.4 Virtual Machine Management

To set up a standard cloud service, we often need more than one virtual machine. When a large number of virtual machines created through the virtualization technology, it is a difficult task to manage by native instructions, and a virtual machine management platform is needed. In Figure 2.6, the virtual management platform usually includes a virtual machine to create, edit, switch, pause, reply, delete, and do live migration. Next, we will introduce several sets of popular open source virtualization management platforms. We also provide a friendly virtual machine building process through a network interface, more suitable for monitoring a large number of virtual machine states, and easier to manage account permissions and other advantageous features.

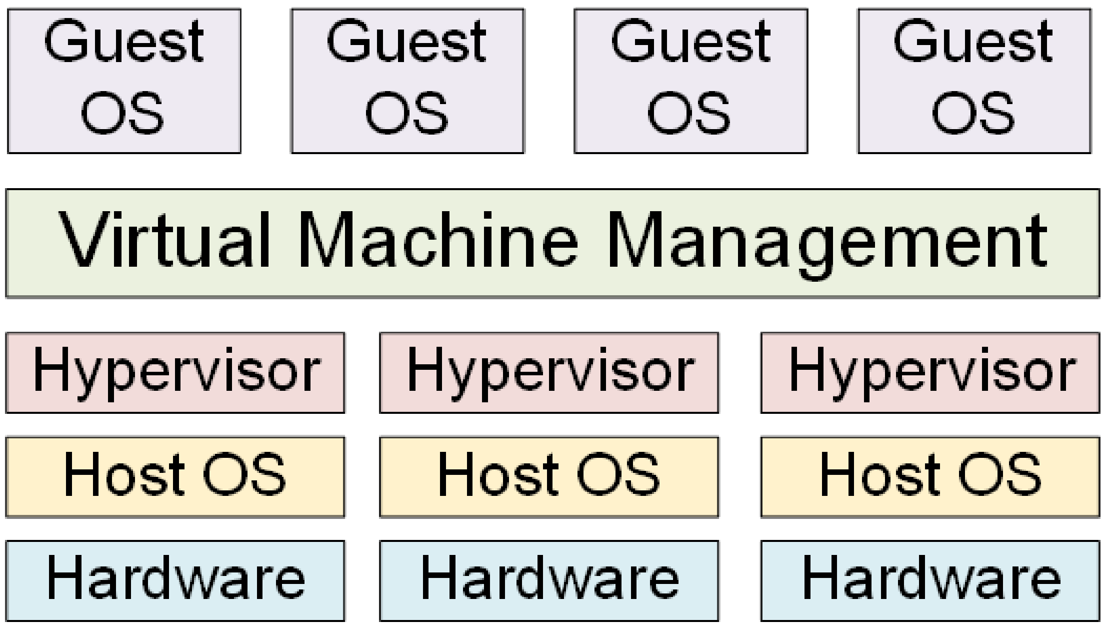


FIGURE 2.6: Virtual Machine Management

2.4.1 Live Migration

Live migration, as shown in Figure 2.7, refers to the process of moving a running virtual machine or application between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the virtual machine are transferred from the original host machine to the destination. Two techniques for moving the virtual machine's memory state from the source to the destination are pre-copy memory migration and post-copy memory migration.

- Warm-up phase: In pre-copy memory migration, the hypervisor typically copies all the memory pages from source to destination while the VM is still running on the source. If some memory pages change (become 'dirty') during this process, they will be re-copied until the rate of re-copied pages is not less than page dirtying rate.
- Stop-and-copy phase: After the warm-up phase, the VM will be stopped on the original host, the remaining dirty pages will be copied to the destination,

and the VM will be resumed on the destination host. The time between stopping the VM on the original host and resuming it on destination is called ”down-time”, and ranges from a few milliseconds to seconds according to the size of memory and applications running on the VM. There are some techniques to reduce live migration down-time, such as using probability density function of memory change.

- Post-copy memory migration: Post-copy VM migration is initiated by suspending the VM at the source. With the VM suspended, a minimal subset of the execution state of the VM (CPU registers and nonpageable memory) is transferred to the target. The VM is then resumed at the target, even though most of the memory states of the VM still reside at the source. At the target, when the VM tries to access pages that have not yet been transferred, it generates page faults.

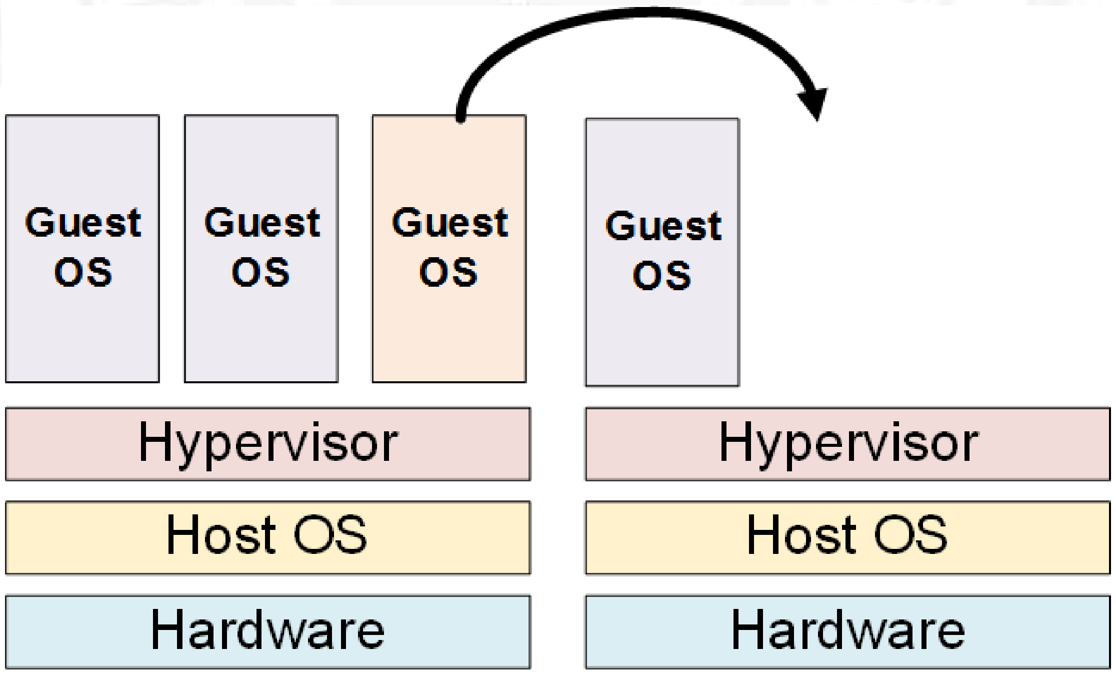


FIGURE 2.7: Live Migration

These faults are trapped at the target and redirected towards the source over the network. Such faults are referred to as network faults. The source host responds to the network-fault by sending the faulted page. Since each page fault of the running VM is redirected towards the source, this technique can degrade performance of applications running inside the VM. However, pure demand-paging accompanied with techniques such as pre-paging can reduce this impact by a great extent.

2.5 SNMP

Simple Network Management Protocol (SNMP) is an ”Internet-standard protocol for managing devices on IP networks”. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks, and more. It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention. SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects.

SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications. In typical SNMP uses, one or more administrative computers, called managers, have the task of monitoring or managing a group of hosts or devices on a computer network. Each managed system executes, at all times, a software component called an agent which reports information via SNMP to the manager.

Essentially, SNMP agents expose management data on the managed systems as variables. The protocol also permits active management tasks, such as modifying and applying a new configuration through remote modification of these variables. The variables accessible via SNMP are organized in hierarchies. These hierarchies, and other metadata (such as type and description of the variable), are described by Management Information Bases (MIBs).

2.6 PDU

A power distribution unit (PDU), as shown in Figure 2.8, is a device with multiple outlets designed to distribute the electric power, especially to racks of computers and networking equipment located within the data center.

The term, PDU, may refer to two major classes of hardware power devices. The first and typically the general unqualified term refers to the category of relatively higher-cost floor-mounted power distribution devices which transform one or more larger capacity raw power feeds into any number of lower capacity distributed power feeds. These floor-mounted PDU devices are typically composed of transformers and circuit breakers and may optionally include monitoring controllers using protocols such as Modbus or SNMP. In a typical data center for example, there would be relatively few of these floor-mounted PDU devices, located along the walls or in central locations for larger spaces. Each floor-mounted PDU would feed a much larger number of racks and rows of racks.

The second class of device is a much smaller and lower cost device with multiple appliance outlets designed to distribute the electric power within a rack, especially to computers and networking equipment located within a data center. The second type of PDU is sometimes called a Smart-PDU, Rack-based PDU, Intelligent PDU or simply ”Power Strip” by various IT professionals.

FIGURE 2.8: PDU

2.7 Power-saving Related Work

In the past years, the research field of green and low power consumption networking infrastructure is of great importance for both service/network providers and equipment manufacturers. The emerging cloud computing technology can be used to increase the utilization and efficiency of hardware equipment, thus, potentially reducing the global CO2 emission. Chang et al. proposed virtual network architecture for cloud computing [7]. Their virtual network can provide communication functions for virtual resources in cloud computing. They designed an energy aware routing algorithm for virtual routers and an efficient method for setting up the virtual network to fulfill the objective of building a green virtual network in cloud computing.

Baliga, J. et al. presented an analysis of energy consumption in cloud computing [4]. The analysis considers both public and private clouds, and includes energy consumption in switching and transmission as well as data processing and data storage. They show that energy consumption in transport and switching can be a significant percentage of total energy consumption in cloud computing.

Cloud computing can enable more energy-efficient use of computing power, especially when the computing tasks are of low intensity or infrequent. However, under some circumstances cloud computing can consume more energy than conventional computing where each user performs all computing on his personal computer. These two papers also aim for power-saving. They proposed a method with good performance on network. The formula can be used to calculate the method's effect on power-saving. This work also proposed a similar formula to calculate the power-saving results mathematically.

Kim et al. suggested a model for estimating the energy consumption of each virtual machine without dedicated measurement hardware [18]. Their model estimated the energy consumption of a virtual machine based on in-processor events generated by the virtual machine. Based on this estimation model, they also proposed a virtual machine scheduling algorithm that can provide computing resources according to the energy budget of each virtual machine. The suggested schemes were implemented in the Xen virtualization system, and an evaluation showed that the suggested schemes estimated and provided energy consumption with errors of less than 5% of the total energy consumption. Power-saving method of this paper is quite similar to that in the thesis. It calculates the resources required for the virtual machine and then schedules the virtual machine to allocate operational resources. The paper-saving principle in the experiments has confirmed again in this work. The power consumption of servers and server CPU Loading has linear relationship. The experiment proved that it could have to 5% power-saving effect. If combined with the proposed power-saving method of this work, it is expected to have more than 20% power-saving effect.

To better manage the power consumption of web services in cloud computing with dynamic user locations and behaviors, Zhengkai Wu et al. proposed a power budgeting design based on the logical level, using a distribution tree [45]. By setting multiple trees, they can differentiate and analyze the effect of workload types and Service Level Agreements (SLAs, e.g. the response time) in terms of power characteristics. Based on these, they introduce classified power capping for different services as the control reference to maximize power saving when there are mixed workloads. This paper also uses the scheduling approach to control the operation of the VM, and achieves power-saving effect. In particular, this paper also takes into account the characteristics of the SLA, same with that used in the thesis. To achieve power-savings by control resource utilization, it is necessary to consider the quality of service and user experience.

Consolidation of applications in cloud computing environments presents a significant opportunity for energy optimization. As a first step toward enabling energy efficient consolidation, Shekhar Srikantaiah studied the inter-relationships between energy consumption, resource utilization, and performance of consolidated workloads [31]. The study reveals the energy performance trade-offs for consolidation and shows that optimal operating points exist. They model the consolidation problem as a modified bin packing problem and illustrate it with an example. They outline the challenges in finding effective solutions to the consolidation problem.

Hai Zhong investigated the possibility to allocate the Virtual Machines (VMs) in a flexible way to permit the maximum usage of physical resources [53]. They use an Improved Genetic Algorithm (IGA) for the automated scheduling policy. The IGA uses the shortest genes and introduces the idea of Dividend Policy in Economics to select an optimal or suboptimal allocation for the VMs requests. The simulation experiments indicate that the dynamic scheduling policy performs much better than that of the Eucalyptus, Open Nebula, Nimbus IaaS cloud, etc. The tests illustrate that the speed of the IGA almost twice the traditional GA scheduling method in Grid environment and the utilization rate of resources always higher than the open-source IaaS cloud systems.

The above two papers capture resource usage in a way that is worthy for reference. This work uses SNMP to retrieve the desired information, but this method is not limited to only one realization. There are varieties of open source software to monitor the resources status on network, like vmstat, Monit, Monitorix. These software are good tools the can be tested in the experiments.

Chapter 3

System Design and Implementation

3.1 System Architecture

The management node consisted of the main monitoring function, the applied method, the judgment function of the shared storage system, the virtual machine management platform, and the user interface. We used two computing nodes running virtual machines, and connected them to the PDU to record power consumption. In order to successfully perform live migration, storages of the two computing nodes are connected to a shared storage via NFS. The system architecture is shown in Figure 3.1.

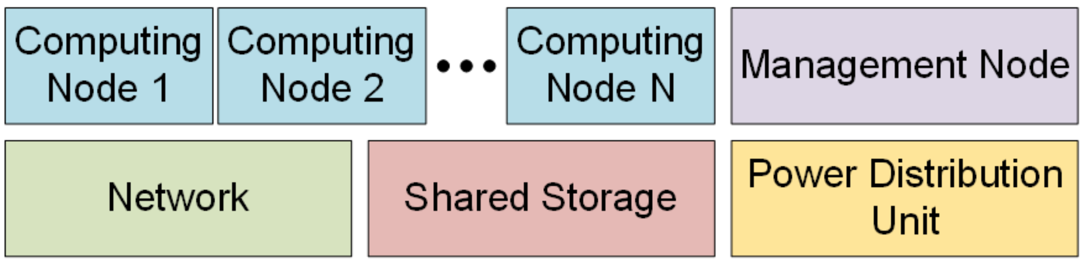


FIGURE 3.1: System Architecture

3.2 Design Flow

In the experiments we used the KVM virtualization technology to build ten virtual machines on three servers (one for the management node, the other two for computing nodes), and SNMP to monitor operations of the two computing nodes and resource usages of CPU and memory of the ten virtual machines. We gradually increased the CPU usage of virtual machines, and used the cpuburn program to simulate various degrees of usage of the cloud computing environment. The states of the machines were automatically acquired by a PHP program and SNMP.

Figure 3.2 shows the flow chart of the power-saving method. When the resources required for the virtual machines were less than those supported by the currently running servers, through the libvirt dynamic migration instructions, the virtual machines were centralized in some servers and some non-essential servers were turned off in order to achieve the goal of energy saving. And if the virtual machine CPU or memory usages increased, the number of servers in operation should be always sufficient to maintain the quality of service. And if the current CPU or memory resources of running servers were not sufficient to maintain quality of service, some standby servers will be elected to join the computing cluster, and appropriate transfer were made for the new virtual machines on the servers through the libvirt dynamic migration instruction. Based on the power-saving method the system will determine whether the current status of the resource usage is well balanced. The following will focus on the description of several key programs.

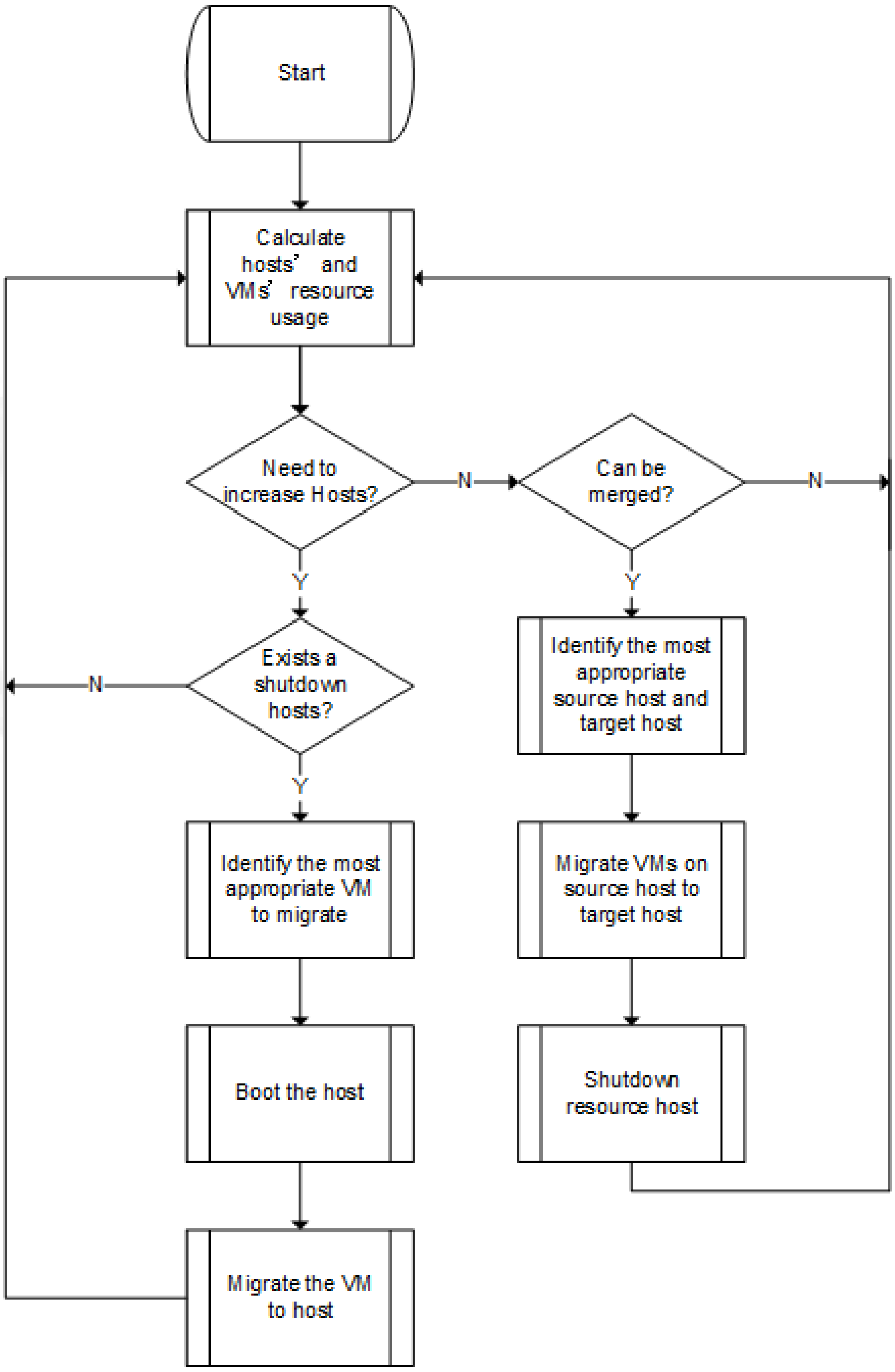

Start
Calculate hosts' and VMs' resource usage
Need to increase Hosts?
N
Can be merged?
N
Y
Exists a shutdown hosts?
N
Y
Identify the most appropriate source host and target host
Y
Identify the most appropriate VM to migrate
Migrate VMs on source host to target host
Boot the host
Shutdown resource host
Migrate the VM to host

FIGURE 3.2: Method flow chart

3.3 An Example for Power-saving Method

This section uses two examples to demonstrate the operation of the process of the power-saving method. Assuming the operating environment has three nodes with same specifications, node1, node2 and node3, all operating three virtual machines. The two cases used for demonstration are expanding and merging computing nodes.

3.3.1 Merging Computing Nodes

The example of the merging operation is shown in Table 3.1, nine VMs are maintained in low usage. The power-saving method will automatically perform live migration on all virtual machines on the server with the lowest total utilization to other servers, and close that server. In this example, node2 had the lowest total CPU usage, so VM21, VM22 and VM23 were migrated to node1 and node3, and node2 was closed as shown in Table 3.2. If the VM's vCPU utilization did not increase, node2 will remain off to save electricity.

Server name	VM name	CPU usage	Total CPU usage
Node 1	VM 11	10%	50%
	VM 12	20%	
	VM 13	20%	
Node 2	VM 21	10%	30%
	VM 22	10%	
	VM 23	10%	
Node 3	vm 31	20%	60%
	VM 32	30%	
	VM 33	10%	

Table 3.1: Example 1 without power-saving method

Server name	VM name	CPU usage	Total CPU usage
Node 1	VM 11	10%	70%
	VM 12	20%	
	VM 13	20%	
	VM 21	10%	
	VM 22	10%	
Node 2			0%
Node 3	VM 23	10%	70%
	VM 31	20%	
	VM 32	30%	
	VM 33	10%	

TABLE 3.2: Example 1 with power-saving method

3.3.2 Expanding Computing Nodes

The example of expanding computing nodes are shown in Table 3.3. If the VM usage increases and causes the server's required CPU usage be over to 100%, the power-saving method will perform live migration on a VM in the largest vCPU usage node and move it to the lowest CPU usage node to prevent degradation of service. For example, following the situation of example 1, if the vCPU usage of VM23 operating on node3 suddenly rises, resulting required CPU usage node3 over 100%, the power-saving methods will wake up node2 from the shutdown status, and perform live migration on VM23 to move it to node2, to maintain the quality of service of the cloud environment.

Server name	VM name	CPU usage	Total CPU usage
Node 1	VM 11	10%	70%
	VM 12	20%	
	VM 13	20%	
	VM 21	10%	
	VM 22	10%	
Node 2			0%
Node 3	VM 23	90%	150%
	VM 31	20%	
	VM 32	30%	
	VM 33	10%	

Table 3.3: Example 2 without power-saving method

Server name	VM name	CPU usage	Total CPU usage
Node 1	VM 11	10%	70%
	VM 12	20%	
	vm 13	20%	
	VM 21	10%	
	VM 22	10%	
Node 2	23	90%	90%
Node 3	VM 31	20%	60%
	VM 32	30%	
	VM 33	10%	

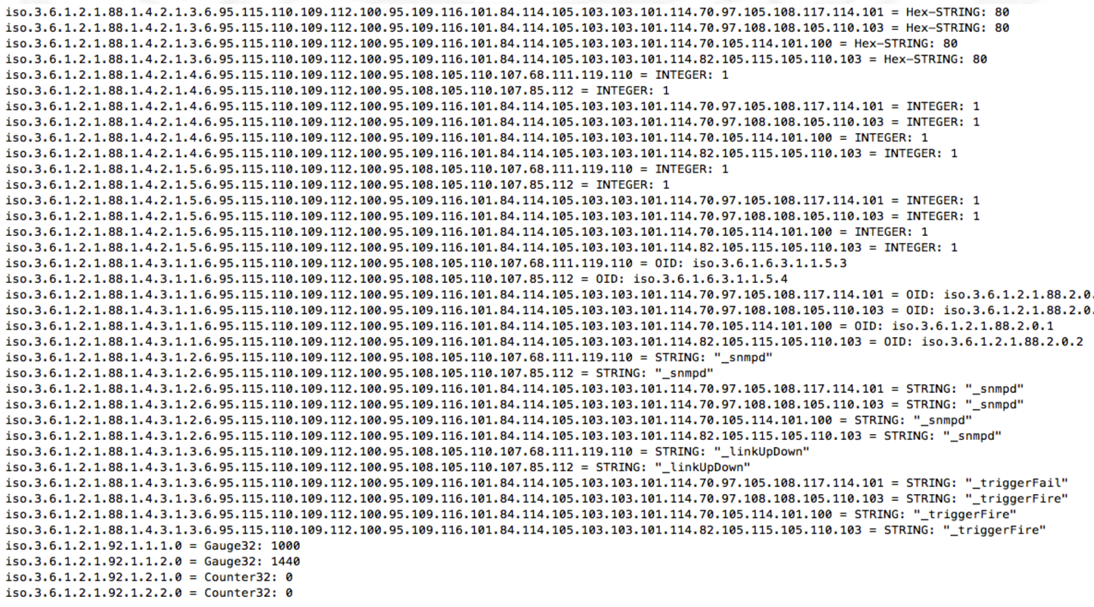
Table 3.4: Example 2 with power-saving method

3.4 System Implementation

In this thesis we uses the PHP language to write a number of automated programs, include the status monitoring program, power consumption recording program, and the power-saving method program. The following is a detailed description of the three programs.

3.4.1 Status Monitoring

To determine whether to shut down or wake up servers for the current environment, the operating states of VMs and servers are needed. Those data were retrieved via SNMP, which can access many types of information. As shown in Figure 3.3 data are obtained through SNMP commands. In order to capture information on each virtual machine, libvirt was run on all virtual machines on the server list as shown in Figure 3.4. Since libvirt supports PHP API, one can write PHP codes on it very fast.



```
iso.3.6.1.2.1.88.1.4.2.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.105.108.117.114.101 = Hex-STRING: 80
iso.3.6.1.2.1.88.1.4.2.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.108.108.105.110.103 = Hex-STRING: 80
iso.3.6.1.2.1.88.1.4.2.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.105.114.101.100 = Hex-STRING: 80
iso.3.6.1.2.1.88.1.4.2.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = Hex-STRING: 80
iso.3.6.1.2.1.88.1.4.2.1.4.6.95.115.110.109.112.100.95.108.105.110.107.68.111.119.110 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.4.6.95.115.110.109.112.100.95.108.105.110.107.85.112 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.4.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.105.108.117.114.101 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.4.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.108.108.105.110.103 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.4.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.105.114.101.100 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.4.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.108.105.110.107.68.111.119.110 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.108.105.110.107.85.112 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.105.108.117.114.101 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.108.108.105.110.103 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.105.114.101.100 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.2.1.5.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = INTEGER: 1
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.108.105.110.107.68.111.119.110 = OID: iso.3.6.1.6.3.1.1.5.3
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.108.105.110.107.85.112 = OID: iso.3.6.1.6.3.1.1.5.4
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.105.108.117.114.101 = OID: iso.3.6.1.2.1.88.2.0.4
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.108.108.105.110.103 = OID: iso.3.6.1.2.1.88.2.0.3
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.105.114.101.100 = OID: iso.3.6.1.2.1.88.2.0.1
iso.3.6.1.2.1.88.1.4.3.1.1.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = OID: iso.3.6.1.2.1.88.2.0.2
iso.3.6.1.2.1.88.1.4.3.1.2.6.95.115.110.109.112.100.95.108.105.110.107.68.111.119.110 = STRING: "_snmpd"
iso.3.6.1.2.1.88.1.4.3.1.2.6.95.115.110.109.112.100.95.108.105.110.107.85.112 = STRING: "_snmpd"
iso.3.6.1.2.1.88.1.4.3.1.2.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.105.108.117.114.101 = STRING: "_snmpd"
iso.3.6.1.2.1.88.1.4.3.1.2.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.108.108.105.110.103 = STRING: "_snmpd"
iso.3.6.1.2.1.88.1.4.3.1.2.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.105.114.101.100 = STRING: "_snmpd"
iso.3.6.1.2.1.88.1.4.3.1.2.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = STRING: "_snmpd"
iso.3.6.1.2.1.88.1.4.3.1.3.6.95.115.110.109.112.100.95.108.105.110.107.68.111.119.110 = STRING: "_linkUpDown"
iso.3.6.1.2.1.88.1.4.3.1.3.6.95.115.110.109.112.100.95.108.105.110.107.85.112 = STRING: "_linkUpDown"
iso.3.6.1.2.1.88.1.4.3.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.105.108.117.114.101 = STRING: "_triggerFail"
iso.3.6.1.2.1.88.1.4.3.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.97.108.108.105.110.103 = STRING: "_triggerFire"
iso.3.6.1.2.1.88.1.4.3.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.70.105.114.101.100 = STRING: "_triggerFire"
iso.3.6.1.2.1.88.1.4.3.1.3.6.95.115.110.109.112.100.95.109.116.101.84.114.105.103.103.101.114.82.105.115.105.110.103 = STRING: "_triggerFire"
iso.3.6.1.2.1.92.1.1.1.0 = Gauge32: 1000
iso.3.6.1.2.1.92.1.1.2.0 = Gauge32: 1440
iso.3.6.1.2.1.92.1.2.1.0 = Counter32: 0
iso.3.6.1.2.1.92.1.2.2.0 = Counter32: 0
```

Figure 3.3: Capture resource's status by SNMP

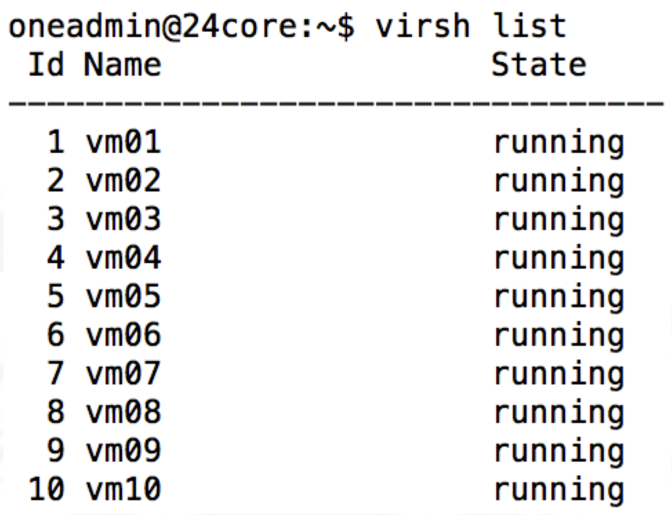

```
oneadmin@24core:~$ virsh list
 Id Name                 State
----------------------------------
  1 vm01                 running
  2 vm02                 running
  3 vm03                 running
  4 vm04                 running
  5 vm05                 running
  6 vm06                 running
  7 vm07                 running
  8 vm08                 running
  9 vm09                 running
 10 vm10                 running
```

FIGURE 3.4: Display VM list on servers by libvirt

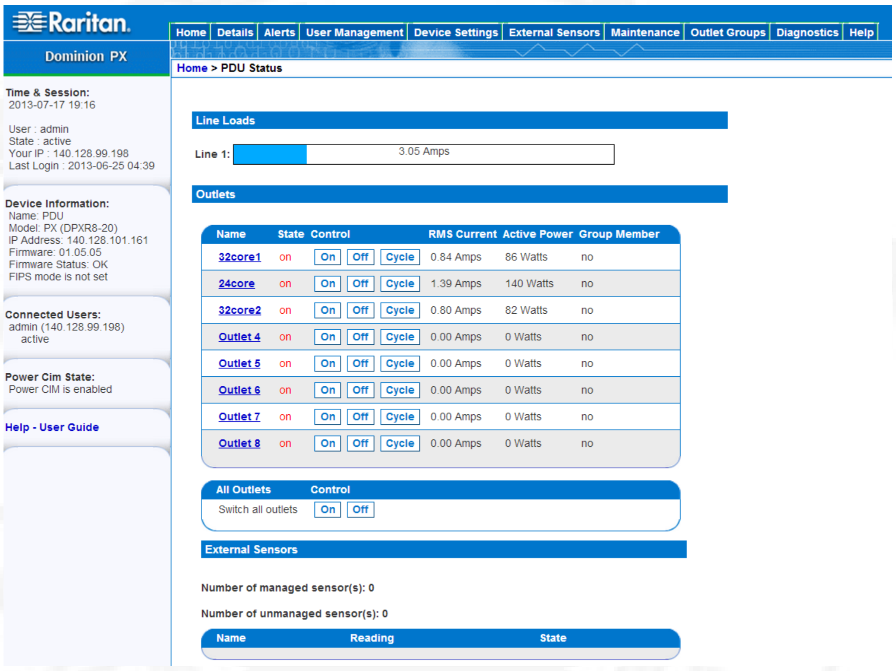
The status monitoring function was developed by the PHP programming language to obtain status data via SNMP, including CPU and memory usages of the physical and virtual machines. SNMP programs were installed to capture and send data. After installing SNMP, the snmpd.conf file in the directory /etc/snmp/ is modified to send the node's SNMP to the management node so that the application running on the management node can acquire the operating status of the two computing nodes with virtual machines.

Algorithm 3.4.1: STATUS MONITORING(c)

```
for $i \leftarrow 1$ to computing node numbers $x$
    do
        connect to computing node $x$
        get libvirt list and calculate $VM$ number
        for $o \leftarrow 1$ to $VM$ numbers $y$
            do
                get $VM$'s $CPU$ usage via $SNMP$
                get $VM$'s $RAM$ usage via $SNMP$
```

3.4.2 Power Consumption Recording

In this work, we retrieved servers' power consumption data via PDU. Figure 3.5 is the web interface provided by the PDU used in this work. Figure 3.6 shows the type of data retrieved, including rms current, power factor, and voltage for the ac power supply. The PDU data can also be obtained via SNMP information. Figure 3.7 shows the screenshot of retrieving information on the console.


Raritan.
Dominion PX
Home | Details | Alerts | User Management | Device Settings | External Sensors | Maintenance | Outlet Groups | Diagnostics | Help

Home > PDU Status

Time & Session:
2013-07-17 19:16

User : admin
State : active
Your IP : 140.128.99.198
Last Login : 2013-06-25 04:39

Device Information:
Name: PDU
Model: PX (DPXR8-20)
IP Address: 140.128.101.161
Firmware: 01.05.05
Firmware Status: OK
FIPS mode is not set

Connected Users:
admin (140.128.99.198)
active

Power Cim State:
Power CIM is enabled

Help - User Guide

Line Loads

Line 1: 3.05 Amps

Outlets

Name	State	Control	RMS Current	Active Power	Group Member
32core1	on	On Off Cycle	0.84 Amps	86 Watts	no
24core	on	On Off Cycle	1.39 Amps	140 Watts	no
32core2	on	On Off Cycle	0.80 Amps	82 Watts	no
Outlet 4	on	On Off Cycle	0.00 Amps	0 Watts	no
Outlet 5	on	On Off Cycle	0.00 Amps	0 Watts	no
Outlet 6	on	On Off Cycle	0.00 Amps	0 Watts	no
Outlet 7	on	On Off Cycle	0.00 Amps	0 Watts	no
Outlet 8	on	On Off Cycle	0.00 Amps	0 Watts	no

All Outlets	Control
Switch all outlets	On Off

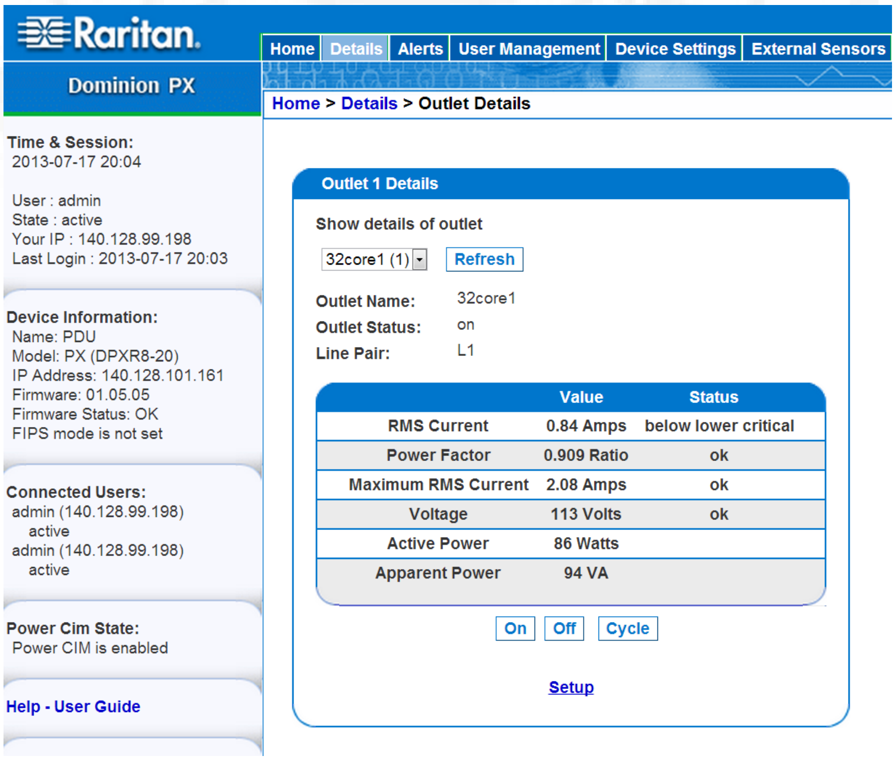
External Sensors

Number of managed sensor(s): 0

Number of unmanaged sensor(s): 0

Name	Reading	State

FIGURE 3.5: PDU's web interface


Raritan.
Dominion PX
Home | Details | Alerts | User Management | Device Settings | External Sensors

Home > Details > Outlet Details

Time & Session:
2013-07-17 20:04

User : admin
State : active
Your IP : 140.128.99.198
Last Login : 2013-07-17 20:03

Device Information:
Name: PDU
Model: PX (DPXR8-20)
IP Address: 140.128.101.161
Firmware: 01.05.05
Firmware Status: OK
FIPS mode is not set

Connected Users:
admin (140.128.99.198)
active
admin (140.128.99.198)
active

Power Cim State:
Power CIM is enabled

Help - User Guide

Outlet 1 Details

Show details of outlet

32core1 (1) Refresh

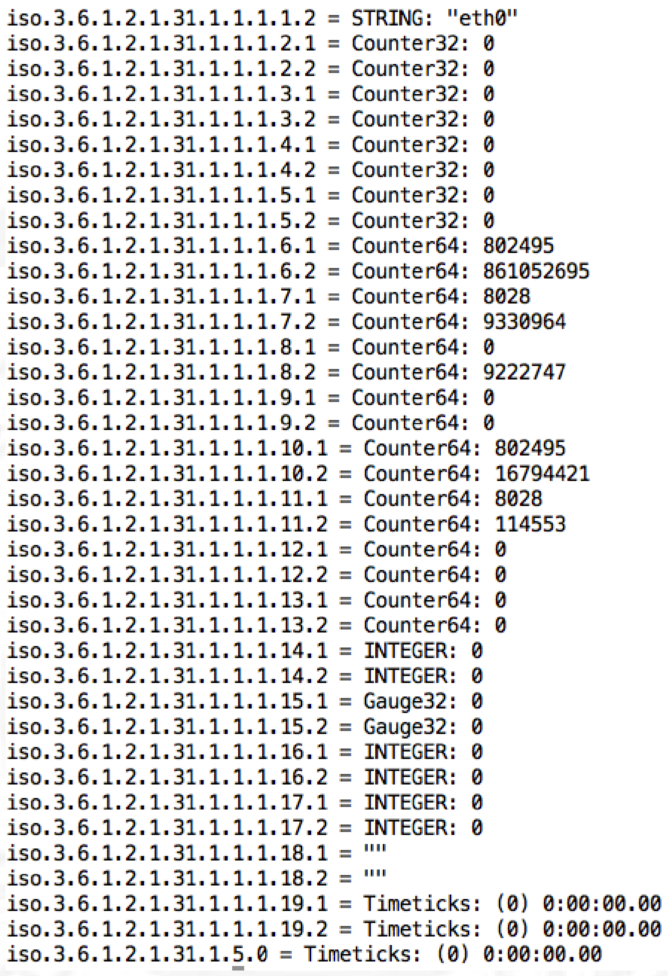
Outlet Name: 32core1
Outlet Status: on
Line Pair: L1

	Value	Status
RMS Current	0.84 Amps	below lower critical
Power Factor	0.909 Ratio	ok
Maximum RMS Current	2.08 Amps	ok
Voltage	113 Volts	ok
Active Power	86 Watts	
Apparent Power	94 VA	

On Off Cycle

Setup

FIGURE 3.6: PDU's web interface in details



```
iso.3.6.1.2.1.31.1.1.1.1.2 = STRING: "eth0"
iso.3.6.1.2.1.31.1.1.1.2.1 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.2.2 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.3.1 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.3.2 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.4.1 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.4.2 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.5.1 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.5.2 = Counter32: 0
iso.3.6.1.2.1.31.1.1.1.6.1 = Counter64: 802495
iso.3.6.1.2.1.31.1.1.1.6.2 = Counter64: 861052695
iso.3.6.1.2.1.31.1.1.1.7.1 = Counter64: 8028
iso.3.6.1.2.1.31.1.1.1.7.2 = Counter64: 9330964
iso.3.6.1.2.1.31.1.1.1.8.1 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.8.2 = Counter64: 9222747
iso.3.6.1.2.1.31.1.1.1.9.1 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.9.2 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.10.1 = Counter64: 802495
iso.3.6.1.2.1.31.1.1.1.10.2 = Counter64: 16794421
iso.3.6.1.2.1.31.1.1.1.11.1 = Counter64: 8028
iso.3.6.1.2.1.31.1.1.1.11.2 = Counter64: 114553
iso.3.6.1.2.1.31.1.1.1.12.1 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.12.2 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.13.1 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.13.2 = Counter64: 0
iso.3.6.1.2.1.31.1.1.1.14.1 = INTEGER: 0
iso.3.6.1.2.1.31.1.1.1.14.2 = INTEGER: 0
iso.3.6.1.2.1.31.1.1.1.15.1 = Gauge32: 0
iso.3.6.1.2.1.31.1.1.1.15.2 = Gauge32: 0
iso.3.6.1.2.1.31.1.1.1.16.1 = INTEGER: 0
iso.3.6.1.2.1.31.1.1.1.16.2 = INTEGER: 0
iso.3.6.1.2.1.31.1.1.1.17.1 = INTEGER: 0
iso.3.6.1.2.1.31.1.1.1.17.2 = INTEGER: 0
iso.3.6.1.2.1.31.1.1.1.18.1 = ""
iso.3.6.1.2.1.31.1.1.1.18.2 = ""
iso.3.6.1.2.1.31.1.1.1.19.1 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.31.1.1.1.19.2 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.31.1.5.0 = Timeticks: (0) 0:00:00.00
```

FIGURE 3.7: Capturing PDU's data by SNMP

The power consumption recording function, developed by the PHP programming language, is an automatic recording program, pdu_recorder.php, running on the management node. PDU supports SNMP to transmit electricity consumption data in the experiment. Through SNMP, the automatic recording program on the management node automatically collects PDU power consumption data of computing nodes every minute and records them in the pdurecord.txt file.

Algorithm 3.4.2: POWER RECORDING(c)

```
while in record range
  do if second = 0
    then { get PDU information via snmp
         { write into pdurecord.txt
```

3.4.3 Power-saving method

The main function of the power-saving method is to obtain and analyze resource usage information of servers and VMs through the resource status monitoring program. If the required resources of VMs are less than those supported by currently running servers, through libvirt live migration instructions the VMs are centralized and some server is shut down to achieve the goal of energy saving. And as the resource requirement of the whole system increases, some of standby servers might be awaken to join the computing cluster, and appropriate live migrations of VMs are performed among the operating servers.

Algorithm 3.4.3: Power-saving method(c)

```
$sumvm \leftarrow all\ VM's\ computation\ sum$
$sumhost \leftarrow all\ host's\ computation\ sum$
$maxvmn \leftarrow vm\ with\ maximum\ computation\ on\ host\ n$
$maxhost \leftarrow host\ with\ largest\ VM's\ computation\ sum$
$minhost \leftarrow host\ with\ smallest\ VM's\ computation\ sum$
$minsumhost \leftarrow host\ with\ smallest\ computation\ sum$
if $sumvm > sumhost$
  then { $start\ an\ host\ in\ shutdown\ status$
         $migrate\ maxvmmaxhost\ to\ the\ host\ just\ start$
  else if $sumvm < sumhost - minsumhost$
  then { $migrate\ all\ VM\ on\ minhost\ to\ other\ host$
         $shutdown\ minhost$
```

3.5 User Interface

In this work we also used a web interface to allow users easily manage VMs, perform live migration for VMs, and obtain information of machine states and electric power consumptions.

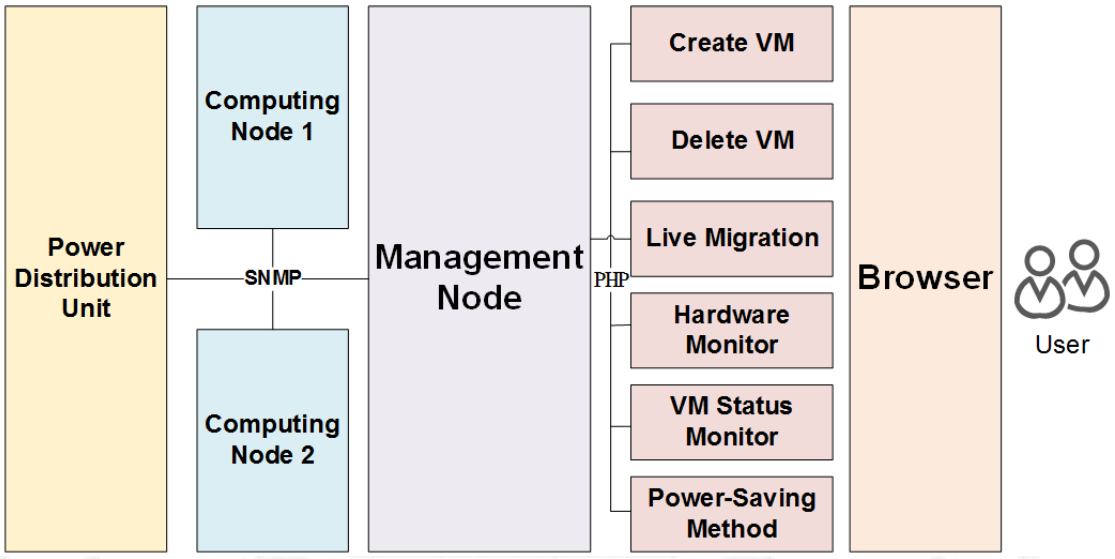
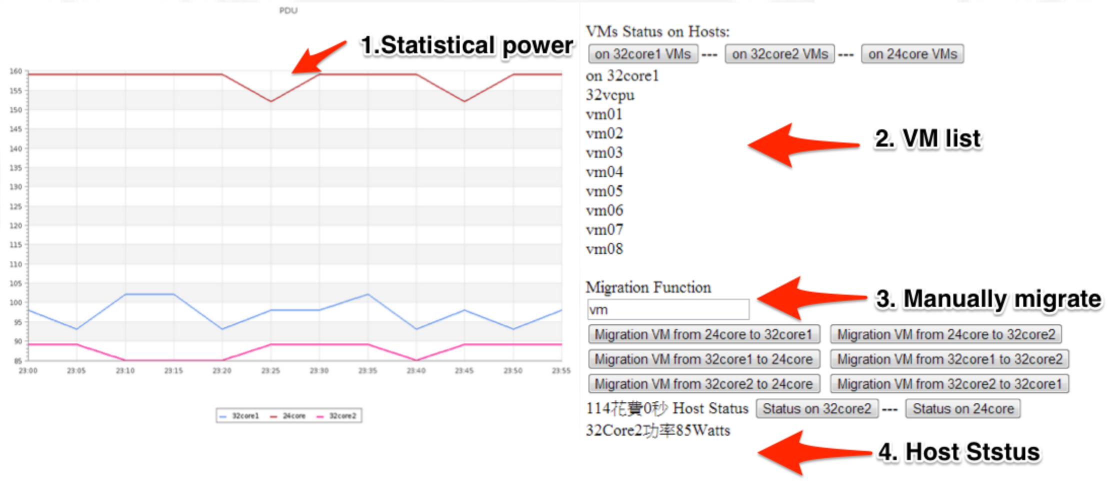


FIGURE 3.8: Service architecture


PDU
1.Statistical power
2. VM list
3. Manually migrate
4. Host Ststus
on 32core1 VMs
on 32core2 VMs
on 24core VMs
vm
Migration VM from 24core to 32core1
Migration VM from 24core to 32core2
Migration VM from 32core1 to 24core
Migration VM from 32core1 to 32core2
Migration VM from 32core2 to 24core
Migration VM from 32core2 to 32core1
Status on 32core2
Status on 24core
32core1
24core
32core2

VMs Status on Hosts:
on 32core1
32vcpu
vm01
vm02
vm03
vm04
vm05
vm06
vm07
vm08

Migration Function

114花費0秒 Host Status
32Core2功率85Watts

FIGURE 3.9: Web interface

The user interface shown in Figure 3.9 is developed by the PHP programming language, and run on the management node. Its key features include power consumption monitoring (on the left), management of servers and virtual machines (on the right), and resource monitoring as described in detail below.

The power consumption of machines can be viewed in one day or one hour periods. By default, the system displays power consumption data recorded every five minutes for the last one hour. Alternately, the user can switch the display mode to see power consumption data recorded every hour for the past 24 hours.

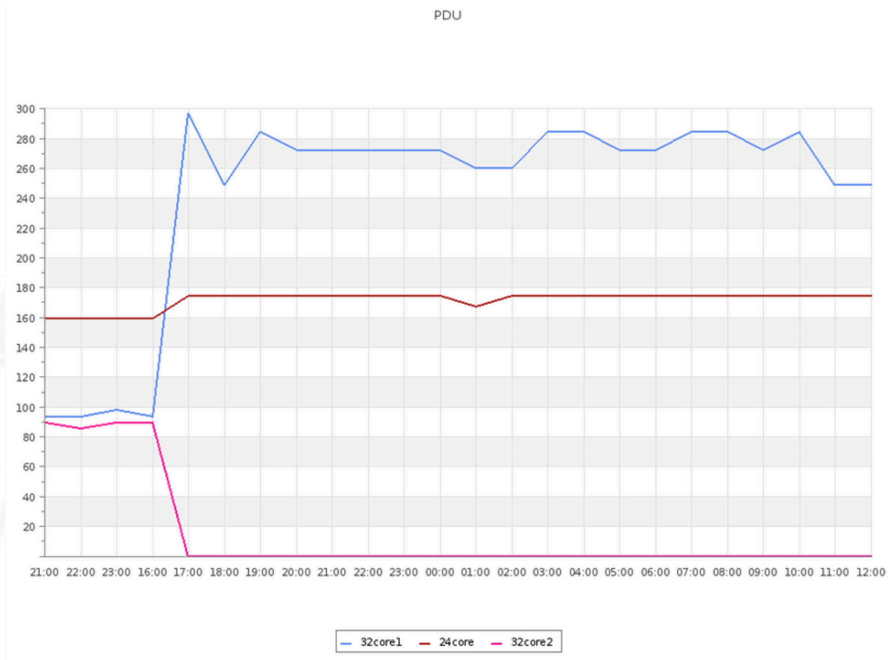


FIGURE 3.10: Power consumption viewed in a day

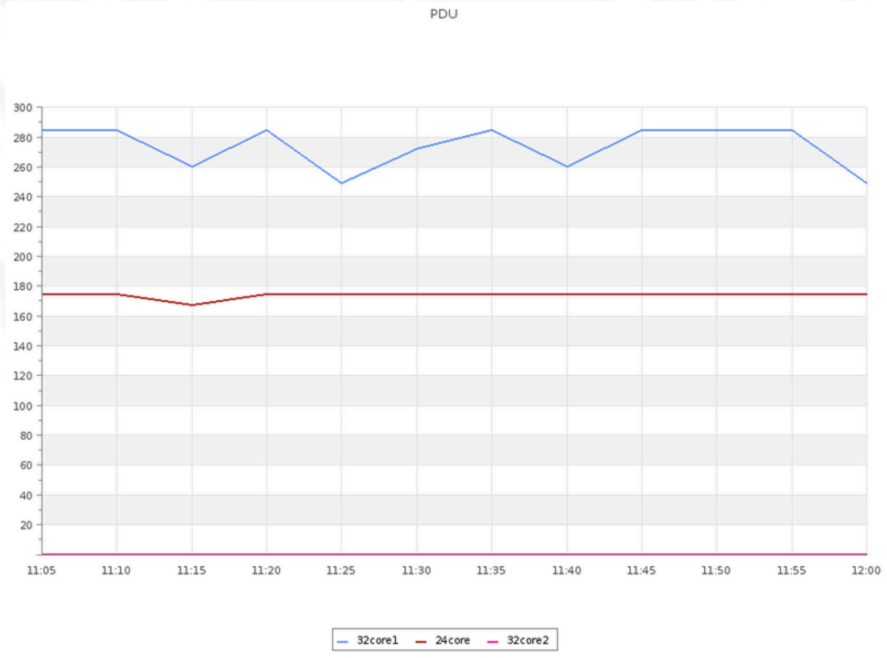
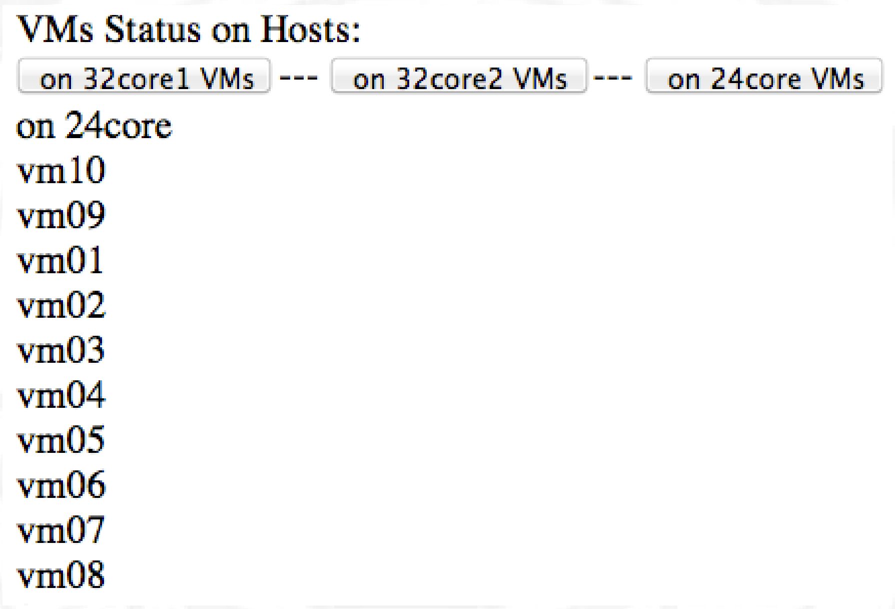


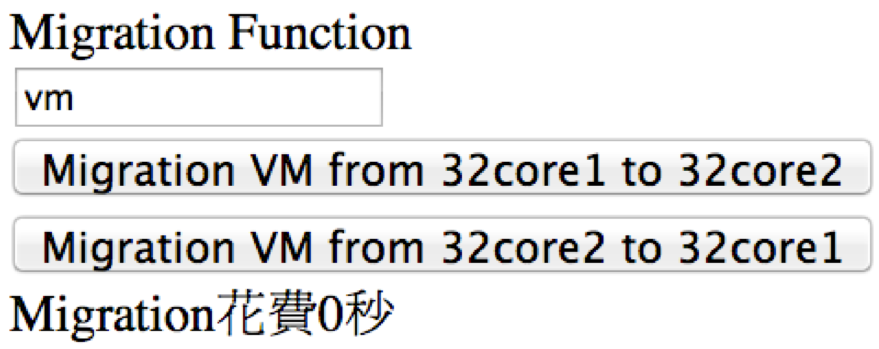
FIGURE 3.11: Power consumption viewed in one hour

Figure 3.12 shows the feature to obtain the virtual machine list, which displays active virtual machines on every server. Figure 3.13 shows the manual migration feature. After obtaining the name of the virtual machine through the query function, name of the virtual machine is typed in the input box, and then the desired destination server is selected. When live migration is complete, the time spent will be displayed below. Figure 3.14 shows the server monitoring function, which displays the current CPU and memory usage status. The function can also be used to display data separately for each server, including the CPU usage, idle CPUs, the total size of memory, and the available memory size.



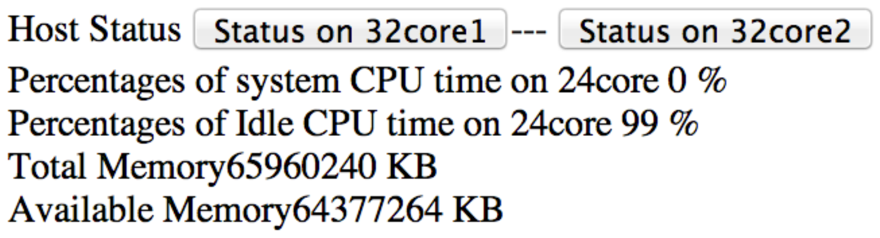
on 24core
vm10
vm09
vm01
vm02
vm03
vm04
vm05
vm06
vm07
vm08

FIGURE 3.12: Display VM lists on servers


Migration Function
vm
Migration VM from 32core1 to 32core2
Migration VM from 32core2 to 32core1

Migration花費0秒

FIGURE 3.13: Manual migration function



Percentages of system CPU time on 24core 0 %
Percentages of Idle CPU time on 24core 99 %
Total Memory65960240 KB
Available Memory64377264 KB

FIGURE 3.14: Monitoring server s status

3.6 Design of Equations

In this work, we present an method to turn off computing nodes to achieve the power-saving purpose. However, the transfer of VMs among computing nodes results in extra time and power consumption. To consider the cost, we formulate a way to calculate the operating costs of the power-saving method. The net efficiency of the power-saving method is found by subtracting the expected power-saving method by the operating costs of the method. This subsection focuses on relationship between the additional operating costs and the efficiency of the power-saving method. The following equations and variables are used to define the operating costs:

3.6.1 Migration Costs

The migration cost is the cost of performing live migration of VMs. Relevant parameters of the migration costs are defined as follows:

$C_{migration}$ is defined as Costs resulting from migration

$P_{standby}$ is defined as Average power spend by standby computing node

$T_{downtime}$ is defined as VM downtime during live migration

$P_{migration}$ is defined as Extra power cost by processing migration

$N_{migrationvm}$ is defined as Number of VMs need to be migrated

The power-saving method may perform live migration on a number of VMs. In the migration process, no service is provided by a VM during the brief downtime of the VM, which contributes to additional costs of migration, and the extra cost of electricity can be calculated by multiplying the downtime of the VM with the average standby power consumption of the server. Also we need to include the additional power consumption occurs at the migration action of each VM. Thus, the migration cost can be calculated with the following formula:

$$C_{migration} = [(P_{standby}?T_{downtime}) + (P_{migration})]?N_{migrationvm} \tag{3.1}$$

3.6.2 Merging and Expanding Costs of Computing Nodes

The merging or expanding cost of computing nodes is considered here; its relevant parameters are defined as follows:

C_{maa} is defined as Costs resulting from merging and expanding computing nodes

T_{bootup} is defined as Time spend in booting up computing nodes

$T_{shutdown}$ is defined as Time spend in shutting-down computing nodes

S is defined as Level of service preset by user

$P_{probability}$ is defined as Probability of processing merging or expanding

In the power-saving method, turning off unnecessary servers to reduce electricity is most efficient for power-saving, but frequent switching of servers also results in additional power costs. Since servers do not provide services during the startup and shutdown time, the costs of booting up and shutting down servers are considered as additional costs, and are calculated by multiplying both times with the average power consumption of the standby server. In order not to reduce service quality due to frequent switching of machines, user can follow the Service-level agreement (SLA) to control the switching frequency. The higher SLA is, the lower the switching frequency becomes.

$$C_{maa} = [(T_{bootup} + T_{shutdown})?P_{standby}]?P_{probability} \tag{3.2}$$

3.6.3 Net Power Saving

Finally, the net power saving can be calculated in the following.

T_{off} is defined as Total time that computing node in power off state

P_{save} is defined as Total power saving by method

Turning off unnecessary servers to reduce power consumption is most efficient; thus, the expected power saving is calculated by multiplying the total shutdown time with the average standby power consumption of servers. And the net power saving of the proposed method can be calculated by the following formula:

$$P_{save} = (P_{standby}?T_{off}) - C_{migration} - C_{maa} \tag{3.3}$$

Chapter 4

Experimental Results

4.1 Experimental Environment

The experimental environment is shown in Table 4.1. The main monitoring function, the method, the judgment function of the shared storage system, the virtual machine management platform, and the user interface were built on the management node, which consists of 24 core CPU, 78GB memory and 1TB disk. The virtual machines were distributed in two servers: computing node 1 and computing node 2, with identical specifications, both of them connected to the PDU to collect electricity consumption data. The management function was evaluated by calculating the functional dependence of such a configuration, management capabilities, and operation efficiency of virtual machines. The experiment focused on the required resources for operation of the virtual machines, such as CPU, memory, and power consumption. In order to enhance the reliability of the experimental data, ten VMs with the same specifications were built on the computing nodes, with preloaded SNMP and cpuburn programs. The detailed specifications of the three servers and ten virtual machines are listed in Table 4.1, and the detailed specifications of software used include KVM, libvirt, PHP and SNMP are listed in Table 4.2.

Table 4.1: Hardware Specification

Host name	CPU	Memory	Disk	OS
Management node	AMD Opteron(TM) Processor 6174 x 24 cores	78GB	1TB	Ubuntu 12.04
Computing node1	AMD Opteron(TM) Processor 6274 x 32 cores	48GB	Shared storage	Ubuntu 12.04
Computing node2	AMD Opteron(TM) Processor 6274 x 32 cores	48GB	Shared storage	Ubuntu 12.04
VM01-VM10	8 cores vCPU	8GB	10GB	Ubuntu 12.04

Table 4.2: Software Specification

Software	KVM	Libvirt	PHP	SNMP
Version	3.2.0	0.9.8	6.3.10	5.4.3

4.1.1 Experimental Flow

After the implementing the complete environment, we performed simulations for possible scenarios on the cloud environment. We gradually increased the CPU usage of each virtual machine. If the requirement of the CPU usage is less than that offered by the servers, the system automatically triggers the power-saving method. For comparison, we recorded power consumptions of the overall cloud environment with or without enabling the power-saving method.

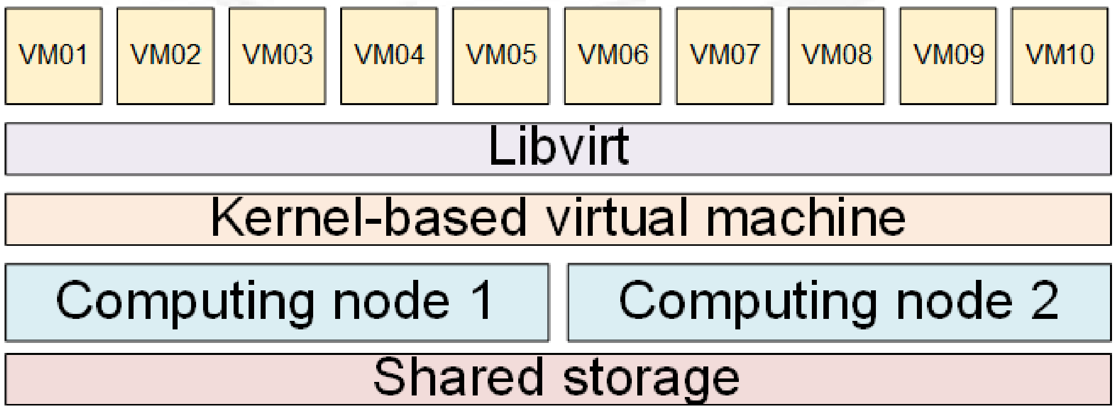


Figure 4.1: Software Architecture

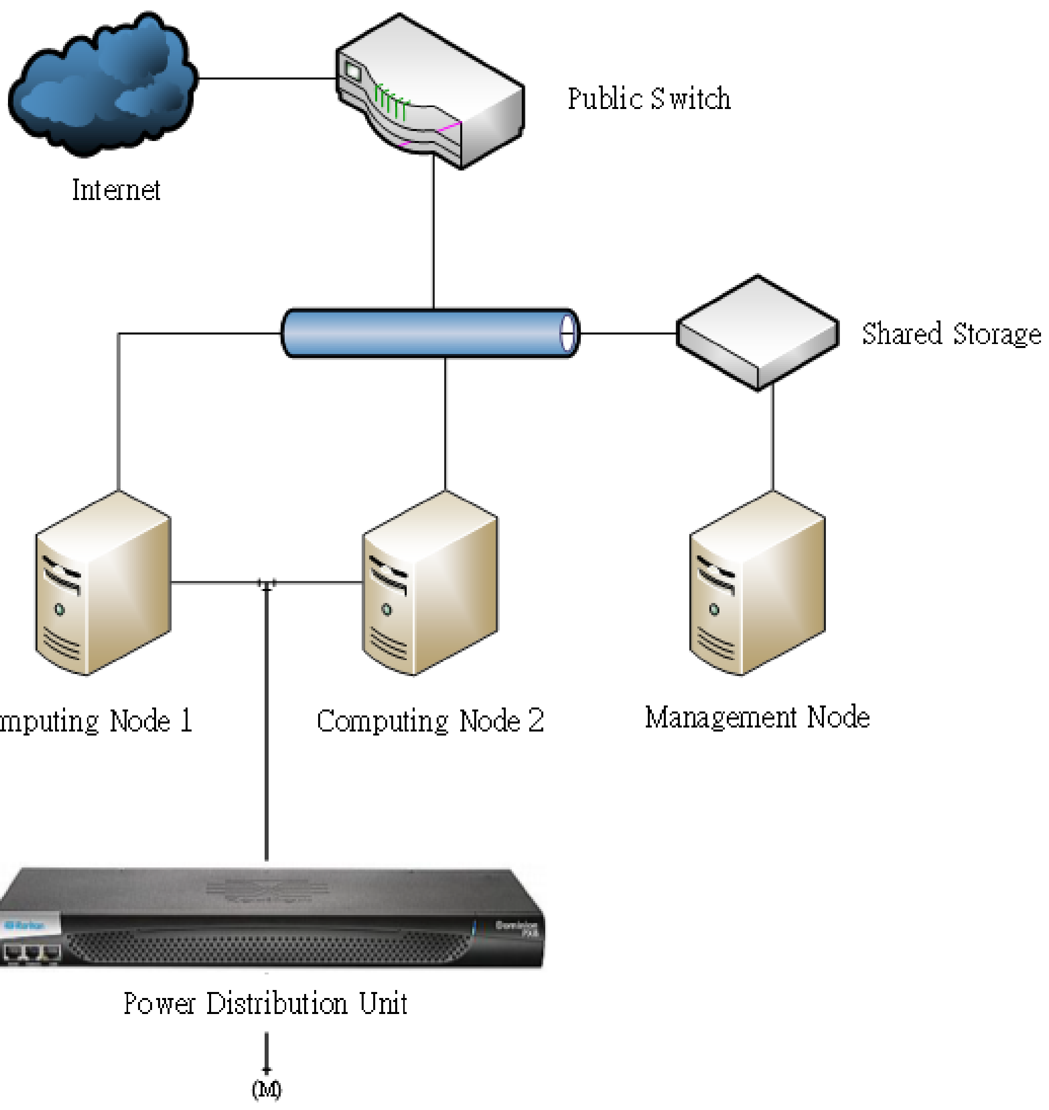


FIGURE 4.2: Experimental Architecture

The average power consumptions of the two computing nodes with virtual machines were recorded. In the first 10 minutes, the two servers did not run any other program besides the standby virtual machines. Then, every five minutes, CPU loading of the virtual machines on the two computing nodes was one by one turned to be full-loaded; thus at 60 minute of the experiment all 10 vCPUs (i.e. virtual CPUs) were full-loaded. Then every five minutes turned one of the virtual machines on the computing nodes back to standby; thus at 110 minutes all 10 virtual machines were in the standby state. We continued recording power consumptions of the two nodes for another ten minutes.

We then performed the above experiment again, but this time applied the power-saving method on the two computing nodes from the onset of the experiment. We expected at the start the 10 virtual machines would be automatically moved to one of the two computing nodes, and the other computing node will be automatically shut down. We also expected that the power-saving method will automatically wake up the standby computing node when five or more vCPUs were full-loaded. The power-saving method would perform live migration on chosen machines until equilibrium was arrived. We expected when the number of virtual machines decreased to be five or less, the power-saving method would again automatically move all virtual machines to one of the computing nodes, and shut down the other computing node.

4.2 Results and Discussion

4.2.1 Prior Experiments

Since the power-saving method experiment involves several factors e.g. relation between live migration, VM vCPU loading, migration time and power consumption, we first carried out several experiments to find answers for following questions:

1. Will virtual machine's vCPU loading affect the time required for live migration?

2. Will CPU loading of the source host or target host affect the required live migration time of virtual machines?

3. Will live migration cause additional power consumption?

The following discusses the results of three experiments. In the first experiment, all 10 VMs have various CPU loading 0%, 25%, 50%, 75% and 100%, and live migration of VMs are performed between two computing nodes. The figure below shows our first experimental data:

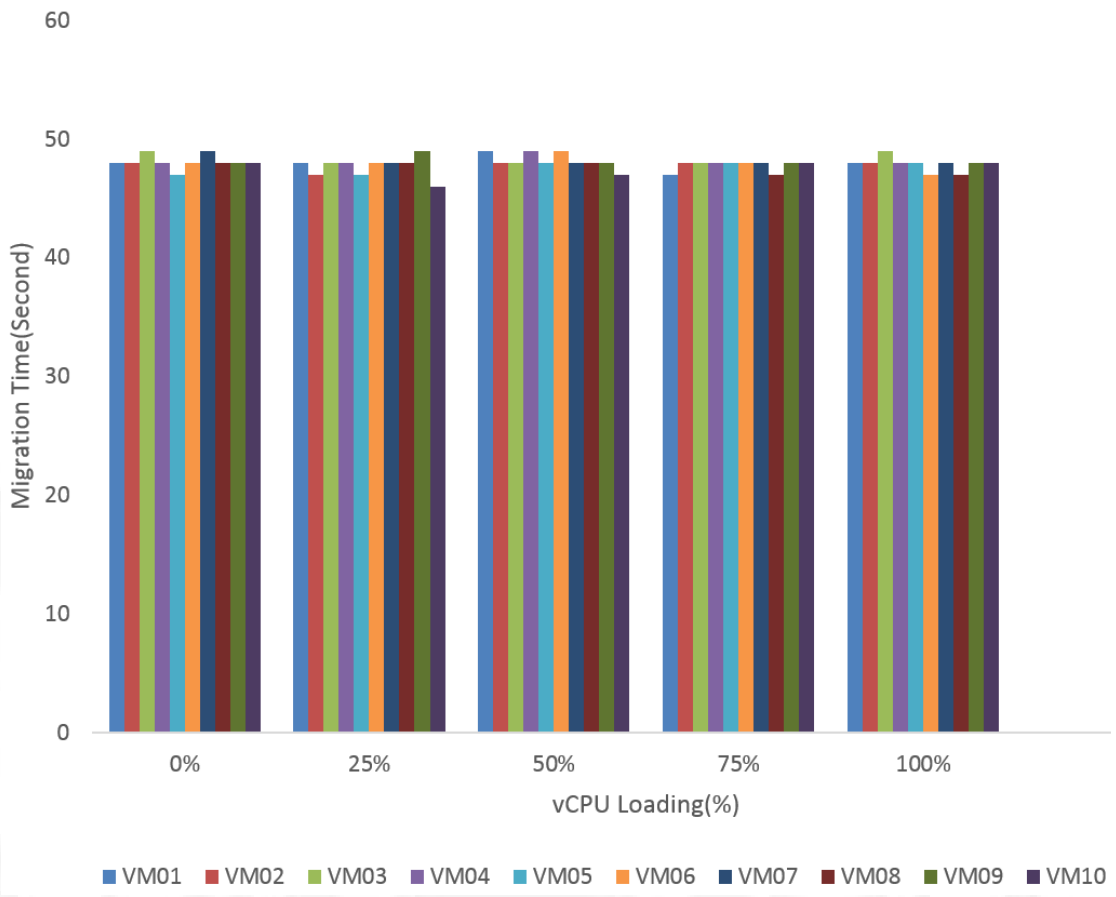


FIGURE 4.3: Relationship between Migration Time and vCPU Loading

From Figure 4.3, no matter how much CPU loading is, performing live migration of VMs required about 48 seconds; thus there is no direct relationship between migration time and vCPU loading. It rules out the assumption that different CPU loading affect migration time in the experiment of the power-saving method.

In the second experiment, we perform live migration of VMs and observe the change of the power consumption of the source host and target host. The experimental results are shown below:

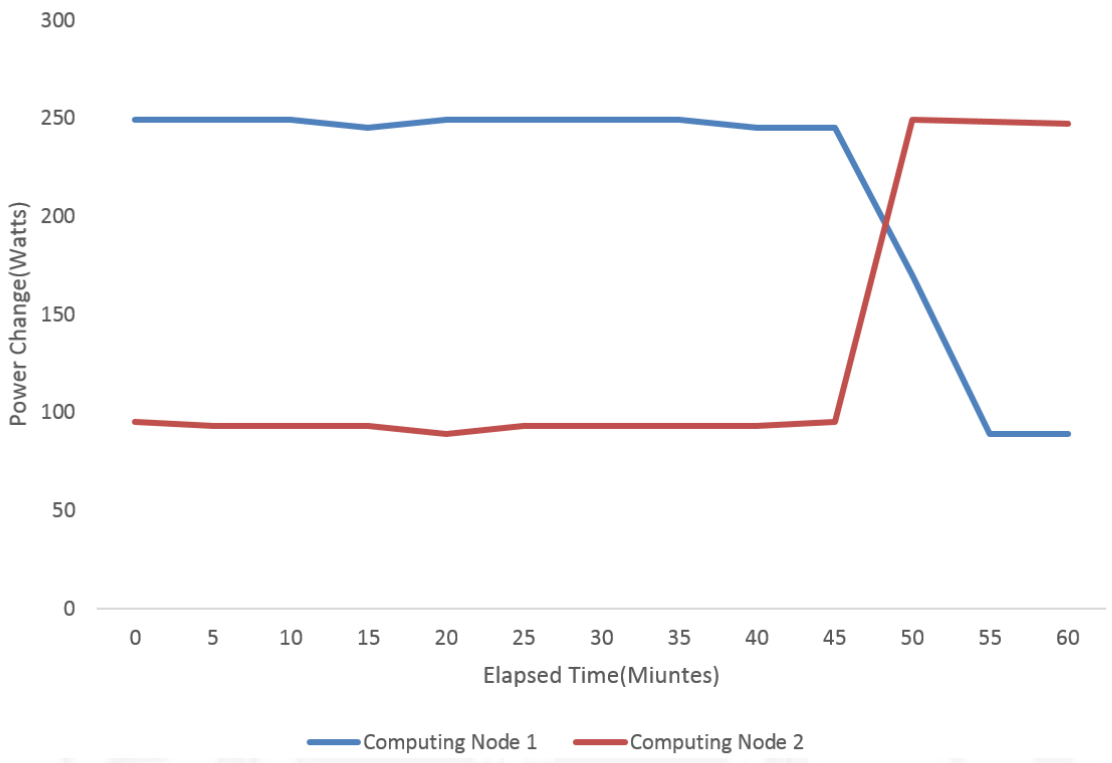


FIGURE 4.4: Change in Power Consumption During Migration

From the experiment we know that the average time of VM migrations is approximately 48 seconds. From Figure 4.4, two hosts' power consumption are very stable during the live migration process, and there is no significant ups and downs. In this experiment, VM's vCPU loading are 100%, and at about 50 seconds after live migration is finished, the power consumption of the source host and target host's status varied significantly. This experiment also proves that after live migration, the virtual machine's vCPU loading effects on the source host and the target host are very fast.

The third experiment studies the effect of the live migration time due to the CPU loading of the source host and target host. Virtual machines' vCPU loading of the hosts in the experiment is changed in the following ways:

1. Change the source host and target host's CPU loading at the same time.

2. Only change the source host's CPU loading.

3. Only change the target host's CPU loading.

The live migration experimental results are shown in the following figure:

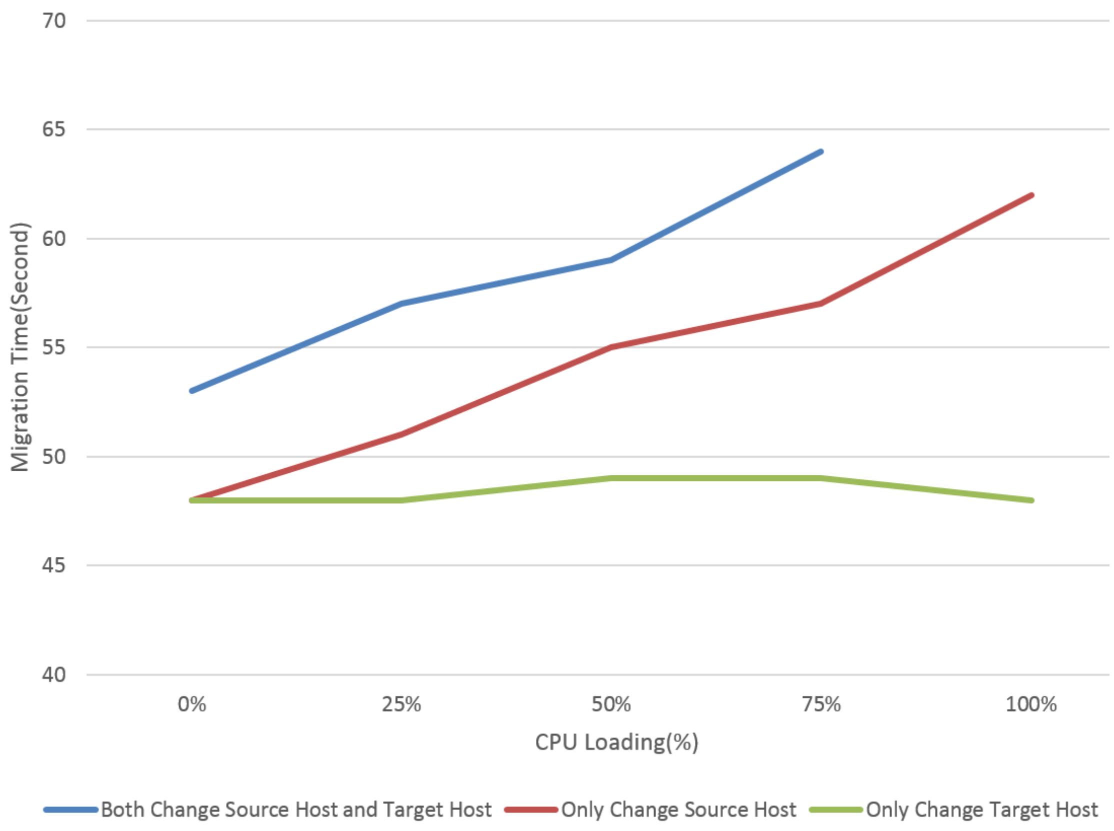


FIGURE 4.5: Relationship between Migration Time and Host's CPU Loading

From Figure 4.5 , one observes that migration time is not related to CPU loading of the target host, but is proportional to the CPU loading of the source host; in particular, when CPU loading of the source host and the target host are changed at the same time, it has extra two seconds average time.

The above experiment proves that migration time is affected by the host's CPU loading, but the saving benefits of the power-saving method are not affected. We will discuss about the time cost later.

4.2.2 Performance of Power-Saving Experiments with 10 VMs

The experiments of the power-saving method are done here. As described in Section 3, the vCPU loading of vm01 to vm10 were sequentially increased up to

100% one by one in every five minutes, and then sequentially decreased the VM vCPU loading back to 0%. The following experimental results shown in Figure 9 to Figure 11 were measured when the system was without or with the power saving method:

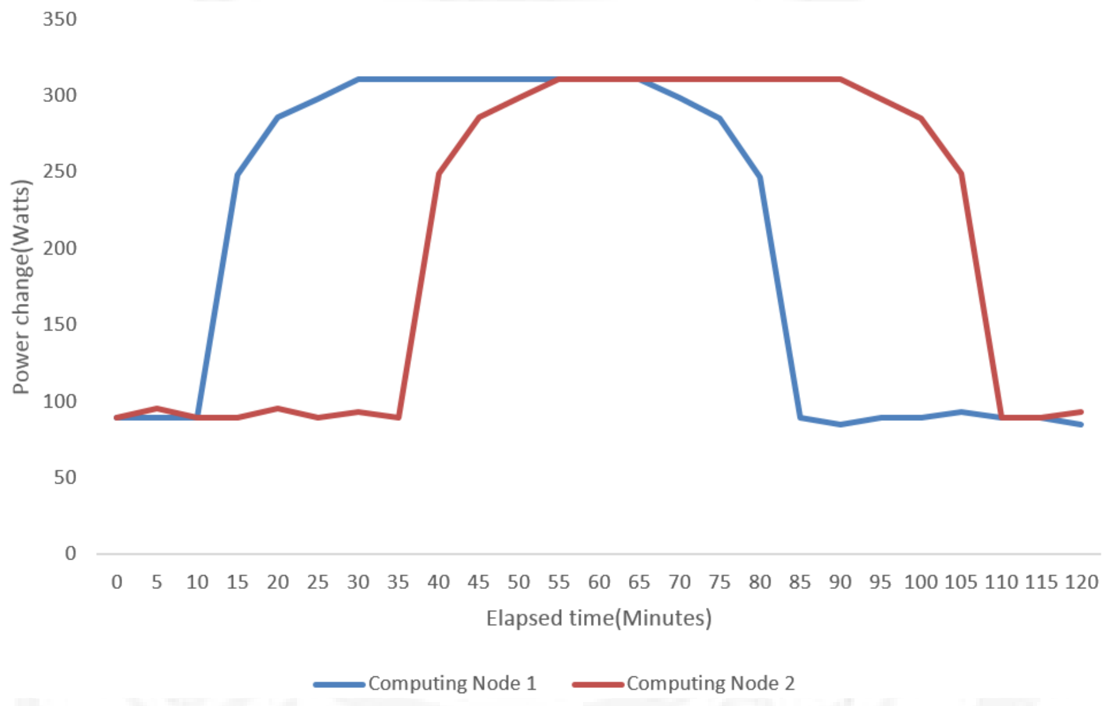


FIGURE 4.6: 10 VMs Result without Power-saving Method

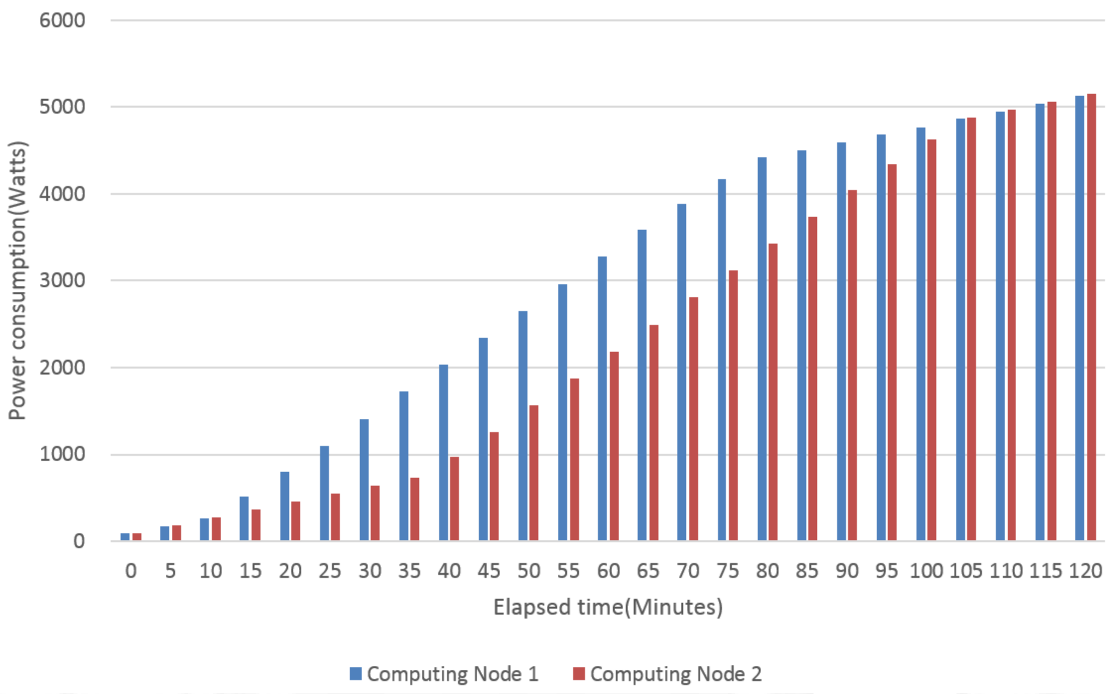


FIGURE 4.7: 2 Computing Node's Cumulative Power Consumption without Power-Saving Method with 10 VMs

Figure 4.6 shows curves of power consumptions of the two computing nodes without power-saving method. The power consumptions of the two computing nodes are observed to have the same trends in response of the increase and decrease of loading of VM vCPUs. Figure 4.7 shows the cumulative power consumptions of the two computing nodes. From the plot one can see that the cumulative power consumptions of them are similar.

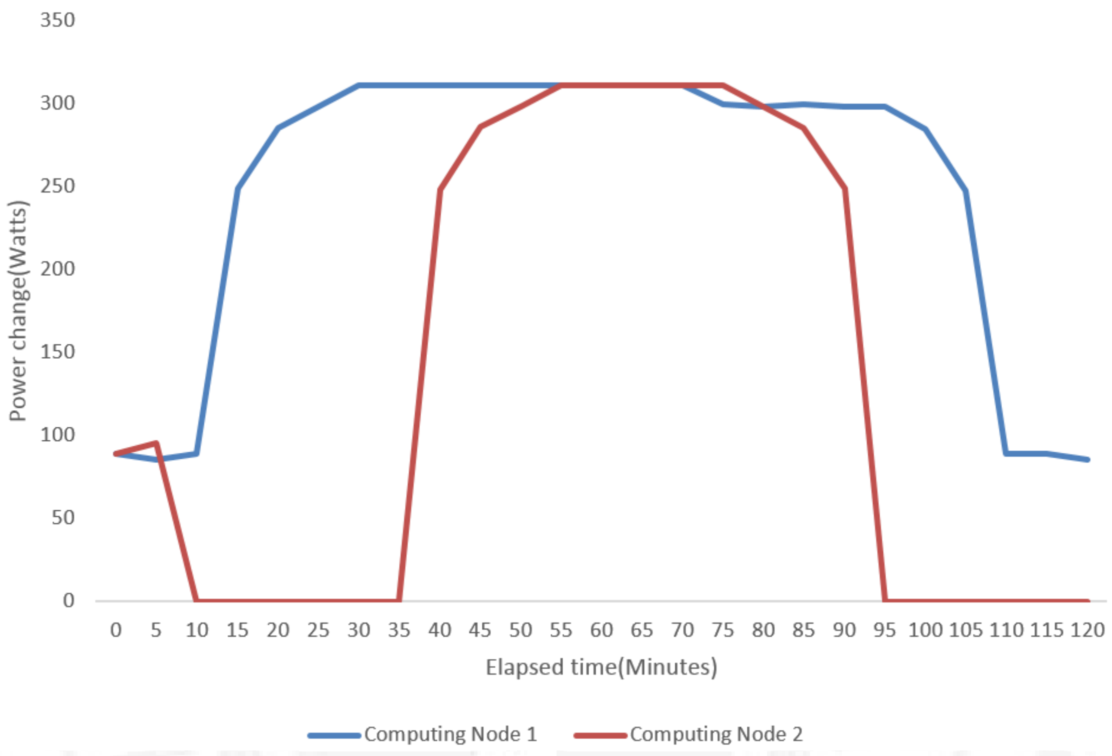


FIGURE 4.8: 10 VMs Result with Power-saving Method

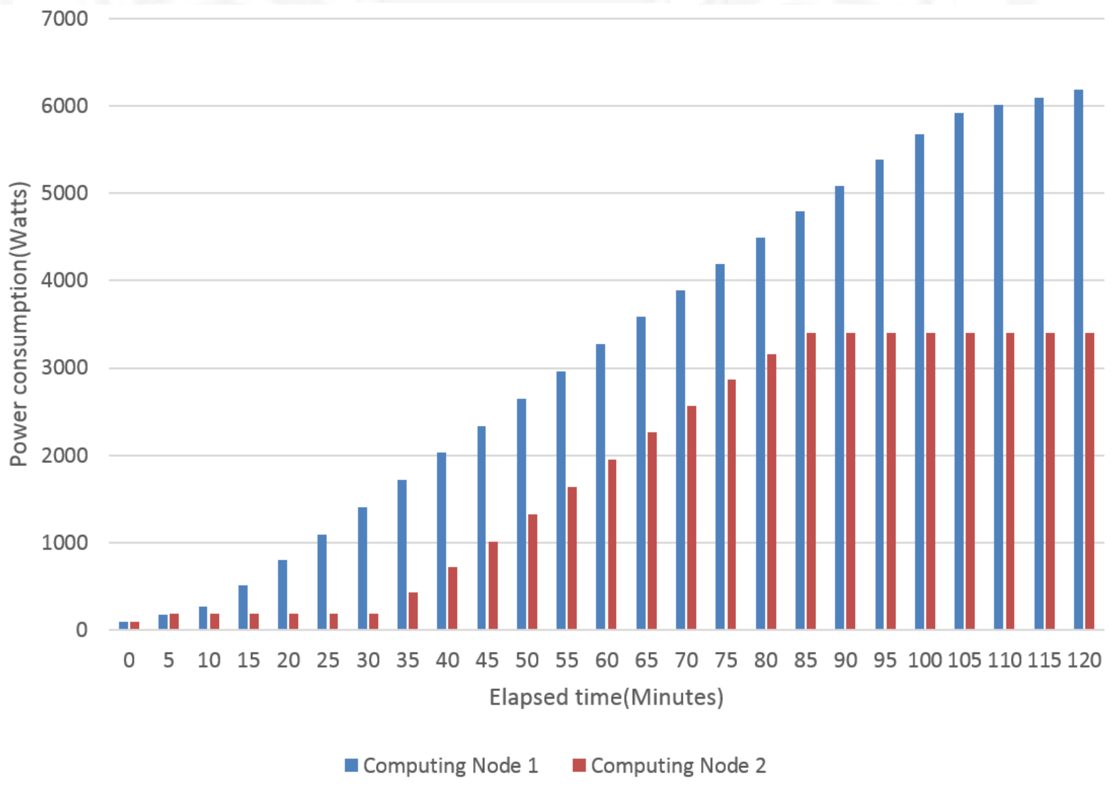


FIGURE 4.9: 2 Computing Node's Cumulative Power Consumption with Power-Saving Method with 10 VMs

Figure 4.8 shows curves of power consumptions of the two computing nodes with the power saving method. In the beginning of the first ten minute, the method determined that the ten standby VMs would not need to use all the computing power of the two computing nodes, so it transferred all the VMs, i.e., five VMs, on computing node 2 to computing node 1. Due to the delay of the judgment and migration time for the five VMs (about five minutes) and the shutdown time of computing node 2 (about one minute), computing node 2 continued consuming the electric power until ten minute of the experiment.

A similar situation appeared at 35 minute: as designed, VM05's vCPU would be fully loaded at 30 minute and it should be beyond the maximum CPU support of computing node 1, but due to delay of the power-saving method, the VM selection and migration time, and the booting time of computing node 2, so the power consumption data did not respond to the change of CPU loading until 40 minute of the experiment. This work also observed similar delay of change of the curve for computing node 2 at 90 minute of the experiment.

Figure 4.9shows the cumulative power consumptions of the two computing nodes with the power-saving method. From the plot one can see that the cumulative power consumptions of the two computing nodes are different and the power saving effect on computing node 2 is obvious.

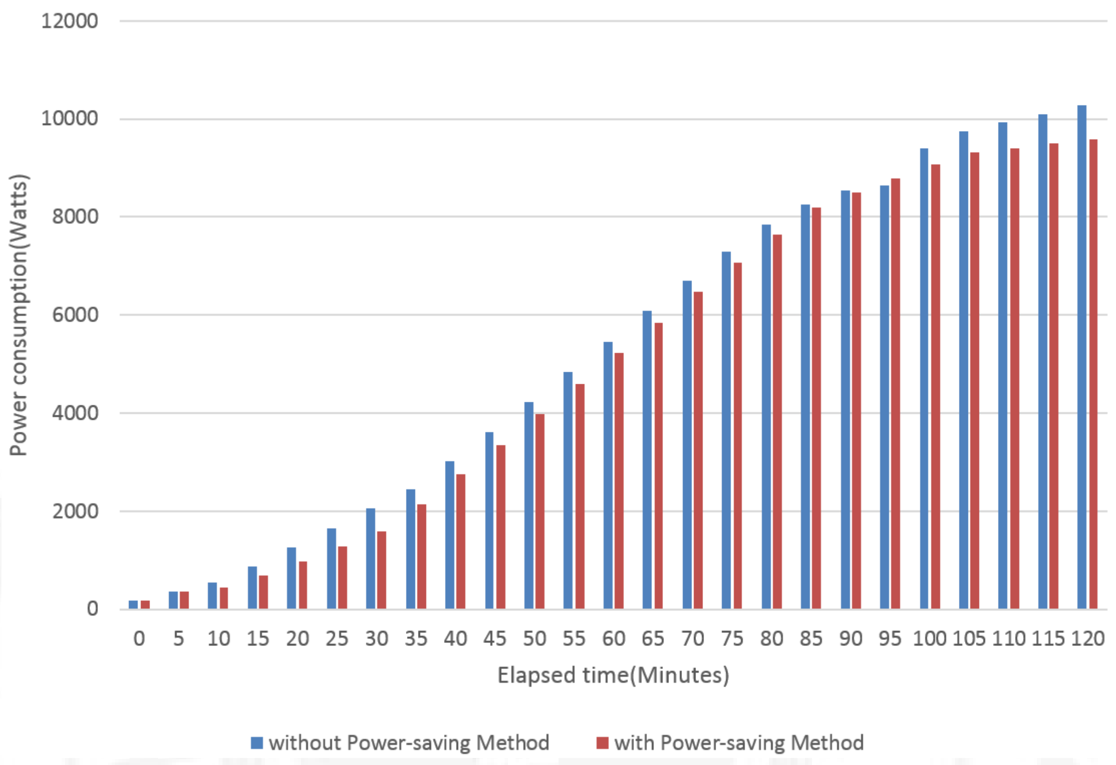


FIGURE 4.10: Cumulative Power Consumption difference between with and without Power-saving Method with 10 VMs

Figure 4.10 shows the difference between cumulative power consumptions of the nodes with or without the power-saving method. It is obvious that the cumulative power consumption of nodes with the power-saving method is indeed lower than that of nodes without the power-saving method. In this experiment, the method is found to save about 7% of the consumption power; unfortunately, the degree of power-saving is not obvious, following reasons are provided:

1. In the experimental with the method, when the VM's total vCPU loading is greater than the current available VMs provided by the computing node, then expanding of computing nodes is proceeded and the VMs to be migrated is determined in accordance with the VM vCPU loading and other factors. In our experiments, VMs are randomly selected to reduce the VM's vCPUs loading, thus, CPU loading of some computing node cannot be effectively reduced.

2. Because 10 VMs with 8 core vCPU were used in this experiment, in theory a computing node can only load 4 VMs with full loading vCPU; therefore, if 10 VM's vCPU are all full, a computing node runs six VMs and cannot be migrated.

Continuation of the first point, if the first and second VM selected to reduce the vCPUs loading are both in this node, the experimental data will not be conducive to the power-saving method.

3. Because the number of VMs is 10, and in the experiment the power is recorded in a time interval of five minutes, then transfer 5 VMs or 4 VMs plus the shutdown time does not reflect which one has the more favorable data in the power consumption data.

4.2.3 Performance of Power-Saving Experiments with 8 VMs

In order to reduce the effects mentioned above, experiments with similar setting as previous experiments were done; except, this time the number of VMs were reduced to 8. The following are the experimental results:

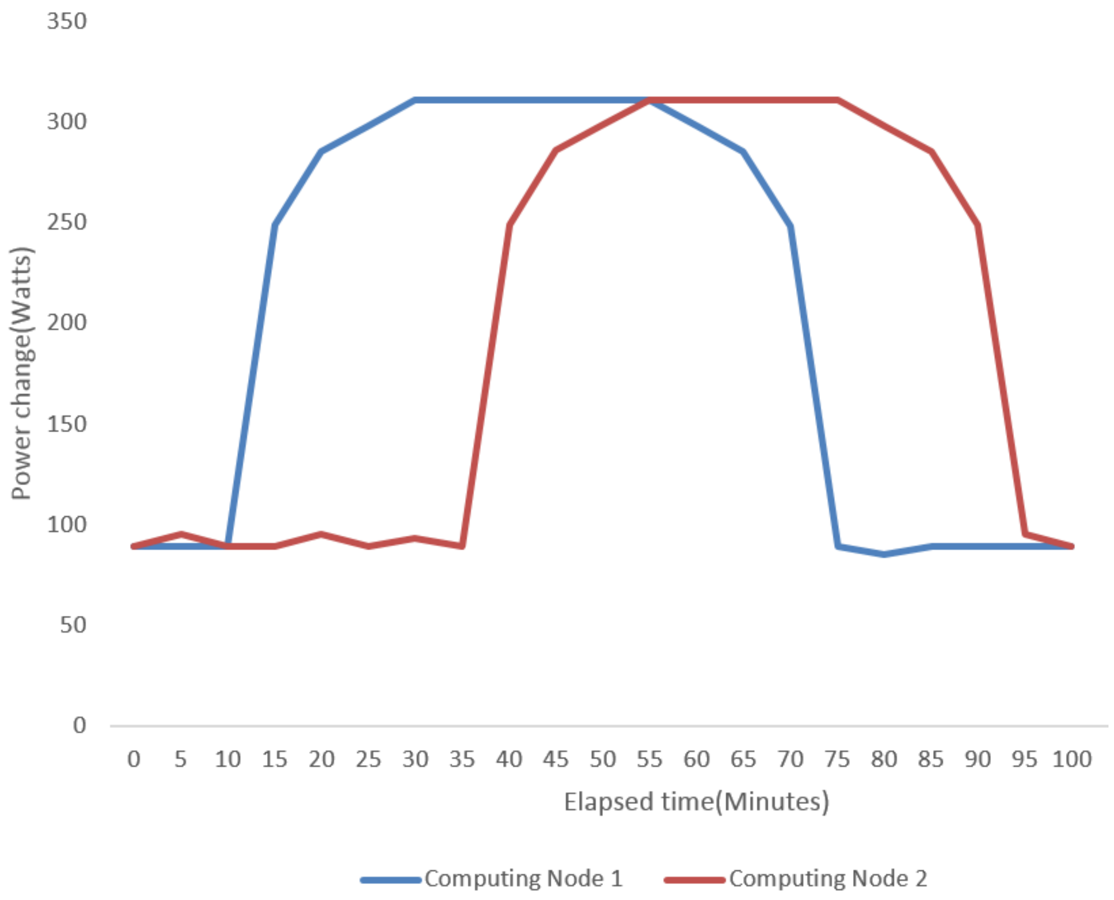


FIGURE 4.11: 8 VMs Result without Power-saving Method

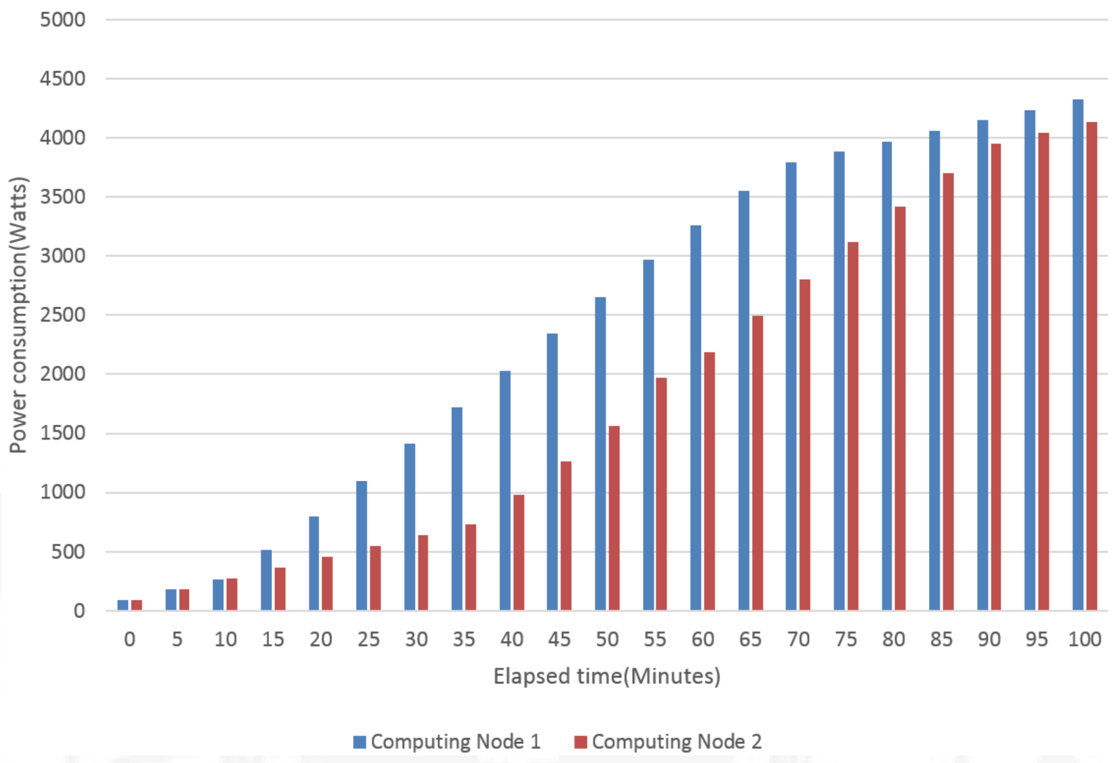


FIGURE 4.12: 2 Computing Node's Cumulative Power Consumption without Power-Saving Method with 8 VMs

Figure 4.11 and Figure 4.12 shows the power consumption without the power-saving method, which is quite similar with the results of experiment with 10 VMs. Similarly, with the vCPU rises and falls, two servers' power consumptions are also up and down, and the two servers' total power consumption statistics are very similar. Because the power-saving method is not used, when the vCPU is not in the fully loaded condition, the system does not detect the servers are in a low efficient usage state, thus causing unnecessary power waste.

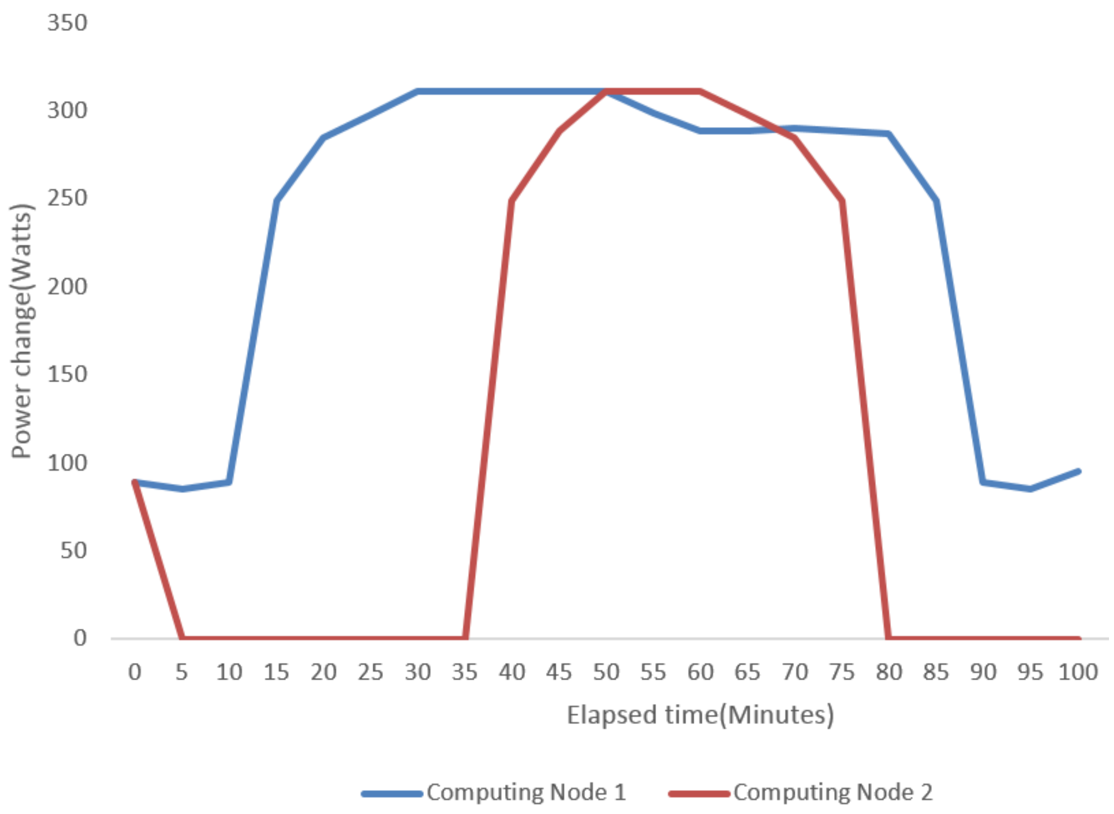


FIGURE 4.13: 8 VMs Result with Power-saving Method

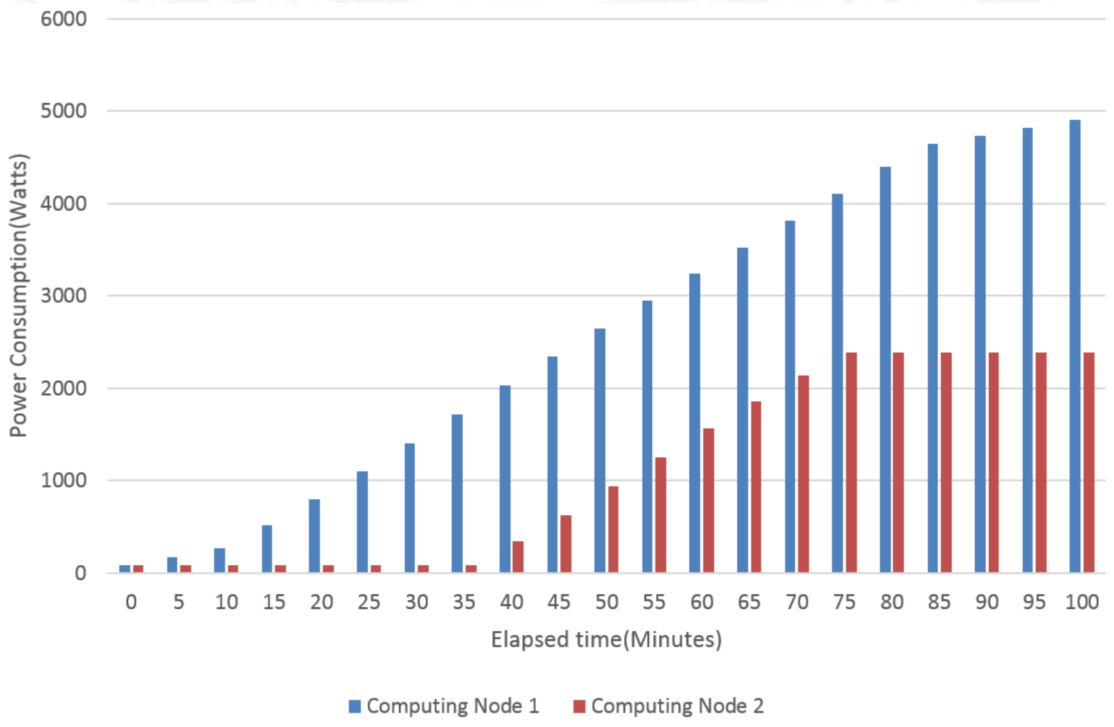


FIGURE 4.14: 2 Computing Node's Cumulative Power Consumption with Power-Saving Method with 8 VMs

Figure 4.13 and Figure 4.14 shows the power consumption with the power-saving method. The curves are similar to the previous experiments. Because the required migration time is shorter, in the 5 minute time point, the power consumption was recorded as 0, meaning that the computing node 2 has been shut down. Similarly, power consumptions recorded in several other time shows the effectiveness of the proposed power-saving method.

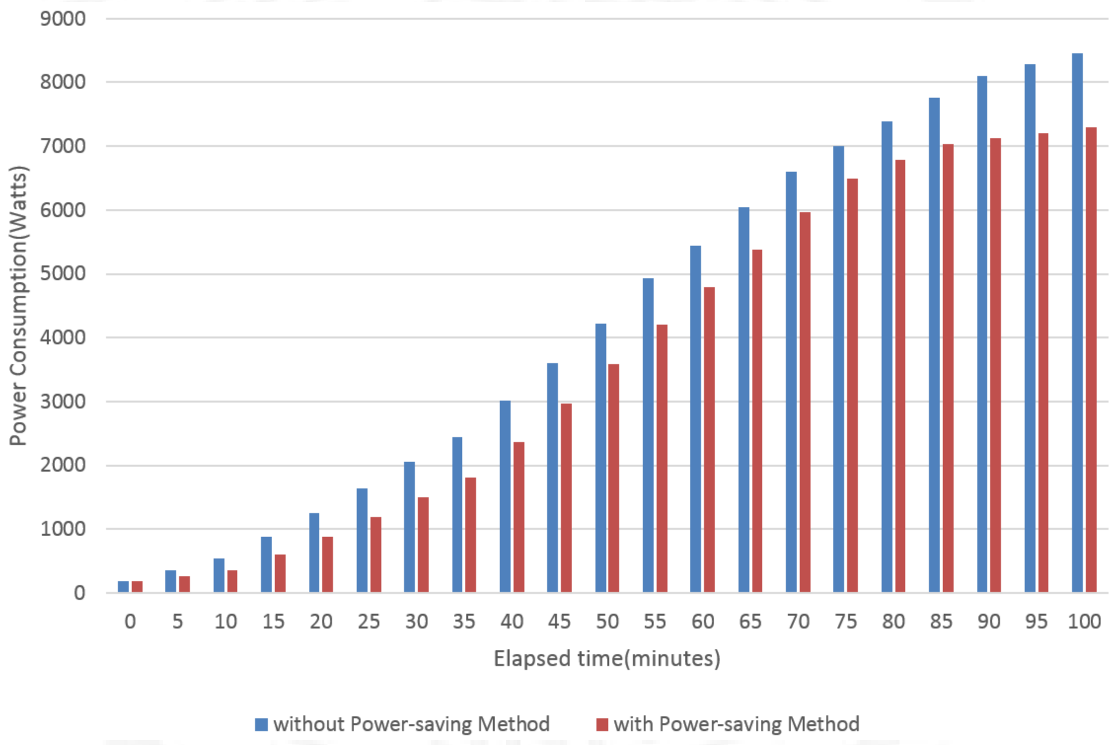


FIGURE 4.15: Cumulative Power Consumption difference between with and without Power-saving Method with 8 VMs

Figure 4.15 shows the experimental results of the total power consumption of the two computing nodes with 8 VMs; and compared to the previous setting with 10 VMs, the power-saving method is indeed more effective and the power-saving rate is doubled to 14%.

Chapter 5

Conclusions and Future Work

For the increasingly high demand of the cloud today, the power demand of the cloud environment issues cannot be ignored. To effectively manage cloud servers, this work presents a power-saving method that, in the case of low usage of the cloud, automatically shuts down several facilities on the cloud to achieve the power efficiency goal.

5.1 Concluding Remarks

After a number of relevant research and experiments, the following conclusions are obtained from analysis of the experimental results.

- There is no direct relationship between migration time and vCPU loading.
- Live migration of virtual machines will not cause significant additional electricity costs.
- The time required for live migration is directly proportional to server CPU loading, but only has a relationship with the source server, and is independent of the target server.
- The power-saving method is effective.

- About 7% to 14% saving of power consumption is achieved.
- The percentage of power saving depends on not only the power-saving method but also the real operation situations of VMs and hosts.

5.2 Future Work

The proposed power-saving method in this work still has many issues and parts need to be explored and reinforced; in order to strengthen the power-saving effect, in the future we plan to:

- Continue to experiment with different settings of the experimental environments and scenarios.
- Reformulate the power-saving method by taking more factors into considerations, such as the disk I/O and network I/O flow that can be easily captured via SNMP.
- Try more diverse computing environments for the power-saving method, such as computing environments using three or more computing nodes, or a combination of hardware specifications of different computing nodes.
- Through more accurate power measurement, understand the additional power cost of operation due to the power-saving method.
- Design a more detailed method such that it can be not only operated in a more diverse environment, but also used to achieve better power-saving effect.

Bibliography

[1] *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, CloudCom 2012, Taipei, Taiwan, December 3-6, 2012.* IEEE, 2012.

[2] K. Adams and O. Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 2–13. ACM, 2006.

[3] K. Alhamazani, R. Ranjan, F. A. Rabhi, L. Wang, and K. Mitra. Cloud monitoring for optimizing the qos of hosted applications. In *CloudCom* [1], pages 765–770.

[4] J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167, 2011.

[5] P. Banerjee, V. Sukthankar, and V. Srinivasan. Method to fairly distribute power saving benefits in a cloud among various customers. In *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*, pages 1–4, 2012.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[7] R.-S. Chang and C.-M. Wu. Green virtual networks for cloud computing. In *Communications and Networking in China (CHINACOM), 2010 5th International ICST Conference on*, pages 1–7, 2010.

[8] D. Chen, D. Lu, M. Tian, S. He, S. Wang, J. Tian, C. Cai, and X. Li. Towards energy-efficient parallel analysis of neural signals. *Cluster Computing*, 16(1): 39–53, 2013.

[9] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[10] A. Corradi, M. Fanelli, and L. Foschini. Increasing cloud power efficiency through consolidation techniques. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 129–134, 2011.

[11] Y. Dong, S. Li, A. Mallick, J. Nakajima, K. Tian, X. Xu, F. Yang, and W. Yu. Extending xen with intel virtualization technology. *Intel Technology Journal*, 10(3):193–203, 2006.

[12] P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok. A survey on open-source cloud computing solutions. In *Brazilian Symposium on Computer Networks and Distributed Systems*, 2010.

[13] S. Figuerola, M. Lemay, V. Reijs, M. Savoie, and B. S. Arnaud. Converged optical network infrastructures in support of future internet and grid services using iaas to reduce ghg emissions. *J. Lightwave Technol.*, 27(12):1941–1946, Jun 2009.

[14] J. G. Hansen and E. Jul. Self-migration of operating systems. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 23. ACM, 2004.

[15] K. Hasebe, T. Niwa, A. Sugiki, and K. Kato. Power-saving in large-scale storage systems with data migration. In *Cloud Computing Technology and*

Science (CloudCom), 2010 IEEE Second International Conference on, pages 266–273, 2010.

[16] Q. Huang, F. Gao, R. Wang, and Z. Qi. Power consumption of virtual machine live migration in clouds. In *Communications and Mobile Computing (CMC), 2011 Third International Conference on*, pages 122–125, 2011.

[17] S. U. Khan, L. Wang, L. T. Yang, and F. Xia. Green computing and communications. *The Journal of Supercomputing*, 63(3):637–638, 2013.

[18] N. Kim, J. Cho, and E. Seo. Energy-based accounting and scheduling of virtual machines in a cloud system. In *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, pages 176–181, 2011.

[19] N. Kim, J. Cho, and E. Seo. Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems*, 2012.

[20] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

[21] KVM. Kernel based virtual machine, 2013. `http://www.linux-kvm.org/page/Main_Page`.

[22] libvirt.org. libvirt, 2013. `http://libvirt.org/`.

[23] C.-C. Lin, P. Liu, and J.-J. Wu. Energy-efficient virtual machine provision algorithms for cloud systems. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 81–88, 2011.

[24] D. S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou. Process migration. *ACM Computing Surveys (CSUR)*, 32(3):241–299, 2000.

[25] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Elastic management of cluster-based services in the cloud. In *Proceedings of the 1st*

workshop on Automated control for datacenters and clouds, ACDC '09, pages 19–24, New York, NY, USA, 2009. ACM.

[26] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009.

[27] H. Raj and K. Schwan. High performance and scalable i/o virtualization via self-virtualized devices. In *Proceedings of the 16th international symposium on High performance distributed computing*, pages 179–188. ACM, 2007.

[28] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.

[29] 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*, pages 417–426. Ieee, 2010.

[30] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *Internet Computing, IEEE*, 13(5):14–22, 2009.

[31] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10. USENIX Association, 2008.

[32] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005.

[33] G. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry. An

overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, 16(1):3–15, 2013.

[34] L. Wang, D. Chen, Y. Hu, Y. Ma, and J. Wang. Towards enabling cyber-infrastructure as a service in clouds. *Computers & Electrical Engineering*, 39(1):3–14, 2013.

[35] L. Wang, D. Chen, J. Zhao, and J. Tao. Resource management of distributed virtual machines. *IJAHUC*, 10(2):96–111, 2012.

[36] L. Wang and S. U. Khan. Review of performance metrics for green data centers: a taxonomy study. *The Journal of Supercomputing*, 63(3):639–656, 2013.

[37] A. Whitaker, R. S. Cox, M. Shaw, and S. D. Gribble. Rethinking the design of virtual machine monitors. *Computer*, 38(5):57–62, 2005.

[38] wikipedia. Cloud computing, 2013. `http://en.wikipedia.org/wiki/Cloud_computing`.

[39] wikipedia. Live migration, 2013. `http://en.wikipedia.org/wiki/Live_migration`.

[40] wikipedia. Protocol data unit, 2013. `https://en.wikipedia.org/wiki/Protocol_data_unit`.

[41] wikipedia. Simple network management protocol, 2013. `http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol`.

[42] wikipedia. Virtualization, 2013. `http://en.wikipedia.org/wiki/Virtualization`.

[43] P. Willmann, J. Shafer, D. Carr, A. Menon, S. Rixner, A. L. Cox, and W. Zwaenepoel. Concurrent direct network access for virtual machine monitors. In *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, pages 306–317. IEEE, 2007.

[44] M. Witkowski, A. Oleksiak, T. Piontek, and J. Węglarz. Practical power consumption estimation for real life {HPC} applications. *Future Generation Computer Systems*, 29(1):208 – 217, 2013. <ce:title>Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures</ce:title>.

[45] Z. Wu, C. Giles, and J. Wang. Classified power capping by network distribution trees for green computing. *Cluster Computing*, 16(1):17–26, 2013.

[46] 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)*, pages 319–324, 2010.

[47] 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)*, pages 319–324, 2010.

[48] M. Yamagiwa and M. Uehara. A study on constructing an energy saving cloud system powered by photovoltaic generation. *2012 15th International Conference on Network-Based Information Systems*, 0:844–848, 2012.

[49] C.-T. Yang, H.-Y. Cheng, and K.-L. Huang. A dynamic resource allocation model for virtual machine management on cloud. In T.-h. Kim, H. Adeli, H.-s. Cho, O. Gervasi, S. Yau, B.-H. Kang, and J. Villalba, editors, *Grid and Distributed Computing*, volume 261 of *Communications in Computer and Information Science*, pages 581–590. Springer Berlin Heidelberg, 2011.

[50] C.-T. Yang, C.-H. Tseng, K.-Y. Chou, and S.-C. Tsaur. A virtualized hpc cluster computing environment on xen with web-based user interface. In *High Performance Computing and Applications*, pages 503–508. Springer, 2010.

[51] X. Zhang and Y. Dong. Optimizing xen vmm based on intel® virtualization technology. In *Internet Computing in Science and Engineering, 2008. ICICSE'08. International Conference on*, pages 367–374. IEEE, 2008.

[52] Y. Zhao and W. Huang. Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 170–175. IEEE, 2009.

[53] H. Zhong, K. Tao, and X. Zhang. An approach to optimized resource scheduling algorithm for open-source cloud systems. In *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*, pages 124–129, 2010.

Appendix A

Inatallation and Setup

I. KVM and Libvirt Install

```
$ sudo apt-get install kvm qemu-kvm bridge-utils
$ sudo apt-get install libvirt-bin virtinst vtun virt-manager
```

II. SNMP Install

```
$ apt-get install snmp snmpd
```

III. PHP Server Install

```
$ sudo apt-get install php5 apache2 mysql-server php5-mysql
```

IV. cpuburn Install

```
$ sudo apt-get install cpuburn
```

Appendix B

Programming Codes

I. Status Monitor

```
<?php
while(date("d")!="18"){
$vm_cpu_sum=0;
$vm_domain_name_arr=array("vm01",.....,"vm10");
$vm_ip_arr=array("172.24.12.221",.....,"172.24.12.230");
$vm_cpu_sum_32core1=0;
$vm_cpu_max_32core1=0;
$conn = libvirt_connect("qemu+ssh://140.128.101.175/system");
$doms = libvirt_list_domains($conn);
$vm_count_on_32core1 = count($doms);
foreach ($doms as $var) {
        $a = snmpwalk($vm_ip_arr[((int)(substr($var,2)))-1], "public",
        "UCD-SNMP-MIB::ssCpuUser");
        foreach ($a as $val) {
                $vm_cpu_sum_32core1 = $vm_cpu_sum_32core1+((int)
                (substr($val, 9))*8);
                if( ((int)(substr($val, 9)))>$vm_cpu_max_32core1 ){
                        $vm_cpu_max_32core1= (int)(substr($val, 9));
                        $vm_cpu_max_domain_32core1 = $var;
                }
        }
}
$vm_cpu_sum_32core2=0;
$conn = libvirt_connect("qemu+ssh://140.128.101.169/system");
$doms = libvirt_list_domains($conn);
$vm_count_on_32core2 = count($doms);
foreach ($doms as $var) {
        $a = snmpwalk($vm_ip_arr[((int)(substr($var,2)))-1], "public",
         "UCD-SNMP-MIB::ssCpuUser");
        foreach ($a as $val) {
                $vm_cpu_sum_32core2 = $vm_cpu_sum_32core2+((int)
                (substr($val, 9))*8);
                        if( ((int)(substr($val, 9)))>$vm_cpu_max_32core2 ){
                        $vm_cpu_max_32core2= (int)(substr($val, 9));
                        $vm_cpu_max_domain_32core2 = $var;
                }
        }
}
$vm_cpu_sum = $vm_cpu_sum_32core1 + $vm_cpu_sum_32core2;
$a = snmpwalk("140.128.101.175", "public", "UCD-SNMP-MIB::ssCpuUser");
        foreach ($a as $val) {
                $cpu_32core1= (int)(substr($val, 9));
        }
$a = snmpwalk("140.128.101.169", "public", "UCD-SNMP-MIB::ssCpuUser");
        foreach ($a as $val) {
                $cpu_32core2= (int)(substr($val, 9));
        }
?>
```

II. Power Consumption Recording

```
<?php
date_default_timezone_set('Asia/Taipei');
while(date("d")!="settimebyuser"){
        if(date("s")=="00"){
                $time = date("Y.m.d/H:i:s");
                $pduget32core2 = snmpget("PDUip", "hpclab", "iso.
                3.6.1.4.1.13742.4.1.2.2.1.7.3");
                $pduget32core1 = snmpget("PDUip", "hpclab", "iso.
                3.6.1.4.1.13742.4.1.2.2.1.7.1");
                $pduget24core = snmpget("PDUip", "hpclab", "iso.
                3.6.1.4.1.13742.4.1.2.2.1.7.2");
                $fp=fopen("pdurecord.txt","a+");
                fputs($fp,$pduinput=($time." ".substr($pduget32core1, 9)."
                 ".substr($pduget24core, 9)."
                 ".substr($pduget32core2, 9)."\n"));
                fclose($fp);
                sleep(5);
        }
}
?>
```

III. Power-saving Method

```
<?php
$hosts_total_cpu_power=6400-(($cpu_32core1*32)+
($cpu_32core2*32)-$vm_cpu_sum);
echo $hosts_total_cpu_power."<br>";
$hosts_total_cpu_power_32core1=3200-(($cpu_32core1*32)-
$vm_cpu_sum_32core1);
$hosts_total_cpu_power_32core2=3200-(($cpu_32core2*32)-
$vm_cpu_sum_32core2);
$max_host_cpu_power = max($hosts_total_cpu_power_32core1,
$hosts_total_cpu_power_32core2);
if($vm_cpu_sum<=$max_host_cpu_power){
        echo ("need only one host<br>");
        if($vm_count_on_32core2 > $vm_count_on_32core1){
                $migrate_source_host = "140.128.101.175";
                $migrate_source_host_ipmi = "140.128.101.164";
                $migrate_tarage_host = "140.128.101.169";
                $migrate_tarage_host_ipmi = "140.128.101.197";
        }
        else if($vm_count_on_32core2 < $vm_count_on_32core1){
                $migrate_source_host = "140.128.101.169";
                $migrate_source_host_ipmi = "140.128.101.197";
                $migrate_tarage_host = "140.128.101.175";
                $migrate_tarage_host_ipmi = "140.128.101.164";
        }
        else if ($vm_count_on_32core2 = $vm_count_on_32core1 ){
                if($vm_cpu_sum_on_32core2 > $vm_cpu_sum_on_32core1){
                        $migrate_source_host = "140.128.101.175";
                        $migrate_source_host_ipmi = "140.128.101.164";
                        $migrate_tarage_host = "140.128.101.169";
                        $migrate_tarage_host_ipmi = "140.128.101.197";
                }
                else if($vm_cpu_sum_on_32core2 <= $vm_cpu_sum_on_32core1){
                        $migrate_source_host = "140.128.101.169";
                        $migrate_source_host_ipmi = "140.128.101.197";
                        $migrate_tarage_host = "140.128.101.175";
                        $migrate_tarage_host_ipmi = "140.128.101.164";
                }
                echo $hosts_total_cpu_power_32core2.
                $hosts_total_cpu_power_32core1;
        }

        $conn = libvirt_connect("qemu+ssh://".$migrate_source_host."/system");
        $doms = libvirt_list_domains($conn);
        foreach ($doms as $var) {
                exec("ssh oneadmin@".$migrate_source_host." virsh migrate
                --live ".$var."
                qemu+ssh://".$migrate_tarage_host."/system");
        }

        $conn = libvirt_connect("qemu+ssh://".$migrate_source_host."/system");
        $doms = libvirt_list_domains($conn);
        if(count($doms)==0){
                exec("ipmitool -H ".$migrate_source_host_ipmi." -U ADMIN -P
                ADMIN power soft");
        }
}
else if($vm_cpu_sum>$max_host_cpu_power &&
$vm_cpu_sum<=$hosts_total_cpu_power){
if($vm_cpu_sum_32core1>$hosts_total_cpu_power_32core1)  {
                if((exec("ipmitool -H 140.128.101.197 -U ADMIN -P
                ADMIN power status"))=="Chassis Power is off"){
                        exec("ipmitool -H 140.128.101.197 -U ADMIN -P
                        ADMIN power on");
                }
                exec("ssh oneadmin@140.128.101.175 virsh migrate --live "
                .$vm_cpu_max_domain_32core1." qemu+ssh://140.128.101.169/system");
        }
        else if($vm_cpu_sum_32core2>$hosts_total_cpu_power_32core2)     {
                exec("ipmitool -H 140.128.101.197 -U ADMIN -P ADMIN power on");
                while((exec("ipmitool -H 140.128.101.197 -U ADMIN -P ADMIN power
                 status"))=="Chassis Power is on"){
                        exec("ssh oneadmin@140.128.101.169 virsh migrate --live "
                        .$vm_cpu_max_domain_32core2." qemu+ssh://140.128.101.175/
                        system");
                }
        }
}
}
?>
```

Appendix C

User Guide

I. VM List with virsh

```
$ sudo virsh list --all
```

II. Start VM with virsh

```
$ sudo virsh start GuestName
```

III. Shutdoen VM with virsh

```
$ sudo virsh destroy GuestName
```

IV. Live Migration with virsh

```
$ virsh migrate --live GuestName DestinationURL
```

V. Retrieve Information with SNMP

```
$ snmpwalk -v 2c -c public localhost system
```

VI. Putting 100% of CPU Usage with cpuburn

```
$ burnP6
```