Future Generation Computer Systems 37 (2014) 26–36

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A method for managing green power of a virtual machine cluster in cloud

Chao-Tung Yang *, Jung-Chun Liu, Kuan-Lung Huang, Fuu-Cheng Jiang

Department of Computer Science, Tunghai University, Taichung 40704, Taiwan

HIGHLIGHTS

- This work proposes a novel method for managing green power of a virtual machine cluster in cloud computing environments.
- A green power management scheme is proposed to determine how many physical machines should be run or turned off based on the gross occupied resource weight ratio of the virtual machine cluster.
- When the gross occupied resource weight ratio is greater than a maximum tolerant occupied resource weight ratio, a standby physical machine in the non-running physical machines is selected and waken up to join as one of the running physical machines.
- A resource allocation process is also used to distribute loads of the running physical machines such that the total number of the running physical machines can be flexibly dispatched to achieve the objective of green power management.

ARTICLE INFO

Article history:
Received 5 July 2013
Received in revised form
3 March 2014
Accepted 8 March 2014
Available online 25 March 2014

Keywords:
Green power management
Resource allocation
Live migration
Virtual machine cluster

ABSTRACT

A green power management scheme is proposed to determine how many physical machines should be run or turned off based on the gross occupied resource weight ratio of the virtual machine cluster. The gross occupied resource weight ratio is defined as the ratio of the sum of resource weights of all virtual machines over the sum of available resource weights of all running physical machines. When the gross occupied resource weight ratio is greater than the maximum tolerant occupied resource weight ratio, preset to ensure quality of service, a standby physical machine in the non-running physical machines is selected and wakened up to join as one of the running physical machines. On the other hand, when the gross occupied resource weight ratio is less than the minimum critical occupied resource weight ratio, preset to trigger energy saving algorithms, one of the running physical machines, selected as a migration physical machine with the virtual machines therein removed after live migration, is moved from other running physical machines, and then turned off. As a result, a resource allocation process is realized to distribute loads of the running physical machines such that the total number of the running physical machines can be flexibly dispatched to achieve the objective of green power management.

1. Introduction

Cloud computing is a concept [1,2], in which computers over a network are able to cooperate with one another to provide far-reaching network services [3–9]. The basic approach of cloud computing is computing through the Internet terminal operations that move workloads from users to the server side to share hardware, software, and information [10–16]; in this way, the previous redundant wastage of resources on individual computers are avoided and the resource efficiency is greatly improved [17,18]. With today's increasingly high demand for cloud, operation of a typical cloud computing such as the large-scale data processing center evolves with a lot of power consumption. Today energy consumption of a large data processing center shares about 0.5% global carbon emissions; with the increase of cloud computing in the future, the estimated proportion will account for 2%, which represents carbon emissions of a large data processing center of some big enterprises, and the estimated proportion for some countries may be beyond [19–21]. Such large amount of power consumption is contrary to today's emphasis on energy conservation and carbon reduction, and it is a major problem that cannot be ignored. How to not only maintain the growth of the cloud computing technology, but also take into account the efficiency of energy use, is the main research aim of this paper [22–24].

The energy demand related issues cannot be ignored in the cloud environment. In order to achieve the purpose of energy saving, unnecessary servers can be turned off through live migration of virtual machines (*VMs*) [25,26]. However, if geared in views of

* Corresponding author. Tel.: +886 4 23590415; fax: +886 4 23590415.
E-mail addresses: ctyang@thu.edu.tw, ctyang.thu@gmail.com (C.-T. Yang), jcliu@thu.edu.tw (J.-C. Liu), admor@thu.edu.tw (K.-L. Huang), peter760504@gmail.com (F.-C. Jiang).

http://dx.doi.org/10.1016/j.future.2014.03.001
0167-739X/

the virtual machine only, the quality of service is probably reduced, contrary to the intent of the cloud computing. In the commercial cloud computing environments, emphasis is on service level agreements. An effective management tool for services and service level expectations can help the service provider and the client, and it can help enterprises to establish channels of communication with the communication plan to establish consistency, reduce conflict, and reach the objective of measuring service performance.

In this study, we propose a green power management scheme to efficiently supervise and allocate computing resources based on the gross occupied resource weight ratio of the virtual machine cluster, in which the gross occupied resource weight ratio, i.e., θ_{Load} defined later in Eq. (1) in Section 3, is the ratio of the sum of resource weights of all virtual machines over the sum of available resource weights of all running physical machines. When θ_{Load} is greater than the maximum tolerant occupied resource weight ratio, i.e., λ, preset to ensure quality of service, a standby physical machine in the non-running physical machines is selected and awakened to join the running physical machines. Then again, when θ_{Load} is less than the minimum critical occupied resource weight ratio, i.e., β, preset to activate energy saving algorithms, one of the running physical machines, selected as a migration physical machine with the virtual machines therein removed after live migration, is moved from other running physical machines, and then turned off. Hence, the green power management scheme can control the load rate of the virtual machine cluster in a range set by the user to avoid too high or too low loads.

Our solution will be helpful for the service level agreements; through the measurement of the CPU and memory usage, the quality of service is ensured as a precondition. Furthermore, through energy saving algorithms of the cloud virtual machine management system, live migration of virtual machines is conducted for energy saving. Finally, experimental results show that the proposed resource allocation algorithm under normal usage scenarios can indeed achieve a certain degree of energy saving effect.

2. Background review and related work

2.1. Cloud computing

Cloud computing is an Internet-based computing; in this way, the shared hardware and software resources and messages can be provided on demand to computers and other devices. The cloud is a metaphor for the network, or the Internet. The users do not need to know the details of the "cloud" infrastructure or have the in-depth expertise, and are without direct control. Cloud computing allows companies to deploy applications more quickly, and reduces the complexity of management and maintenance costs to rapidly reallocate IT resources in response to business needs [10,27]. Cloud computing describes new Internet-based services to increase IT use and easily deliver models to provide dynamic and often a virtual extension of the resource. Cloud computing can be considered as including following levels of service: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

2.2. Hardware-assisted virtualization

Computer operating systems provide different levels of access to resources. On most operating systems, Ring 0 is the level with the most privileges and interacts most directly with the physical hardware such as the CPU and memory. Hardware-assisted virtualization [28–30] is used to overcome the problem that the *VM* operating system kernel cannot be placed in the Ring 0 privilege level of the processor. Since the hypervisor, which creates and runs virtual machines, and virtual operating system kernel can be issued in Ring 0, the hypervisor will automatically intercept the instruction dealing with the virtual operating system directly with the hardware processing. The full virtualization Binary Translation or para-virtualization Hypercall operation is no longer needed.

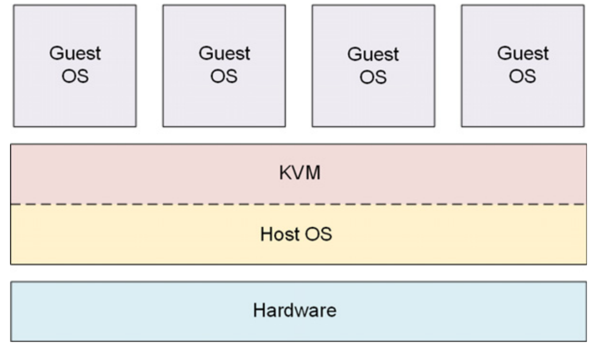


Fig. 1. Hardware-assisted virtualization.

Hardware-assisted virtualization basically eliminates the difference between full virtualization and para-virtualization. Based on the hardware-assisted full virtualization or para-virtualization program virtual machines are high performance and independent. Representative program of the hardware-assisted virtualization is *VM*ware ESXi with open source Kernel-based Virtual Machine (KVM) [31]. KVM needs a host operating system, and the core of the system provides virtualization services. *VM*ware ESXi is the equivalent of a specialized virtualization thin client operating system, installed directly on a physical machine. The architecture of the hardware-assisted virtualization is illustrated in Fig. 1.

2.3. Xen

Xen [32,33], an open source host virtualization technology, can divide a host into multiple (Linux, Windows, and other OS) hosts. Xen released from the Computer Laboratory of the University of Cambridge, UK, which established the XenoServer Project research and development, was created for wide area distributed computing. Licensed by the GNU General Public License (GNU GPL), a variety of open source programs and related technologies of Xen continue to develop. On the other hand, Xen's research team responsible for the establishment of the XenSource released Xen-Based Enterprise Edition solutions in 2005. In late 2006, XenSource released XenEnterprise 3.0, which was meant to directly compete with *VM*ware.

Xen uses hyper-hypervisor type Virtual Machine Monitor (VMM) architecture, which is efficient and has secure control of CPU, memory, and other resources of the host. In general the host virtualization software is divided as the Host OS type and hyper-hypervisor type [11,34]. The virtualization layer of the Host OS type is installed above Windows, Linux and other OS; above the virtualization layer other OS called Guest OS is installed. The hyper-hypervisor is installed directly above the host machine, and other OS is installed on top of it. Xen divides the resources needed by the Host OS, offers better performance and easier management of CPU, memory, network, storage and other resources. The Xen virtualization architecture is shown in Fig. 2.

2.4. VM management

To set up a standard cloud service, we need a virtual machine. When a large number of virtual machines are created through virtualization technology, it becomes very cumbersome to manage them with native instructions; hence a virtual machine management platform is needed [12,35]. The virtual machine management platform includes a virtual machine to create, edit, switch, pause, reply, delete, and may perform live migration operations. Next,

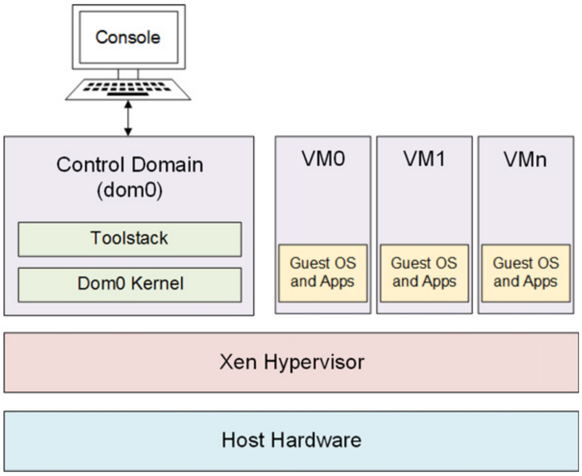


Fig. 2. Xen architecture.

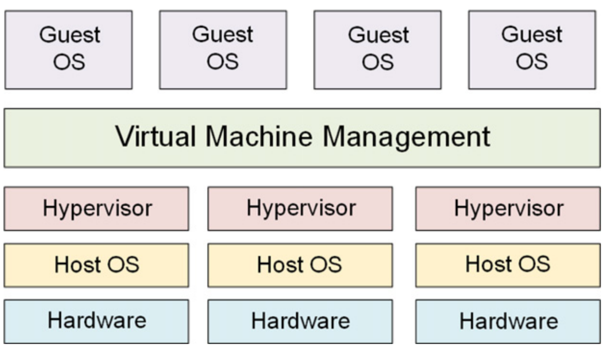


Fig. 3. *VM* management.

some popular open source virtualization management platforms can be used as the network interface to provide a virtual building process with many advantages, for example, a more friendly and suitable interface is provided for monitoring states of a large number of virtual machines, and the account permissions are also easier to manage. The VMM management architecture is shown in Fig. 3.

2.5. Live migration

Live migration refers to the process of moving a running virtual machine or application between different physical machines without disconnecting the client or application [18]. The memory states, 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 called pre-copy memory migration and post-copy memory migration. The architecture of live migration of *VM*s is shown in Fig. 4.

2.6. Related work

Recently, the research field of green and low power consumption networking infrastructure has been vigorous for service/network providers and equipment manufacturers as well, such as energy aware routing algorithms for virtual routers [36], schemes to estimate energy consumption and used as billing bases for cloud systems [37], analysis of energy consumption in cloud computing [27], automated resource scheduling [38], power budgeting designs to manage the power consumption of web services [39], and consolidation of applications in cloud environments [40].

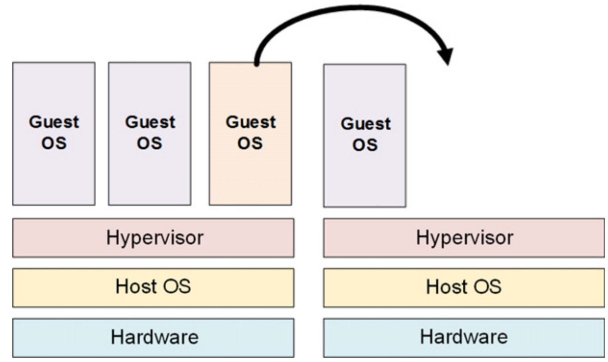


Fig. 4. Live migration of *VM*.

The emerging cloud computing technology can increase the utilization and efficiency of hardware equipments; hence it can potentially decrease the global CO_2 emission. Chang et al. proposed a virtual network architecture for cloud computing [36], in which the virtual network can provide communication functions for virtual resources in cloud computing. In order to build a green virtual network in cloud computing, they designed an energy aware routing algorithm for virtual routers, and an efficient method for setting up the virtual network. The energy consumed by the network component is estimated to decide the packet forwarding route. They demonstrated that by the proposed algorithm the energy consumption of the communication can be minimized in theory; however, more research needs to be done to make the algorithm more efficient and to measure the actual performances.

Kim et al. [37] suggested a model without dedicated measurement hardware to estimate the energy consumption of a virtual machine based on in-processor events generated by the virtual machine. Based on this estimation model, a virtual machine scheduling algorithm is proposed to provide computing resources according to the energy plan of each virtual machine. The suggested scheme was implemented in the Xen virtualization system, and the evaluation shows that the suggested scheme estimates energy consumption and accordingly provides computing resources with errors of less than 5% of the total energy consumption. The suggested scheme can be used as a billing basis for cloud systems; however, since only processor energy consumption was considered in the model, diverse components other than processors, including storage devices and network interface cards, should be included to build more accurate models for energy-based billing systems.

Baliga et al. presented an analysis of energy consumption in cloud computing [27]. Their analysis considered both the public and private clouds, and included energy consumption in switching and transmission as well as data processing and data storage. They showed that more energy-efficient use of computing power can be obtained in cloud computing, especially for computing tasks of low intensity or infrequent nature. Nevertheless, power consumption in transport stands for a major proportion of total power consumption for cloud storage services at medium and high usage rates. Similarly, for cloud software services, power consumption in transport is negligibly small at very low screen refresh rates, but at middle and high screen refresh rates, power consumption in transport becomes significant and energy savings over PCs are reduced.

Resource scheduling is a key process for IaaS clouds. Zhong et al. investigated the possibility to flexibly allocate *VM*s to permit the maximum usage of physical resources [38]. For the automated scheduling policy, they adopted an Improved Genetic Algorithm (IGA), which uses the shortest genes and exploits the idea of Dividend Policy in Economics to choose an optimal or suboptimal allocation for *VM*s requests. The simulation experiments showed that the speed of the IGA is almost twice the traditional GA scheduling method in the grid environment, and the resources utilization rate of IGA is always higher than that of scheduling strategies such as First fit, Round robin algorithms, and average queuing and configurable scheduling used in open-source IaaS cloud systems.

To better manage the power consumption of web services in cloud computing with dynamic user locations and behaviors, Wu et al. proposed a power budgeting design based on the logical level, using a distribution tree [39]. By setting multiple trees, they differentiated and analyzed the influence of workload types and Service Level Agreements (SLAs, e.g. the response time) in terms of power properties. Based on these, they introduced classified power capping for different services as the control reference to maximize power saving for mixed workloads situations. Simulation shows that classified power capping sets the power budget of mixed network groups closer to actual power needs based on performance requirements, and thus, power can be saved.

Consolidation of applications in cloud computing environments offers an important prospect for energy optimization. To investigate possibility of energy efficient consolidation, Srikantaiah et al. experimentally studied the interrelationships between energy usage, resource utilization, and performance of consolidated workloads [40]. The consolidation problem was modeled as a modified bin packing problem, and the challenges in finding effective solutions to the consolidation problem were also outlined. The study indicates the energy performance trade-offs for consolidation and shows that optimal operating points exist. The paper focuses only on manageable and important factors, i.e., CPU and disk resources. To achieve a real world implementation of energy efficient consolidation, many other issues affecting consolidation should be considered, including server and workload behavior, security restrictions, and power line redundancy restrictions.

3. The proposed method

Accordingly, the objective of this study is to provide a method for managing green power of a virtual machine cluster by controlling load ratios. In order to achieve our aim, we used a virtual machine cluster consisting of a plurality of physical machines such as servo hosts. The total number of the physical machines is represented by P, of which there are a number of running physical machines with one or more virtual machines. The total number of the running physical machines is represented by p, and the remaining physical machines represented by (P-p) are off and in the standby state.

The Green Power Management scheme (GPM) comprises following steps:

Step 1. Calculate the gross occupied resource weight ratio of the virtual machine cluster, which is the ratio of the sum of resource weights of all virtual machines over the sum of available resource weights of the p running physical machines. The gross occupied resource weight ratio, θ_{Load} is calculated with Eq. (1):

$$\theta_{Load} = \frac{\sum_{i=1}^{n}(VM_{jiCPUuse} \times VM_{jiRAMallocate})}{\sum_{j=1}^{p}(PM_{jCPU} \times PM_{jRAM})}, \quad (1)$$

where j is the serial number of the respective physical machine; i, the serial number of the respective virtual machine; p, the total number of the running physical machines of the virtual machine cluster; n, the total number of the virtual machines; $VM_{jiCPUuse}$, the processor load rate of virtual machine i in physical machine j; $VM_{jiRAMallocate}$, the memory allocation of virtual machine i in physical machine j; PM_{jCPU}, the processor resource in physical machine j; and PM_{jRAM}, the memory resource in physical machine j.

Step 2. When the gross occupied resource weight ratio is greater than the maximum tolerant occupied resource weight ratio λ set by the user to ensure quality of service, and $p < P$, select and wake up a standby physical machine in non-running physical machines to join the other running physical machines, i.e., $p = p + 1$. The selected standby physical machine is selected from non-running physical machines to obtain a gross occupied resource weight ratio closest to $\frac{(\lambda+\beta)}{2}$ after running it.

Step 3. When the gross occupied resource weight ratio is less than the minimum critical occupied resource weight ratio β set by the user to trigger energy saving algorithms, and $p > 1$, select one of the running physical machines with the least load (or the least loaded virtual machine) as the migration physical machine, move the virtual machines of the migration physical machine to other running physical machines, and shut off the migration physical machine.

Step 4. Execute a resource allocation process to evenly distribute loads of the running physical machines.

As a dynamic resource allocation process for evenly distributing loads of the running physical machines, the resource allocation process further comprises following steps:

Step 4.1. Calculate the following three weights: the virtual machine occupying resource weight of the respective virtual machine, i.e., VM_{jiRate} defined later by Eq. (2); the physical machine occupying resource weight of the respective physical machine, i.e., $HOST_{jRate}$ defined by Eq. (3); and the average physical machine occupying resource weight of all the physical machines, i.e., α defined by Eq. (4).

Step 4.2. When the difference between the physical machine occupying resource weight and the average physical machine occupying resource weight is greater than a default migration value, execute following steps:

1: while $((HOST_{jRate} - \alpha)! = 0)$ **do**
2: elect a PM with a $max\ VM_{jiRate}$ as the PM_{max};
3: elect a PM with a $min\ VM_{jiRate}$ as the PM_{min};
4: calculate $(HOST_{jRate} - \alpha)$;
5: if there is a VM in the PM_{max}
with $min\left(\left|VM_{jiRate} - (HOST_{jRate} - \alpha)\right|\right)$ **then**
6: set the VM as $VM_{migration}$;
7: migration $(VM_{migration} \rightarrow PM_{max})$;
8: end if
9: end while

The parameters are defined as follows:

- PM: the physical machine
- VM: the virtual machine
- $VM_{jiCPUuse}$: occupying CPU resource weight of VM_{ji}
- $VM_{jiRAMallocate}$: occupying RAM resource weight of VM_{ji}.

The following equations are used in calculation:

$$VM_{jiRate} = \frac{(VM_{jiCPUuse} \times VM_{jiRAMallocate})}{\sum_{i=1}^{n}(VM_{jiCPUuse} \times VM_{jiRAMallocate})} \quad (2)$$

$$HOST_{jRate} = \sum_{i=1}^{v} VM_{jiRate} \quad (3)$$

$$\alpha = \frac{1}{P}, \quad (4)$$

where v represents the total number of the virtual machines in the respective physical machine; VM_{jiRate}, the virtual machine occupying resource rate of virtual machine i in physical machine j to act as the virtual machine occupying resource weight in Step 4.1, being

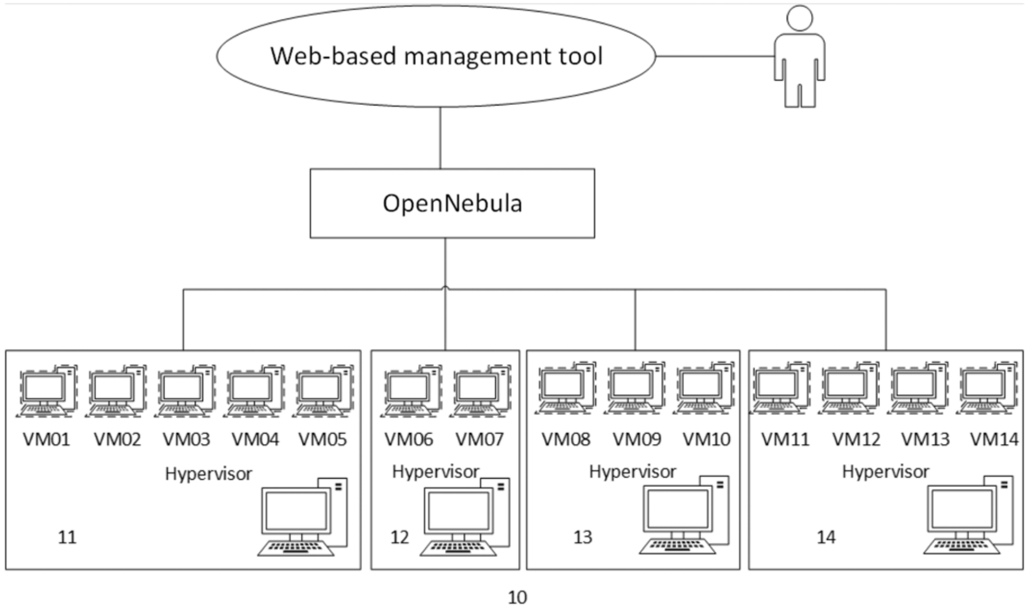


Fig. 5. The system framework of the method for managing green power of a virtual machine cluster.

defined as the rate of the occupying resource in virtual machine i of physical machine j divided by the occupying resource in the whole virtual machine cluster; $HOST_{jRate}$, the physical machine occupying resource rate of physical machine j to act as the physical machine occupying resource weight in Step 4.1, being defined as the sum of the virtual machine occupying resource rates in physical machine j; α, an average physical machine occupying resource rate of the physical machines to act as the average physical machine occupying resource weight in Step 4.1.

From above, the proposed method for managing green power accordingly is capable of controlling the load rate of the virtual machine cluster in a range set by the user to avoid too high or too low loads. In addition, virtual machines in physical machines with high load rates can be migrated to physical machines with low load rates to conserve total energy.

4. Description of the method and examples

The system structure, applied principles, functions, and the effectiveness of the proposed method are further illustrated in this section.

As shown in Fig. 5, a web-based management tool is used for managing green power in virtual machine cluster 10 consisting of P physical machines such as servo hosts, in which there are p physical machines: 1, 2, . . . , 11, 12, 13, 14, . . . , and p in the running state (only four running physical machines 11, 12, 13, 14 are explicitly shown to represent the p running physical machines), and P-p physical machines being in the off and standby state. Each of the running physical machines 11, 12, 13, 14 executes Xen Hypervisor software to simulate one or more virtual machines *VM*xx. Beside, live migration of the virtual machines *VM*xx is operated and managed using OpenNebula software.

In Step A1 of Fig. 6, the gross occupied resource weight ratio of the virtual machine cluster is computed. The gross occupied resource weight ratio is referred to the ratio of sum of occupied resource weights of all the virtual machines *VM*xx divided by the sum of resource weights of all the running physical machines 11, 12, 13, 14 shown in Fig. 6.

Next, in Step A2 we determine if the total load of running physical machines 11, 12, 13, 14 is excessively high. When the gross occupied resource weight ratio of these running physical machines 11, 12, 13, 14 is greater than a maximum tolerant occupied resource weight ratio λ set by the user, the total load is excessively high; and at the moment, if there are physical machines still not running, that is, a condition $p < P$ is true, one of the standby physical machines is selected to join the running physical machines in Step A3, i.e., the number of the running physical machines is increased to $p + 1$ from p. And afterward, it enters Step A8 to move some of the virtual machines *VM*xx associated with the running physical machines 11, 12, 13, 14 into the newly added physical machine in order to evenly allocate loads on physical machines again.

In Step A2, if the load of the physical machines 11, 12, 13, 14 is not excessively high, it enters Step A4 to determine if the load is excessively low. When the gross occupied resource weight ratio corresponding to running physical machines 11, 12, 13, 14 is lower than a minimum critical occupied resource weight ratio β set by the user, the load is excessively low; at the moment, if there are two or more running physical machines, that is, a condition $p > 1$ is true, one of the running physical machines 11, 12, 13, 14 is selected as a migration physical machine in Step A5, and virtual machines *VM*xx in the selected migration physical machine are moved to the other running physical machines; then Step A6 is executed to check if the migration is finished, and Step A7 is executed to shut off the selected migration physical machine as a standby physical machine when the migration is done; in other words, the number of the running physical machines is decreased from p to $p - 1$; furthermore, it enters Step A8 to evenly allocate resources for loads on the running physical machines.

In Step A1, the gross occupied resource weight ratio θ_{Load} is calculated as the ratio of the sum of occupied processors times allocated memories of the virtual machines over the sum of that of physical machines 11, 12, 13, 14 with Eq. (1) in Section 3.

In Step A3, one standby physical machine is selected to join running physical machines and the gross occupied resource weight ratio is calculated again. It is a principle that the recalculated gross occupied resource weight ratio should be closest to $\frac{(\lambda+\beta)}{2}$; in other words, the physical machines of the virtual machine cluster are capable of running in a state of better load condition after the standby physical machine joins as one of the running physical machines.

In Step A5, when selecting the migration physical machine, it is a principle that among the running physical machines 11, 12, 13,

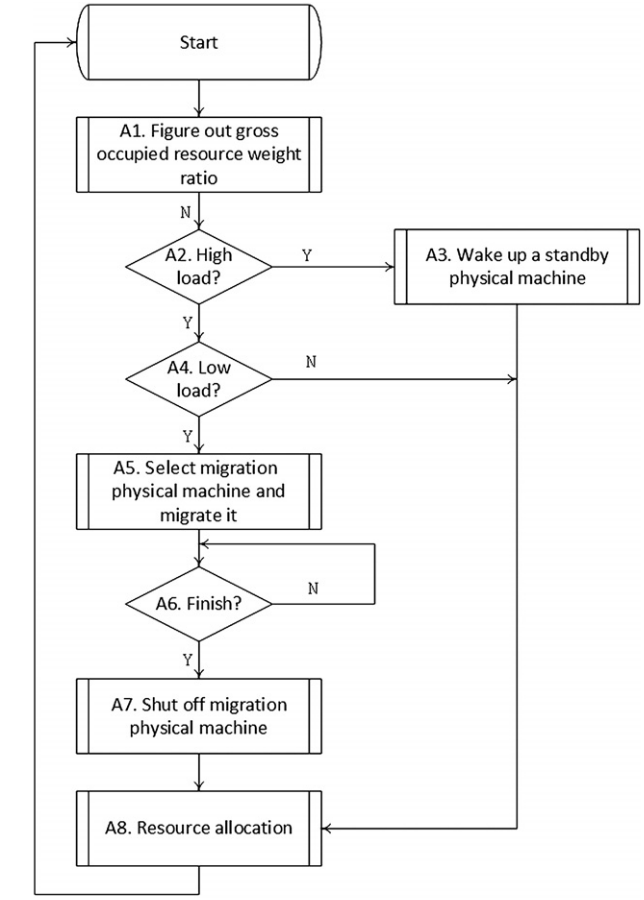


Fig. 6. Flow chart of the method for managing green power of a virtual machine cluster.

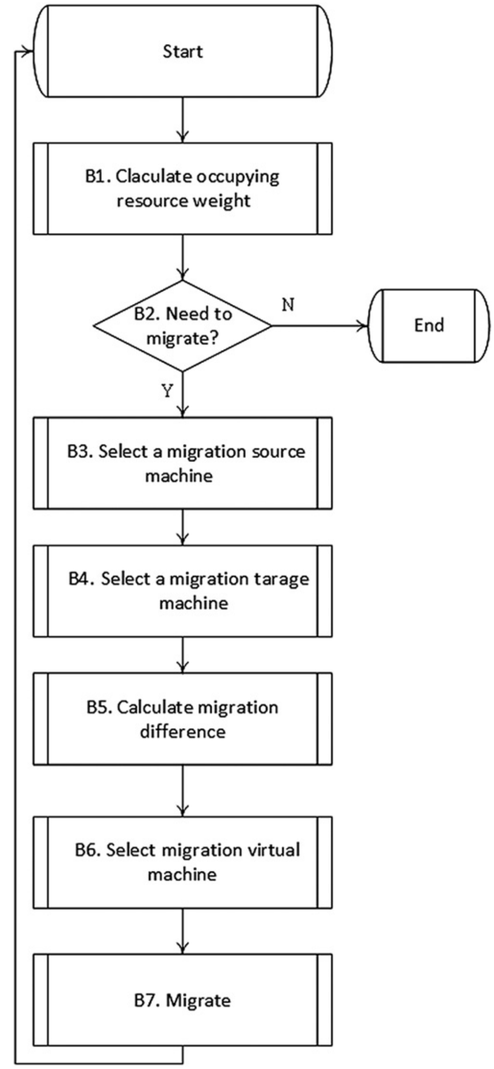


Fig. 7. The flow chart of the resource allocation process.

14, the one with the least load or virtual machines is selected to facilitate the process of migration.

The flow chart shown in Fig. 7 further explains how to evenly allocate loads of the running physical machines 11, 12, 13, 14 by a dynamic resource allocation process.

Step B1 used Eqs. (2)–(4) in Section 3 to calculate the virtual machine occupying resource weight of each of the virtual machines *VM*xx, the physical machine occupying resource weight of each of the physical machines, and average physical machine occupying resource weight of all the physical machines, respectively.

In Eqs. (1)–(4), although the load rate $VM_{jiCPUuse}$ and the memory allocation $VM_{jiRAMallocate}$ of the respective virtual machine in each of the physical machines 11, 12, 13, 14 are calculated in percentage to obtain the gross occupied resource weight ratio θ_{Load}, the virtual machine occupying resource rate VM_{jiRate}, the physical machine occupying resource rate $HOST_{jRate}$, and the average physical machine occupying resource rate α, respectively, a person skillful in the art should know that other resources in the physical machines 11, 12, 13, 14 such as storage devices can be taken into account, or the weight values can be calculated with different equations.

After finding the virtual machine occupying resource weight of each of the virtual machines *VM*xx, the physical machine occupying resource weight of each of the physical machines 11, 12, 13, 14, and the average physical machine occupying resource weight of all physical machines 11, 12, 13, 14, Step B2 determines whether to do live migration of virtual machines. When the difference between the physical machine occupying resource weight of any one of the physical machines 11, 12, 13, 14 and the average physical machine occupying resource weight is greater than the default migration value σ set by the user, Steps B3–B7 are conducted to proceed migration.

In Step B3, a physical machine with the maximum physical machine occupying resource weight is elected as a migration source machine. Then, in Step B4, a physical machine with the minimum physical machine occupying resource weight is elected as a migration target machine. Further, in Step B5, a migration difference between the physical machine occupying resource weight of the migration source machine and the average physical machine occupying resource weight is calculated. Furthermore, in Step B6, a virtual machine with the virtual machine occupying resource weight thereof closest to the migration difference is elected as a migration virtual machine. Finally, in Step B7, the migration virtual machine is moved to the migration target machine to complete a resource allocation cycle and enter another resource allocation cycle if necessary.

It is supposed that each of the physical machines 11, 12, 13, 14 in Fig. 5 is a host with an 8-core processor and an 8192 KB memory such that the available processor load rate of each of the physical machines 11, 12, 13, 14 is $8 \times 100 = 800$. Table 1 lists resource weights before migration, including the processor load rate $VM_{jiCPUuse}$, the memory allocation $VM_{jiRAMallocate}$ of the virtual machines *VM*xx in each of the physical machines 11, 12, 13, 14, the

Table 1
VM occupying resource weight before migration.

PM	*VM*	$VM_{jiCPUuse}$	$VM_{jiRAMallocate}$	VM_{jiRate}	$HOST_{jRate}$
11	*VM*01	95	512	0.08	0.45
	*VM*02	100	1024	0.17	
	*VM*03	40	2048	0.14	
	*VM*04	10	512	0.01	
	*VM*05	30	1024	0.05	
12	*VM*06	70	1024	0.12	0.17
	*VM*07	60	512	0.05	
13	*VM*08	10	1024	0.02	0.05
	*VM*09	15	512	0.01	
	*VM*10	20	512	0.02	
14	*VM*11	45	1024	0.08	0.33
	*VM*12	60	512	0.05	
	*VM*13	30	512	0.03	
	*VM*14	100	1024	0.17	

Table 2
VM occupying resource weight after migration.

PM	*VM*	$VM_{jiCPUuse}$	$VM_{jiRAMallocate}$	VM_{jiRate}	$HOST_{jRate}$
11	*VM*01	95	512	0.08	0.28
	*VM*03	40	2048	0.14	
	*VM*04	10	512	0.01	
	*VM*05	30	1024	0.05	
12	*VM*06	70	1024	0.12	0.17
	*VM*07	60	512	0.05	
13	*VM*02	100	1024	0.17	0.22
	*VM*08	10	1024	0.02	
	*VM*09	15	512	0.01	
	*VM*10	20	512	0.02	
14	*VM*11	45	1024	0.08	0.33
	*VM*12	60	512	0.05	
	*VM*13	30	512	0.03	
	*VM*14	100	1024	0.17	

virtual machine occupying resource weight VM_{jiRate}, and the physical machine occupying resource weight $HOST_{jRate}$. The unit of the memory allocation $VM_{jiRAMallocate}$ is in Kbytes.

It is supposed that the default migration value $\sigma = 0.05$, and by Eq. (4) the average physical machine occupying resource rate $\alpha = \frac{1}{4} = 0.25$; thus, the difference $\left(HOST_{jRate} - \alpha\right)$ for the respective physical machine is found to be 0.20, −0.08, −0.20, and 0.08. Hence some of the difference values are greater than the default migration value σ. As a result, migration has to be performed. Physical machine 11, which has the greatest physical machine occupying resource ratio 0.45, can be used as the migration source machine; whereas physical machine 13, which has the least physical machine occupying resource ratio 0.05, can be used as the migration target machine. In addition, the virtual machine occupying resource ratio VM_{jiRate} of virtual machine *VM*02 is 0.17, which is closest to the migration difference 0.20 of physical machine 11 such that virtual machine *VM*02 is used as the migration virtual machine and is migrated to physical machine 13. The results after migration are listed in Table 2.

After the second migration, the difference $\left(HOST_{jRate} - \alpha\right)$ is calculated again for each of the physical machines and is found to be 0.03, −0.08, −0.03, and 0.08, respectively. There are still two difference values greater than the default migration value σ. It is found that physical machine 14, which has the greatest physical machine occupying resource ratio 0.33, can be used as the migration source machine, and physical machine 12, which has the smallest physical machine occupying resource ratio 0.17, can be used as the migration target machine. In addition, the virtual machine occupying resource ratio VM_{jiRate} of virtual machine *VM*11 is 0.08, which is closest to the migration difference 0.08 of physical machine 14 such that virtual machine *VM*11 is used as the migration virtual machine and is migrated to physical machine 12. The difference $\left(HOST_{jRate} - \alpha\right)$ for each of the physical machines is again found to be 0.03, 0, −0.03, and 0, respectively. Apparently, the effect of load balance is substantively achieved.

While the proposed system has been illustrated with referencing to the preferred implementation thereof, it is to be understood that modifications or variations may be easily made without departing from the spirit of this system.

5. System implementation

5.1. System architecture

Besides managing individual life cycles of *VM*s, we also designed the core to deploy services typically including a set of interrelated components (for example, a web server and database back end) requiring several *VM*s. Thus, we could treat a group of related *VM*s as a first-class entity in OpenNebula. In addition to managing *VM*s as a unit, the core also handles the context information delivery, such as the Web server's IP address, digital certificates, and software licenses to *VM*s [25]. Fig. 8 shows the perspective of system architecture, in which a cluster system is built using OpenNebula. We also used a web interface to manage virtual and physical machines. The cluster system consisted of four homogeneous computers, each with following specifications: Intel i7 CPU with a 2.8 GHz clock rate, 4 GB memory, 500 GB disk, installed with the Debian operating system, and a GB switch to connect to the network.

5.2. Management interface

We designed a useful web interface for end users with the goal to provide users with a fast and friendly way to implement virtualization environments. Fig. 9 shows the authorization mechanism. Through the core of the web-based management tool, one can easily control and manage both physical machines and life cycles of *VM*s.

The entire web-based management tool includes physical machine management, virtual machine management, and performance monitoring. As shown in Fig. 10, one can set attributes of *VM*s, such as the memory size, IP address, root password, and name of *VM*s. It includes the live migration function as well. Live migration means *VM*s can be moved to any working physical machine without suspending in-service programs. Live migration is one of the advantages of OpenNebula. Therefore, we can perform migration on any *VM* under any situation according to the GPM mechanism described in the previous sections to perform meaningful migrations.

RRDtool, the open source industry standard high performance data logging and graphing system, is used for time series data. RRDtool can be used to write customized monitoring shell scripts or create whole applications using its Perl, Python, Ruby, TCL, or PHP bindings [41]. In this paper we used RRDtool to monitor the entire system. Fig. 11(a) and (b) show available CPUs and memory usage of current physical machines, respectively.

Fig. 12 is a GMP setting page. It shows information such as which hosts are currently controlled by OpenNebula, which hosts have enabled the GPM mechanism, and host states of GPM. Moreover once the host enables GPM, the system will automatically control the *VM* on the host and start the loading balance mechanism.

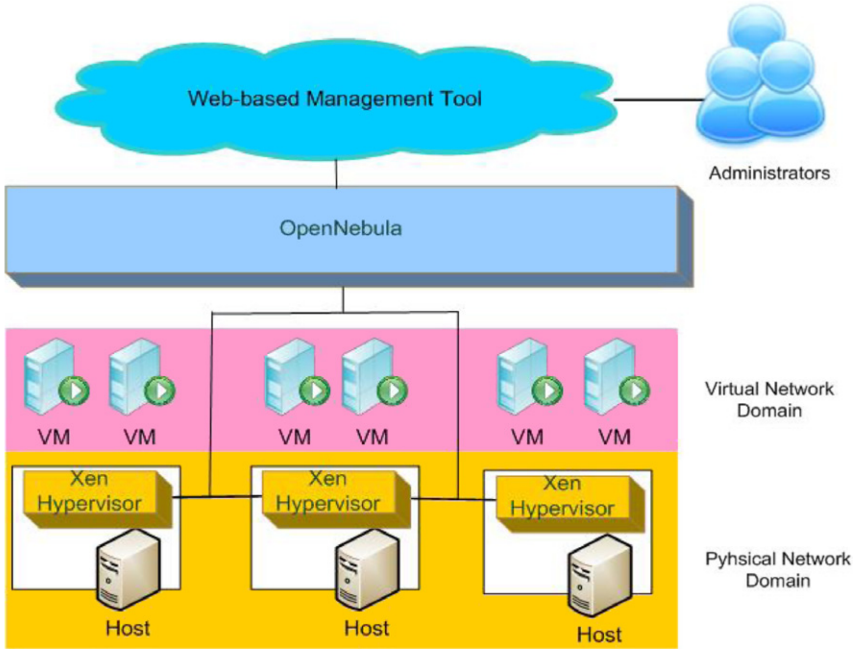


Fig. 8. System architecture used in the experiment.

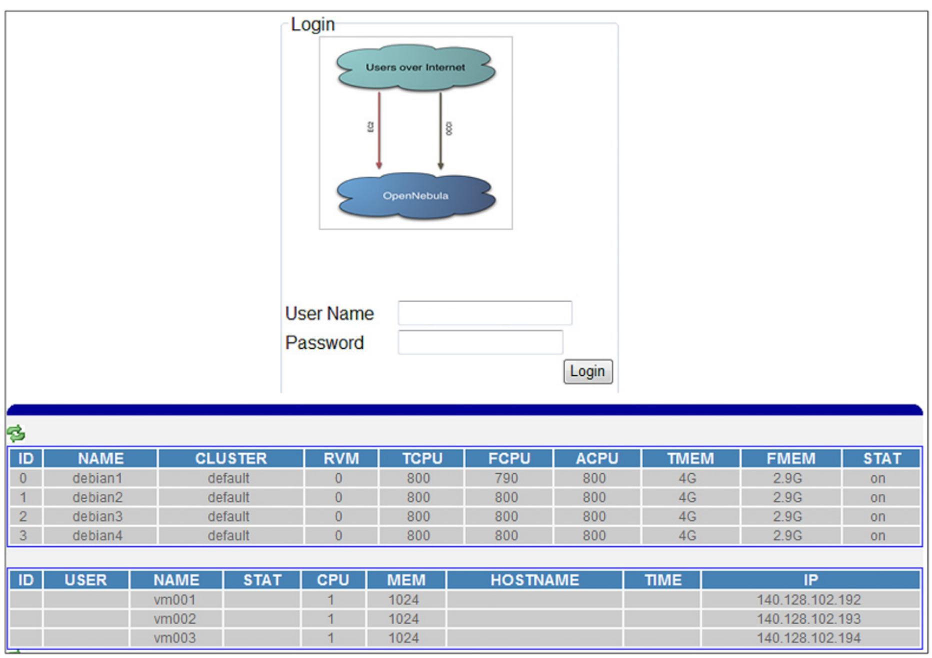


Fig. 9. Web-based interface.

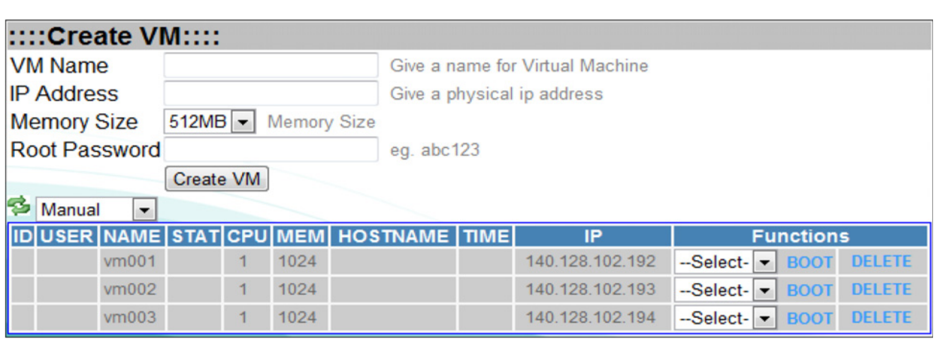
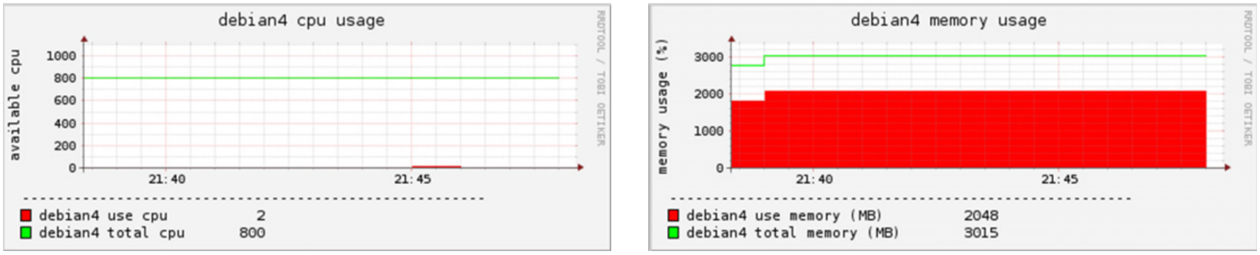


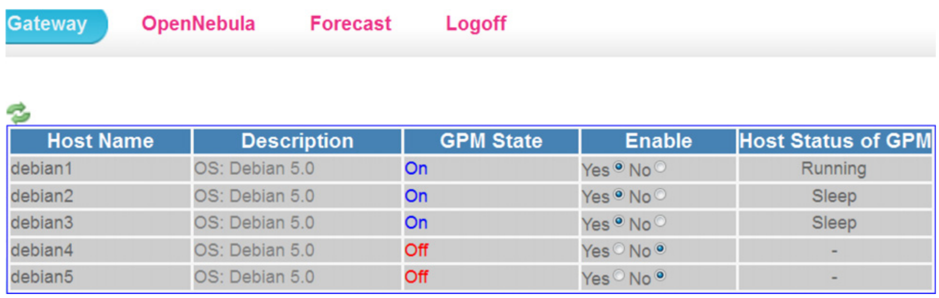
Fig. 10. The GUI for virtual machine management.



(a) Map of available CPUs.

(b) Map of memory usage.

Fig. 11. Maps of available CPUs and memory usage for current physical machines.



Host Name	Description	GPM State	Enable	Host Status of GPM
debian1	OS: Debian 5.0	On	Yes ● No ○	Running
debian2	OS: Debian 5.0	On	Yes ● No ○	Sleep
debian3	OS: Debian 5.0	On	Yes ● No ○	Sleep
debian4	OS: Debian 5.0	Off	Yes ○ No ●	-
debian5	OS: Debian 5.0	Off	Yes ○ No ●	-

Fig. 12. GPM setting page.

6. Experimental results

First of all, we focused on resource utilization of computing of the proposed method model. Therefore, we used High Performance Computing Challenge (HPCC) software to verify that method enhances performance and has efficient resource utilization in the virtualization cluster [20,21,34,42]. The HPC Challenge Benchmark from HPCC consists of a set of benchmarks targeting to test multiple attributes that can contribute substantially to the real-world performance of HPC systems [43].

We used three physical machines in our first experimental environment. We created six virtual machines and distributed them on three different host machines. Each virtual machine had one virtual CPU with a 512 MB virtual memory. The virtual machine with high workloads on HOST 1 was migrated to a physical machine with lower resource load when the method function was enabled. Figs. 13 and 14 show the experimental results, in which the red curve represents results with method function disabled; and the blue curve, with method function enabled.

Fig. 13 shows trends of the HPCC computing time with various HPCC problem sizes. We notice that as the HPCC problem size increases, the difference of HPCC computing time with or without the method function becomes obvious. In this experiment, we ran HPCC programs in six virtual machines and aggregated HPCC performance on these six virtual machines. An abrupt increase of CPU usage in the virtual machines cluster would somehow affect CPU usage of host machines. When the method function disabled, virtual machines located on the same host machine processed HPCC computing simultaneously and competed physical resources with one another. When the method function enabled, it would know that resources usage on each host machine was balanced or not; therefore, virtual machines on some host machines would be automatically migrated by the method function to others.

Fig. 14 also shows the effectiveness of the method function. The vertical axis represents performance of floating point operations on virtual machines. Better performance was found with the method function enabled. It also proves that our approach is effective under this circumstance. In Fig. 14, when the problem size is small, it shows better performance when virtual machines centralized on the same host rather than on distributed hosts. Because in

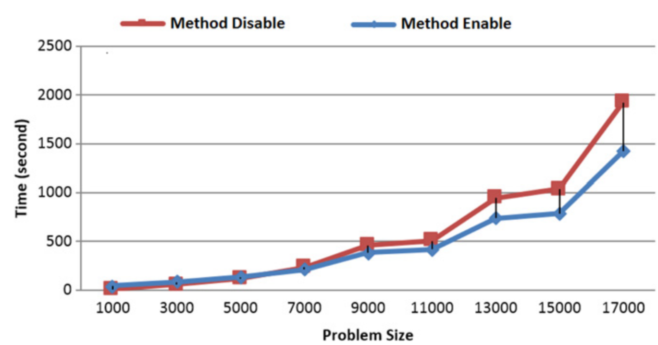


Fig. 13. Execution time for running HPCC on *VMs*.

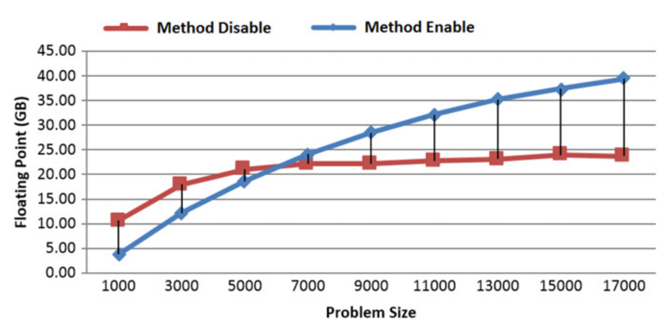


Fig. 14. Performance of Floating Point per second on *VMs*.

order to do HPCC performance computing on the virtual machines cluster, computing data were transferred to each virtual machine, which in turn delivered message to one another by the virtual switch of host. However, we observe that when the problem size reaches about 6000, the HPCC performance with the method function enabled, i.e. virtual machines distributed to different hosts, is better than that with the method function disabled. It is due to the fact that when the problem size becomes too big, the virtual machines cluster on the single host can no longer effortlessly handle the computation.

Furthermore, we built application servers to offer services, including computing services, the teaching website, and multi-media

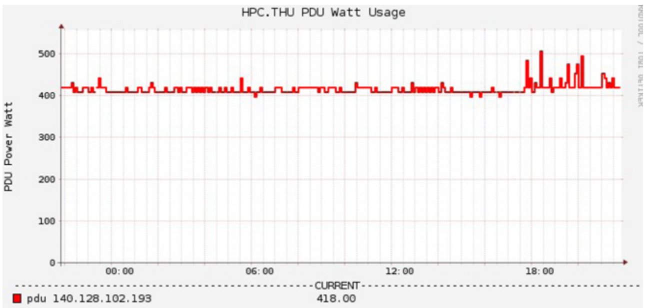


Fig. 15. Power monitoring information.

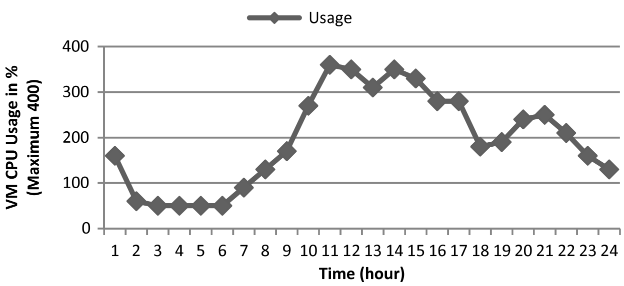


Fig. 16. CPU usage of *VM*s.

services for compressing and decompressing media files in the virtual environment. All the services were implemented on four physical machines set on a power distribution unit (PDU). The PDU is a device with multiple appliance outlets designed to distribute the electric power. We continuously monitored instant power consumption on the four physical machines. We used RRDtool [28] to plot power consumption charts as shown in Fig. 15. We observed that the power consumption was about 400 W or more. The four physical machines as OpenNebula clients ran a total of four *VM*s. Each *VM* provided an application service. Fig. 16 shows the average total power usage per hour in a 24 h period of the four *VM* CPUs with data recorded within one month. Here we used the Simple Network Management Protocol (SNMP) to record the *VM* CPU usage per hour, then found that between 2 and 7 a.m., utilization of CPU was low; and from 10 a.m. to 4 p.m., relatively high. Consequently, *VM*s need more physical resources in the 10 a.m.–4 p.m. interval.

With the same time period as that in Fig. 15, Fig. 17 shows average total power consumption per hour in the 24 h period of the system with GPM enabled or disabled. The unit in the X-axis is hour for time, and the Y-axis, watt for the total power consumption of the four physical machines. It illustrates that when GPM was turned off, four machines operated all the time and the power consumption was over 400 W. But as we observe in Fig. 17, from 2 a.m. to 7 a.m., the total CPU demand of *VM*s was relatively small. The decision-making function based on the GPM front-end would migrate *VM*s to one physical machine and shut down the other physical machines to save energy. On the other hand, from 10 a.m. to 4 p.m., GPM was aware that the CPU demand of *VM*s exceeded the level that can be handled by any single physical machine. For that reason, the front-end automatically woke up other machines by the Wake on LAN (WOL) technology with the load balance approach. We conclude that the system is able to turn on or shut down physical machines according to the computing demand to effectively achieve the aim of energy saving.

7. Conclusions

In this work, we present a dynamic resource allocation method on virtualization platforms for green power management, which

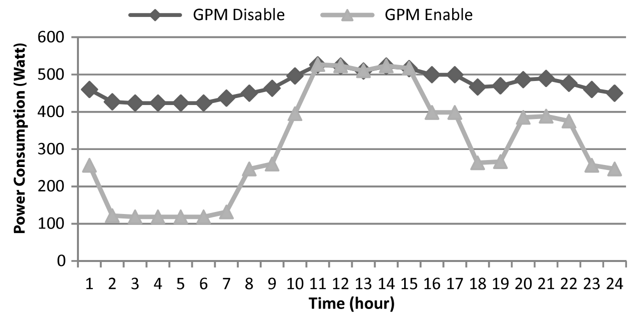


Fig. 17. Power consumption of the system with or without GPM.

allows flexible supervision of resources in cloud computing to conserve energy. Our research work includes (1) support of the GPM mechanism, (2) implementation of resource monitoring with an OpenNebula web-based interface, and (3) advantageous features based on GPM and OpenNebula instead of traditionally scheduled booting of physical machines. Moreover, we expect to improve violent CPU heavy loading events, because our goal is to have smooth rather than dramatic changes when using virtual machines. For instance, some sensitivity parameters can be preset for the entire mechanism to function well. Finally, compared to traditional approaches, the proposed GPM approach certainly reaches the goal of significant energy saving. However, in this study, since only CPU usage and memory allocation are considered in the equations, other factors such as disk spaces and communication bandwidth might be used to construct a more actual model. This constitutes our future work.

Acknowledgments

This work is sponsored by Tunghai University the U-Care ICT Integration Platform for the Elderly, No. 102GREEnS004-2, Aug. 2013. This work was supported in part by the National Science Council Taiwan, under grant numbers NSC 101-2218-E-029-004 and NSC 102-2218-E-029-002.

References

[1] R. Moreno-Vozmediano, Ruben S. Montero, Ignacio 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, ACM, Spain, 2009, pp. 19–24.
[2] Patrícia Takako Endo, Glauco Estácio Gonçalves, Judith Kelner, Djamel Sadok, A survey on open-source cloud computing solutions, in: Proceedings of VIII Workshop em Clouds, Grids e Aplicações, 2010, pp. 3–16.
[3] W. Hagen, Professional Xen Virtualization, Wrox Press Ltd, Birmingham, UK, 2008.
[4] Chao-Tung Yang, Chien-Hsiang Tseng, Keng-Yi Chou, Shyh-Chang Tsaur, A virtualized HPC cluster computing environment on Xen with web-based user interface, in: The second International Conference on High Performance Computing and Applications, HPCA 2009, in: Lecture Notes in Computer Science, vol. 5938, Springer, Shanghai, China, 2009, pp. 503–508.
[5] Borja Sotomayor, Ruben S. Montero, Ignacio M. Llorente, Ian Foster, Virtual infrastructure management in private and hybrid clouds, IEEE Internet Comput. 13 (5) (2009) 14–20.
[6] Jia-Liang Yen, Ming-Jeng Yang, Chao-Tung Yang, Power-saving management for energy-efficient green clouds, J. Internet Tech. (2014) in press.
[7] Chao-Tung Yang, Jung-Chun Liu, Rajiv Ranjan, Wen-Chung Shih, Chih-Hao Lin, On construction of heuristic QoS bandwidth management in clouds, Concurrency Comput. Pract. Exp. 25 (18) (2013) 2540–2560.
[8] Chao-Tung Yang, Jung-Chun Liu, Hsien-Yi Wang, Ching-Hsien Hsu, The implementation of GPU virtualization using PCI pass-through mechanism, J. Supercomput. (2014) in press.
[9] Chao-Tung Yang, Chi-Jui Liao, Jung-Chun Liu, Walter Den, Ying-Chyi Chou, Jaw-Ji Tsai, Construction and application of an intelligent air quality monitoring system for healthcare environment, J. Med. Syst. 38 (2014) 15.
[10] Chao-Tung Yang, Jung-Chun Liu, Ching-Hsien Hsu, Wei-Li Chou, On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism, J. Supercomput. (2014) in press.

[11] Xiantao Zhang, Yaozu Dong, Optimizing Xen VMM based on intel virtualization technology, in: 2008 International Conference on Internet Computing in Science and Engineering, ICICSE 2008, 2008, pp. 367–374.
[12] E.S. Jae-Wan Jang, Heeseung Jo, Jin-Soo Kim, A low-overhead networking mechanism for virtualized high-performance computing systems, J. Supercomput. (2010).
[13] Lizhe Wang, Dan Chen, Yangyang Hu, Yan Ma, Jian Wang, Towards enabling cyberinfrastructure as a service in clouds, Comput. Electr. Eng. 39 (1) (2013) 3–14.
[14] Lizhe Wang, Dan Chen, Jiaqi Zhao, Jie Tao, Resource management of distributed virtual machines, IJAHUC 10 (2) (2012) 96–111.
[15] Lizhe Wang, Marcel Kunze, Jie Tao, Gregor von Laszewski, Towards building a cloud for scientific applications, Adv. Eng. Softw. 42 (9) (2011) 714–722.
[16] Lizhe Wang, Dan Chen, Fang Huang, Virtual workflow system for distributed collaborative scientific applications on grids, Comput. Electr. Eng. 37 (3) (2011) 300–310.
[17] S. Figuerola, et al., Converged optical network infrastructures in support of future internet and grid services using IaaS to reduce GHG emissions, J. Lightwave Technol. 27 (2009) 1941–1946.
[18] Qiang Huang, Fengqian Gao, Rui Wang, Zhengwei Qi, Power consumption of virtual machine live migration in clouds, in: 2011 Third International Conference on Communications and Mobile Computing, CMC, 2011, pp. 122–125.
[19] Ruben S. Montero, et al., An elasticity model for high throughput computing clusters, J. Parallel Distrib. Comput. 71 (6) (2011) 750–757.
[20] W. Emeneker, D. Stanzione, HPC cluster readiness of Xen and user mode linux, in: Proceedings of 2006 IEEE International Conference on Cluster Computing, 2006, pp. 1–8.
[21] Dave Turner, Xuehua Chen, Protocol-dependent message-passing performance on linux clusters, in: Proceedomgs of Cluster 2002 Conference, Chicago, September 25, 2002, pp. 187–194.
[22] Z. Hai, et al. An approach to optimized resource scheduling algorithm for open-source cloud systems, in: 2010 Fifth Annual ChinaGrid Conference, ChinaGrid, 2010, pp. 124–129.
[23] Lizhe Wang, Gregor von Laszewski, Marcel Kunze, Jie Tao, Jai Dayalet, Provide virtual distributed environments for grid computing on demand, Adv. Eng. Softw. 41 (2) (2010) 213–219.
[24] M. Witkowskia, A. Oleksiaka, T. Pionteka, J. Węglarza, Practical power consumption estimation for real life HPC applications, Future Gener. Comput. Syst. 29 (1) (2013) 208–217.
[25] S. Soltesz, H. Potzl, M.E. Fiuczynski, A. Bavier, L. Peterson, Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors, in: EuroSys '07, Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pp. 275–287.
[26] H. Raj, 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, HPDC 2007, 2007, pp. 179–188.
[27] Jayant Baliga, Robert W.A. Ayre, Kerry Hinton, Rodney S. Tucker, Green cloud computing: balancing energy in processing, storage, and transport, Proc. IEEE 99 (2011) 149–167.
[28] K. Adams, O. Agesen, A comparison of software and hardware techniques for x86 virtualization, in: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM Press, New York, NY,USA, 2006, pp. 2–13.
[29] C. Huang, G. Zheng, S. Kumar, L.V. Kalé, Performance evaluation of adaptive MPI, in: Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming 2006, March 2006, pp. 12–21.
[30] F. Wong, R. Martin, R. Arpaci Dusseau, D. Culler, Architectural requirements and scalability of the NAS parallel benchmarks, in: Proceedings of the 1999 ACM/IEEE Conference on Supercomputing (CDROM), ACM Press, New York, NY, USA, 1999, p. 41.
[31] http://www.linux-kvm.org/ last accessed: March 2, 2014.
[32] Yaozu Dong, Shaofan Li, Asit Mallick, Jun Nakajima, Kun Tian, Xuefei Xu, Fred Yang, Wilfred Yu, Extending Xen with Intel virtualization technology, Intel Technol. J. 10 (3) (2006) 193–204.
[33] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the Art of Virtualization, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles, ACM Press, New York, USA, 2003, pp. 164–177.
[34] Arun Babu Nagarajan, Frank Mueller, Christian Engelmann, Stephen L. Scott, Proactive fault tolerance for HPC with Xen virtualization, in: Proceedings of the 21st Annual International Conference on Supercomputing, Seattle, Washington, June 17–21, 2007.
[35] J.S. Paul Willmann, David Carr, Aravind Menon, Scott Rixner, Alan L. Cox, Willy Zwaenepoel, Concurrent direct network access for virtual machine monitors, in: Proceedings of the Second International Conference on High Performance Computing and Applications, HPCA, 2007, pp. 306–317.
[36] Ruay-Shiung Chang, Chia-Ming Wu, Green virtual networks for cloud computing, in: 2010 5th International ICST Conference on Communications and Networking in China, CHINACOM, 2010, pp. 1–7.
[37] Nakku Kim, Jungwook Cho, Euiseong Seo, Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems, Future Gener. Comput. Syst. 32 (2014) 128–137.
[38] Hai Zhong, Kun Tao, Xuejie Zhang, An approach to optimized resource scheduling algorithm for open-source cloud systems, in: Fifth Annual China Grid Conference, 2010.
[39] Z. Wu, 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 and International Conference on Cyber, Physical and Social Computing (CPSCom), 2010, pp. 319–324.
[40] S. Srikantaiah, et al. Energy aware consolidation for cloud computing, in: Proceedings of the 2008 Conference on Power Aware Computing and Systems, San Diego, California, 2008.
[41] http://oss.oetiker.ch/rrdtool, last accessed: March 2, 2014.
[42] Hitoshi Oi, Fumio Nakajima, Performance analysis of large receive offload in a Xen virtualized system, in: Proceedings of 2009 International Conference on Computer Engineering and Technology, ICCET 2009, vol. 1, Singapore, January 2009, pp. 475–480.
[43] Ching-Chi Lin, Pangfeng Liu, Jan-Jan Wu, Energy-efficient virtual machine provision algorithms for cloud systems, in: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, UCC, 2011, pp. 81–88.

Chao-Tung Yang. He is a Professor of Computer Science at Tunghai University in Taiwan. He received the Ph.D. in Computer Science from National Chiao Tung University in July 1996. In August 2001, he joined the Faculty of the Department of Computer Science at Tunghai University. He is serving in a number of journal editorial boards, including *International Journal of Communication Systems*, *Journal of Cloud Computing,* "Grid Computing, Applications and Technology" Special Issue of *Journal of Supercomputing*, and "Grid and Cloud Computing" Special Issue of *International Journal of Ad Hoc and Ubiquitous Computing*. Dr. Yang has published more than 200 papers in journals, book chapters and conference proceedings. His present research interests are in cloud computing and service, grid computing, parallel computing, and multicore programming. He is a member of the IEEE Computer Society and ACM.

Jung-Chun Liu. He received the B.S. degree in electrical engineering from National Taiwan University in 1990. He received M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering at the University of Texas at Austin, in 1996 and 2004, respectively. He is currently an assistant professor in the Department of Computer Science at the Tunghai University, Taiwan. His research interests include cloud computing, embedded systems, wireless networking, artificial intelligence, digital signal processing, and VLSI design.

Kuan-Lung Huang is now working at IISI Company, he was a member of HPC (high performance computing) Lab of computer science at Tunghai University in Taichung, Taiwan. He received the M.S. in computer science from Tunghai University in July 2013. Huang has published more than 10 papers in book chapters and conference proceedings. His present research interests are in cloud and grid computing and parallel computing.

Fuu-Cheng Jiang. He worked as a design engineer with the Aeronautical Research Lab., Chung Shan Institute of Science and Technology (CSIST) when he was assigned to a partnership project at General Dynamic, Fort Worth, Texas. Currently, he is a member of faculty in the department of computer science at Tunghai University in Taiwan. Dr. Jiang was the recipient of the Best Paper Award at the 5th International Conference on Future Information Technology 2010 (FutureTech 2010), which ranked his paper first among the 201 submittals. He has served the TPC of the ICCCT 2011–2012 International Conference, BWCCA2010, IEEE CloudCom 2012 and CSE2011 Session as Chair. Moreover, he served as journal reviewer of the Computer Journal, Ad hoc Networks, Journal of Supercomputing, Journal of Internet Technology and the International Journal of Communication Systems. His research interests include network modeling, cloud computing, wireless networks and simulation. Dr. Jiang is a member of IEEE society.