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 wakened 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.

© 2014 Elsevier B.V. All rights reserved.

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
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., $\theta$ 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 $\theta_{\text{load}}$ is greater than the maximum tolerant occupied resource weight ratio, i.e., $\lambda$, preset to ensure quality of service, a standby physical machine in the non-running physical machines is selected and awakes to join the running physical machines. Then again, when $\theta_{\text{load}}$ is less than the minimum critical occupied resource weight ratio, i.e., $\beta$, 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.

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 VMware 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. VMware 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 XenServer 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 VMware.

Xen uses hyper-vhypervisor 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 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 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,
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 VMs 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].
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 server 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, \( \theta_{\text{load}} \) is calculated with Eq. (1):

\[
\theta_{\text{load}} = \frac{\sum_{i=1}^{n}(VM_{jCPU\text{Use}} \times VM_{jRAM\text{Allocate}})}{\sum_{j=1}^{p}(PM_{jCPU} \times PM_{jRAM})}, \tag{1}
\]

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; \( VM_{jCPU\text{Use}} \), the processor load rate of virtual machine \( i \) in physical machine \( j \); \( VM_{jRAM\text{Allocate}} \), 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 resource weight ratio \( \lambda \) 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 \( \frac{1+\lambda}{2} \) after running it.

Step 3. When the gross occupied resource weight ratio is less than the minimum critical occupied resource weight ratio \( \beta \) 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., \( VM_{j\text{allocate}} \) defined later by Eq. (2); the physical machine occupying resource weight of the respective physical machine, i.e., \( HOST_{j\text{rate}} \) defined by Eq. (3); and the average physical machine occupying resource weight of all the physical machines, i.e., \( \alpha \) 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 \(((HOST_{j\text{rate}} - \alpha)! = 0)\) do
2: elect a PM with a max \( VM_{j\text{allocate}} \), as the \( PM_{\text{max}} \);
3: elect a PM with a min \( VM_{j\text{allocate}} \), as the \( PM_{\text{min}} \);
4: calculate \((HOST_{j\text{rate}} - \alpha)\);
5: if there is a VM in the \( PM_{\text{max}} \) with \( \min (\{VM_{j\text{allocate}} - (HOST_{j\text{rate}} - \alpha)\}) \) then
6: set the VM as \( VM_{\text{migration}} \);
7: migration \( VM_{\text{migration}} \rightarrow PM_{\text{max}} \);
8: end if
9: end while

The parameters are defined as follows:

- \( PM \): the physical machine
- \( VM \): the virtual machine
- \( VM_{jCPU\text{Use}} \): occupying CPU resource weight of \( VM_{ij} \)
- \( VM_{jRAM\text{Allocate}} \): occupying RAM resource weight of \( VM_{ji} \).

The following equations are used in calculation:

\[
VM_{j\text{allocate}} = \frac{(VM_{jCPU\text{Use}} \times VM_{jRAM\text{Allocate}})}{\sum_{i=1}^{n}(VM_{jCPU\text{Use}} \times VM_{jRAM\text{Allocate}})} \tag{2}
\]

\[
HOST_{j\text{rate}} = \frac{\sum_{i=1}^{v} VM_{j\text{allocate}}}{v} \tag{3}
\]

\[
\alpha = \frac{1}{p} \tag{4}
\]

where \( v \) represents the total number of the virtual machines in the respective physical machine; \( VM_{j\text{allocate}} \), 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
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 \( j \); \( \alpha \), 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: \( 1, 2, \ldots, 11, 12, 13, 14, \ldots \), 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 VMxx. Besides, live migration of the virtual machines VMxx 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 VMxx 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 \( \lambda \) 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 VMxx 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 \( \beta \) 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 VMxx 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 \( \theta_{load} \) 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,
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 VMxx, 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 \( \theta_{\text{load}} \), the virtual machine occupying resource rate VM_jiRate, the physical machine occupying resource rate HOST_jiRate, and the average physical machine occupying resource rate \( \alpha \), 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 VMxx, 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 \( \sigma \) 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 VMxx in each of the physical machines 11, 12, 13, 14, the
virtual machine occupying resource weight $VM_{jRate}$, and the physical machine occupying resource weight $HOST_{jRate}$. The unit of the memory allocation $VM_{jRAMallocate}$ 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 $(HOST_{jRate} - \alpha)$ 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 $\sigma$. 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 $VM_{jRate}$ of virtual machine VM02 is 0.17, which is closest to the migration difference 0.20 of physical machine 11 such that virtual machine VM02 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 $(HOST_{jRate} - \alpha)$ 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 $\sigma$. It is found that physical machine 14, which has the greatest physical machine occupying resource ratio 0.33, can be used as the migration target 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 $VM_{jRate}$ of virtual machine VM11 is 0.08, which is closest to the migration difference 0.08 of physical machine 14 such that virtual machine VM11 is used as the migration virtual machine and is migrated to physical machine 12.

The difference $(HOST_{jRate} - \alpha)$ 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 VMs, 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 VMs. Thus, we could treat a group of related VMs as a first-class entity in OpenNebula. In addition to managing VMs as a unit, the core also handles the context information delivery, such as the Web server’s IP address, digital certificates, and software licenses to VMs [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 VMs.

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 VMs, such as the memory size, IP address, root password, and name of VMs. It includes the live migration function as well. Live migration means VMs 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.
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 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
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 VMs that the power consumption was about 400 W or more. The four physical machines as OpenNebula clients ran a total of four VMs. 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, VMs 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 VMs was relatively small. The decision-making function based on the GPM front-end would migrate VMs 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 VMs 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 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 rebooting 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
