Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 328536, 7 pages
http://dx.doi.org/10.1155/2014/328536

Research Article

An Automatic Decision-Making Mechanism for Virtual Machine Live Migration in Private Clouds

Ming-Tsung Kao,[1] Yu-Hsin Cheng,[2] and Shang-Juh Kao[1]

[1] *Department of Computer Science and Engineering, National Chung-Hsing University, 250 Kuo-Kuang Rd., 40227 Taichung City, Taiwan*
[2] *Department of Information Networking and System Administration, 1 Ling Tung Road, Taichung, Taiwan*

Correspondence should be addressed to Shang-Juh Kao; sjkao@cs.nchu.edu.tw

Received 12 June 2014; Accepted 27 July 2014; Published 31 August 2014

Academic Editor: Teen-Hang Meen

Due to the increasing number of computer hosts deployed in an enterprise, automatic management of electronic applications is inevitable. To provide diverse services, there will be increases in procurement, maintenance, and electricity costs. Virtualization technology is getting popular in cloud computing environment, which enables the efficient use of computing resources and reduces the operating cost. In this paper, we present an automatic mechanism to consolidate virtual servers and shut down the idle physical machines during the off-peak hours, while activating more machines at peak times. Through the monitoring of system resources, heavy system loads can be evenly distributed over physical machines to achieve load balancing. By integrating the feature of load balancing with virtual machine live migration, we successfully develop an automatic private cloud management system. Experimental results demonstrate that, during the off-peak hours, we can save power consumption of about 69 W by consolidating the idle virtual servers. And the load balancing implementation has shown that two machines with 80% and 40% CPU loads can be uniformly balanced to 60% each. And, through the use of preallocated virtual machine images, the proposed mechanism can be easily applied to a large amount of physical machines.

1. Introduction

Cloud computing [1] is booming in the internet world. In addition to public and education clouds [2], enterprises also start deploying private clouds [3]. A cloud system can decrease the cost of physical machines and power consumption and can increase the utilization rate of physical machines. By adopting the private cloud, an enterprise is able to install operating systems of different types and versions, allowing a variety of services and applications to be provided. Prior to the use of cloud systems, achieving this goal may require many physical machines together with a large amount of human power. Instead, enterprises can use virtual machines [4] to provide the same functionalities while decreasing the use of physical machines and power consumption.

Several studies [5–7] have shown that virtual machine migration can effectively enable dynamic management for physical machine resources. And CPU loading is an effective factor in determining the energy consumption [8]. This paper presents a study to further reduce power consumption by migrating virtual machines, during the off-peak period, and consolidating physical machines. We propose a decision-making mechanism for directing virtual machine migration and server consolidation. By automatically managing resources of physical machines, virtual machines may quickly obtain sufficient resources to provide services.

2. Related Works

To enable automatic VM migration, we have investigated private cloud building related technologies and used KVM [9]/QEMU [10] virtualization technology. In addition to the use of Libvirt [11] as a virtual machine control panel and GlusterFS [12] as a file system for disk images, we adopted the MongoDB [13] database system and analysed energy consumption (watt) by ZigBee [14] digital power meter.

All are implemented using PHP with the operating system environment of Ubuntu 12.04 LTS. Related technologies used in this study are described hereunder.

2.1. KVM/QEMU. Cloud computing is booming in the current information technology industry, and virtualization plays an important role. Therefore, competition in this area is very intense. Currently, VMware, Xen, KVM/QEMU, VirtualBox, UML, and Hyper-V technologies are fairly popular in the virtualization world. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). KVM can switch CPU to guest model. The operating system can operate without modifying through CPU, MMU, and IO virtualization technologies. KVM can be operated by loading Linux kernel module, taking its efficiency close to the speed of a physical CPU. Scholar Younge et al. [15] have pointed out that KVM (Kernel-Based Virtual Machine) is superior to other virtualization technologies in analysis and evaluation of high-performance computing (HPC) environments. In this study, we also choose KVM/QEMU for virtual machine development.

QEMU is a software which can be used to simulate many devices, such as disk, network card, and USB. It can also be used to simulate many CPU systems, such as PowerPC, Space, MIPS, and ARM. All computer devices which are simulated by QEMU can provide virtual machines with better performance as integrating with KVM virtualization system.

KVM/QEMU can be used to operate on various virtual machines, which are installed with different operating systems such as Linux or Windows. Each VM has its own virtual hardware, such as network cards, disk drives, and graphic cards. KVM/QEMU allows sharing the hardware resources efficiently over the hosted physical machine. In addition, each VM operates independently, which may avoid suffering from virus intrusion to the entire service.

2.2. Libvirt. Libvirt mechanism can be used to report detailed implementation states of each virtual machine, such as CPU time, memory, and network traffic. It also supports the features of virtual machines migration and database information. Libvirt is an open source API for virtual platform management. It can support various hypervisors, such as KVM/QEMU, Xen, User Mode Linux, OpenVZ, and VirtualBox. C, C++ C#, Java, OCaml, Perl, PHP, Python, and Ruby are all supported by Libvirt, allowing programmers to manage various hypervisors and develop different cloud management systems efficiently.

2.3. Live Migration. To transfer the CPU load across physical machines, migration of memory contents is the most important task for real-time live migration [16]. Virtual machine migration consists of the following steps: premigration, reservation, iterative precopy, stop and copy, and the commitment activation. Precopy is the most commonly used virtual machine migration technique. With precopy all memory pages are migrated in preparation stage; once the writable work sequence (WWS) is small enough or the predefined iteration threshold has been reached, the target virtual machine will be brought down and moved. All CPU states and memory pages are transferred to the destination machine. Although precopy can compromise the system downtime due to migration, it cannot guarantee transferring data seamlessly if the work queue limit is reached, especially when the virtual machine is running read-intensive operations [17, 18].

Virtual machines can be migrated from one physical machine to another in two ways. One is office migration, which requires shutting down the to-be-migrated virtual machines and stopping their services temporally. Another is live migration, allowing virtual machines to be migrated while maintaining the services. As we focus on the automatic live migration, the migration time should be kept as short as possible. Only CPU states, device states, and memory are migrated. We have prestored the disk image of the virtual machines in an online file system instead of installing it during the migration time.

As reported by Hirofuchi et al. [19], there are two ways of live migration: precopy migration and postcopy migration, which differ in the timing of memory copying. Precopy copies memory to the target virtual machine until dirty memory pages fall below a threshold. Then, the CPU states, device states, and remaining memory pages are finally moved and the VM starts to operate.

Postcopy firstly copies CPU states and device states and resumes its operation, followed by copying memory. At the beginning, there will be no memory pages in the destination machine and the VM starts to function only when all memory images are completely copied. Even though postcopy could be superior to precopy, as reported in the paper by Hirofuchi et al. [19], in this paper, we present a simple, successful KVM-based live migration.

2.4. GlusterFS. As of VM operating management, we found that it is relatively easy to recover from the faults of CPU, memory, network, and display adapter, while the damage of disk image usually causes a big trouble to bring up the system. To enable disk images to be fault-tolerant, GlusterFS [12] is the online file system for storage management.

GlusterFS is an open source distributed file system with petabyte-level scale-out storage and is able to manage thousands of clients. It uses client/server framework and combines all blocks through TCP/IP or InfiniBand RDMS technology to provide better protection and performance. GlusterFS can also collect disk, memory resources, and management information at a global namespace. Using GlusterFS, all client operations can be carried out from applications using standard IP networks.

2.5. MongoDB. To automatically manage a cloud cluster, a database system is required to collect the information of system operating status and internet traffic from both physical and virtual machines. Since the volume of collected data is huge, the database system needs to support big data storage and efficient data manipulation. In this study, we choose MongoDB [13] database system for data management

purpose. MongoDB is an open base system, which is easy for development and is capable of manipulating massive data.

MongoDB belongs to document-based database systems, mainly for the storage of unstructured documents. It can support drivers and client libraries of various program languages, such as C, C++, C#, Java, PHP, Python, and Ruby. As opposed to traditional database systems, MongoDB is a type of NoSQL databases. NoSOL stands for "Not Only SQL," which was invented as a next-generation database to solve several major problems, such as nonrelation, distribution, open source, and horizontal scalability. At present, all database systems based on demand of high traffic social network services, indexing massive information, or streaming media are urgently requested without a uniform standardization.

2.6. ZigBee Digital Power Meter. This study has used ZigBee [14] digital power meter to monitor the energy consumption of the hosts. It has advantages of wireless data transmission, convenience for setting, low energy consumption, longer transmission distance, and data centralization. These advantages allow us to construct and maintain power control. ZigBee is a newly developed wireless technology, designed to provide a short distance and low bandwidth solution to data transmission of automatic control. It is based on medium access control layer and physical layer by IEEE 802.15.4 and is capable of handling a wireless transmission network of up to 65,000 devices. The ZigBee Alliance (http://www.zigbee.org) now leads the industry. The distance between each device ranges from standard 75 m to several hundred meters or even several kilo meters. A ZigBee network contains three types of nodes as shown in Figure 1. One of them is the coordinator, which is mainly responsible for network establishment and network address assignment. One ZigBee network can have only one coordinator. Another one is router, which is responsible for building and recovering the router path of data packets. It also needs to transform data packets and assign network address to its child devices. All others are end devices, which can join an existing ZigBee network and transform/receive data packets.

3. Design of Automated Decision-Making Mechanism

3.1. System Architecture. The architecture of the implementation system is shown in Figure 2. The control node acts as the coordinator for collecting information from both physical and virtual machines. The control node is equipped with MongoDB and executes the decision-making mechanism. A physical machine is a tangible machine which provides resources to virtual machines. Virtual machines may also run different operation systems to provide the various services. In the architecture, GlusterFS cluster provides disk space to virtual machines. In addition ZigBee digital power meter is adopted to monitor the usage of electricity. One digital power meter needs to be installed in each physical machine, so that the control node can collect the power usage information through a ZigBee coordinator.

3.2. Decision-Making Mechanism. A decision-making mechanism is proposed to dynamically determine the resource allocation over the cluster. During the peak period, it aims at distributing virtual machine loading to all physical machines as balanced as possible. During the off-peak period, through the virtual machine migration and server consolidation, we may turn off unnecessary physical machines to reduce the power consumption.

As shown in Figure 3, the decision-making mechanism consists of four modules: monitor module, load calculation module, load balance module, and migration module. CPU loadings are monitored for all virtual machines. In the monitor module, either overloading event or periodical triggering event may be initiated. When the CPU loading of a physical machine exceeds a certain threshold, an overloading event will enter into the load calculation module. Otherwise, a periodical triggering event is issued in every 10 minutes. The purpose of both events is to determine whether a virtual machine should be migrated and when a physical machine should be consolidated or shut down for energy reduction.

The monitoring operations can be further explained as follows. Once the monitor collects CPU utilization of all VMs, the monitor determines whether the average system CPU loading is over Th1; currently it is defined as 80% for overloading indicator. If the system is overloaded, another physical machine should be turned on. If no one is available, a warning message is sent to management staff. Th2 represents 80% CPU loading of $N - 1$ physical machines, where N is the number of current active physical machines. The average CPU loading is less than Th2 indicating that the system is capable of server consolidation. In such a case, the virtual machine in the physical node with the lowest loading is migrated, as carried out through the migration module, to free the resources. In the case of normal operations, the load balance module may initiate VM migration for the purpose of loading balancing.

3.2.1. Monitor Module. Monitor module is used to record CPU loading of physical and virtual machines, internet traffic, and data from ZigBee digital power meter. The data will be used in other modules. Monitor module can acquire virtual machine information through connection to physical machines by Libvirt mechanism and internet traffic of physical machines through SSH. All the monitored data is recorded in MongoDB. Monitor module can also acquire energy consumption data (watt) by connection with ZigBee coordinator through /dev/tty/BSD0. CPU loading collection for virtual machines will be in real time, so it can be computed to examine whether it has reached Th1 or not. If it does, an overloading event will be issued; otherwise, a periodical trigger event will be sent to enable the decision-making process.

3.2.2. Load Balance Module. This module is to evenly distribute virtual machine loading to run physical machines. The module mainly presents a differencing load balance algorithm and its flow chart is shown in Figure 4. First, we collect CPU loading of physical machines according to

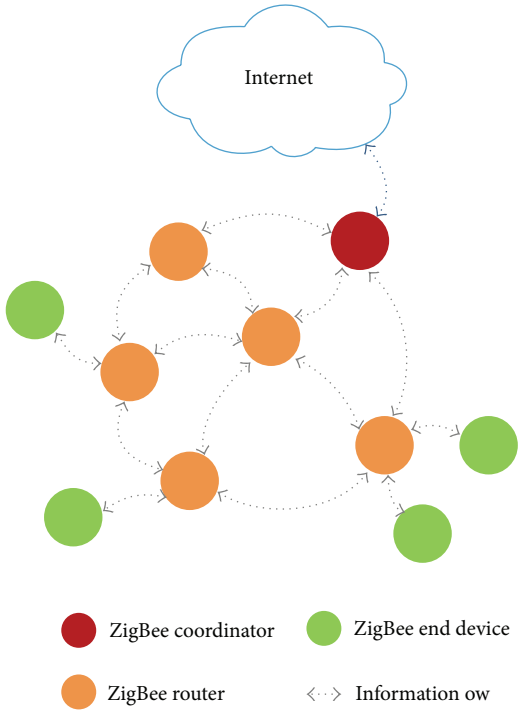

Figure 1: ZigBee Network Topology [13].

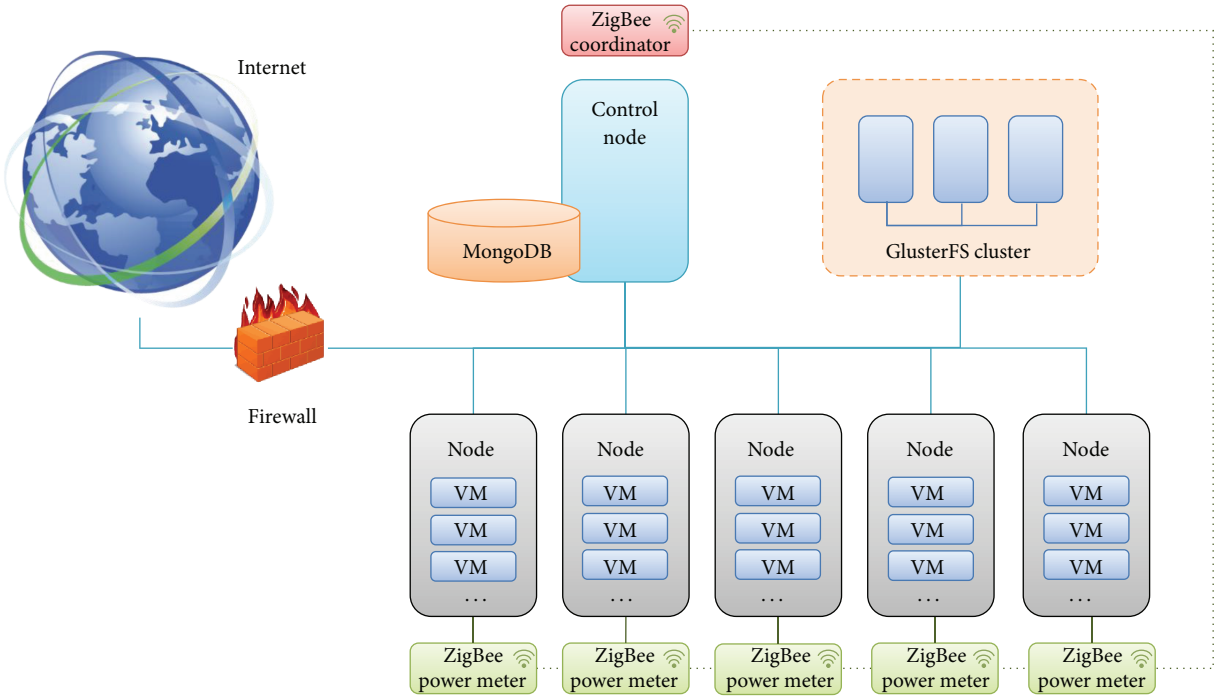

Figure 2: Implementation environment system chart.

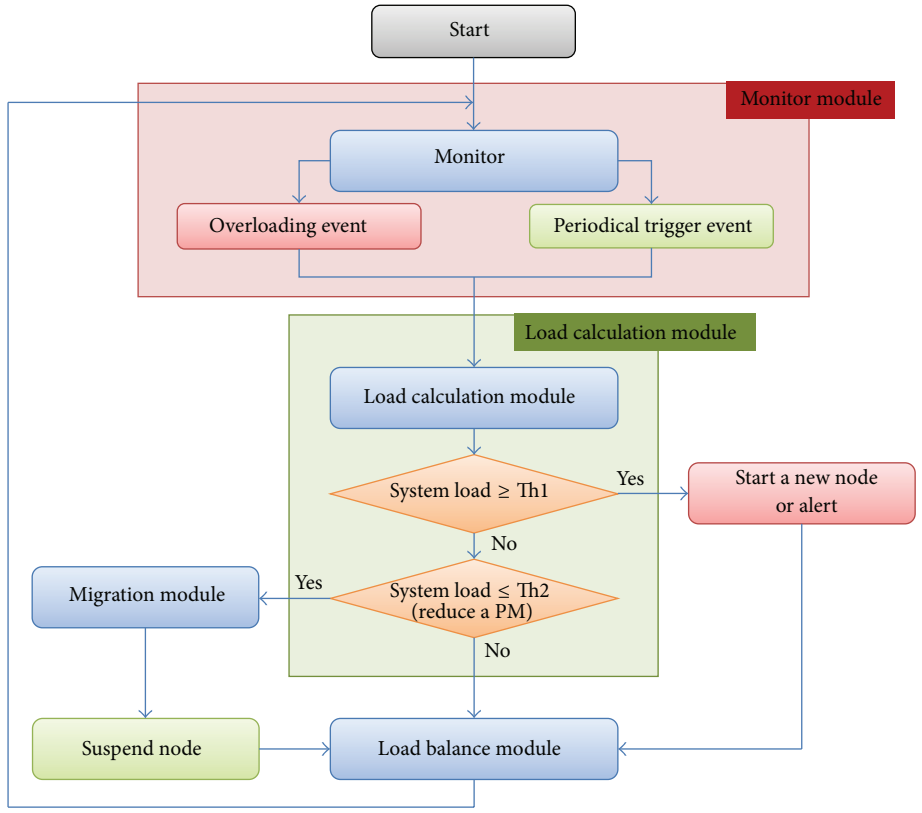

Figure 3: System flow chart of the decision-making mechanism.

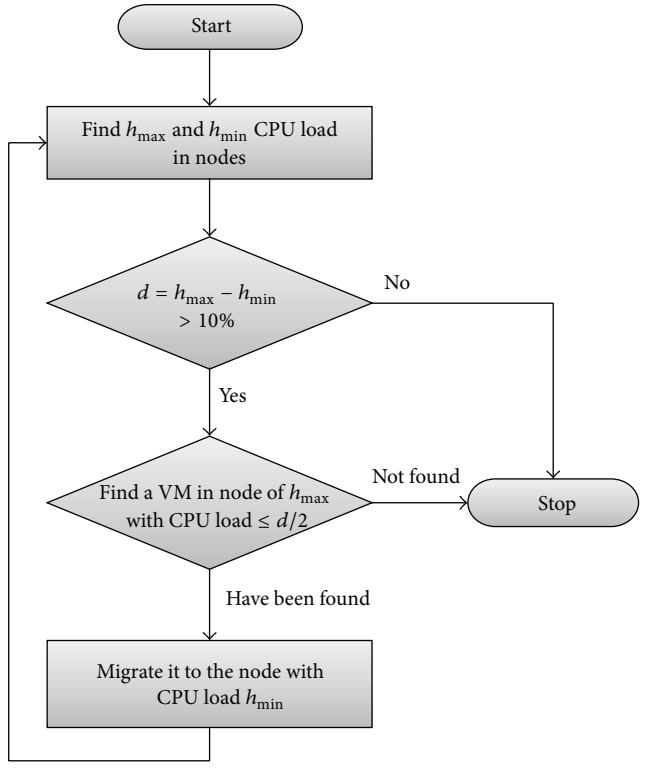

Figure 4: Flow chart of differencing load balance.

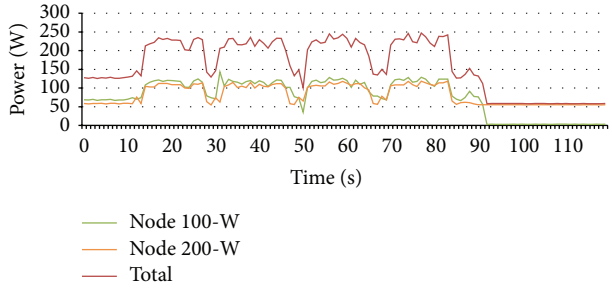

Figure 5: Decision-making mechanism result 1: energy consumption chart.

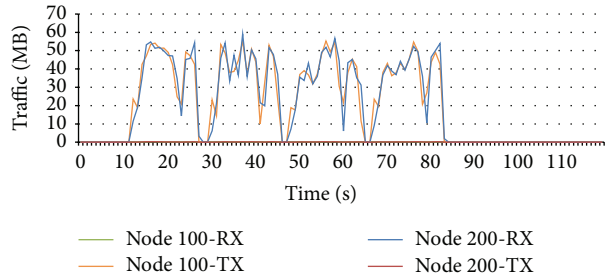

Figure 6: Decision-making mechanism result 1: internet traffic chart.

system monitor and sort the loadings from the largest loading (h_{max}) to the smallest loading (h_{min}). The loading difference between the two physical machines is defined as d. Less than 10% of the difference means the load balance has been reached, while a greater value of d indicates that a virtual machine migration is needed. We then search for the virtual machine in node of h_{max} with CPU loading less than $d/2$; if so, we start the VM migration from node of h_{max} to node of h_{min}. Finally, we recompute the CPU loading of all physical machines to discover the new h_{max} and h_{min}. The above procedure is repeated until the difference between h_{max} and h_{min}, is less than 10%.

4. Power Saving through VM Migration

4.1. Load Balancing over the Physical Machines. Initially, four virtual machines (VMs) are deployed in each physical machine and each VM is equipped with 2 CPU and 1 GB of memory. When the system is idle, all virtual machines on the physical machine, node 100, will be migrated to node 200. We observed that the migration and power-off procedure took around 86 seconds. Figure 5 displays the distribution of power consumption during the migration period, where the "total" stands for the total energy consumption (in watt) of two physical machines. At the beginning, the total is 127W. And, it occasionally reaches up to 228 W while the migration is processing. Once all VMs were successfully migrated to node 200, node 100 was turned off. The experimental results show that only 58 W of energy is required to run eight active virtual machines on node 200, which is 69 W less as compared with running over both physical machines. The network traffic used during the migration process is shown in Figure 6 and the perspective CPU loading of each VM is depicted in Figure 7.

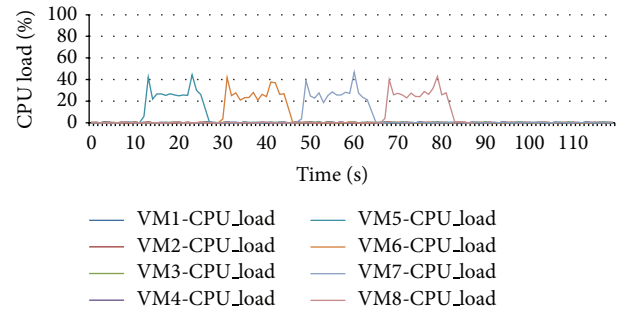

Figure 7: Decision-making mechanism result 1: CPU load in virtual machines.

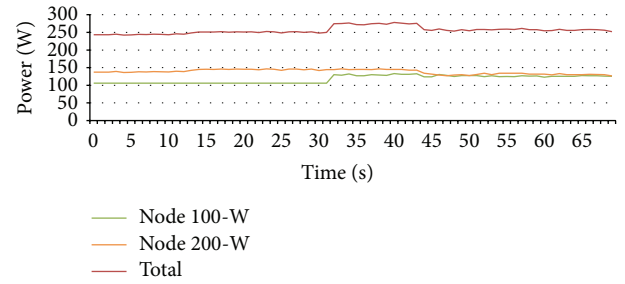

Figure 8: Decision-making result 2: energy consumption chart.

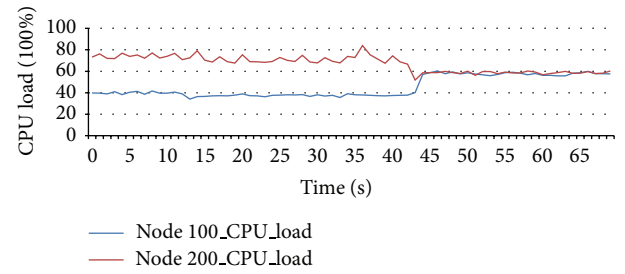

Figure 9: Decision-making result 2: CPU load in physical machines.

4.2. Decision Results of Load Balance in Physical Machines. To demonstrate the effectiveness of load balancing, we apply Lookbusy [20] to generate the CPU loading of each virtual machine (VM). Among eight VMs, which are evenly distributed over two physical machines, the CPU loadings are particularly assigned as 80%, 40%, and six VMs with 20% each. Both VMs with 80% and 40% CPU utilization reside at the same physical machine, node 100, to experience the unbalanced loading. By performing the proposed decision-making mechanism, the VM with 40% CPU utilization was chosen as a target for the purpose of migration. Through the operations of the migration module, this targeted VM is migrated to another physical machine, node 200. The effectiveness of load balancing is shown in both Figures 8 and 9. Figure 8 indicates that the migration process takes around 11 seconds, from 33 seconds to 44 seconds. When the migration is in progress, there are about 10% of increases in

the total power usage. Once the migration is accomplished, both CPU utilization and power consumption of two physical machines tend to be equivalent.

5. Conclusion and Future Work

This paper proposes a mechanism to achieve load balancing through migration in small-scale private clouds. The physical machine resources can be efficiently utilized to meet the quality of service request. In the experiment, when the CPU load of one physical machine is approaching 100%, another physical machine is automatically booted up by the execution of the decision-making algorithm. When the system is not busy, the consolidation process of the proposed mechanism is accomplished by activating the virtual machine live migration automatically. This results in powering off the idle physical machine and, consequently, reducing the energy consumption. In the experiment of load balancing, initially, CPU loads of two physical machines are 80% and 40%, respectively. The proposed mechanism enables the assignment of new-coming request, so that the CPU utilizations of both the machines are promptly balanced with around 60% each.

The mechanism and the system framework proposed in this study are flexible and can be easily extended. Since a large cloud involves more physical machines, and so are the virtual machines, an efficient VM migration with multiple machines introduces another critical management task.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] "Cloud Computing—Wikipedia," http://en.wikipedia.org/wiki/Cloud_computing#Private_cloud.

[2] F. H. Mbuba and W. Y. C. Wang, "Software as a service adoption: impact on IT workers and functions of IT department," *Journal of Internet Technology*, vol. 15, no. 1, pp. 103–114, 2014.

[3] M.-Y. Luo, "Design and implementation of an education cloud," *Journal of Internet Technology*, vol. 15, no. 2, pp. 229–240, 2014.

[4] "Virtual machine—Wikipedia," http://en.wikipedia.org/wiki/Virtual_machine.

[5] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, 2012.

[6] L. Liu, R. Chu, Y. Zhu, P. Zhang, and L. Wang, "DMSS: a dynamic memory scheduling system in server consolidation environments," in *Proceedings of the 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW '12)*, pp. 70–75, April 2012.

[7] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 826–831, May 2010.

[8] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "Dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, 2000.

[9] KVM Main Page, http://www.linux-kvm.org/page/Main_Page.

[10] QEMU, 2014, http://wiki.qemu.org/Main_Page.

[11] libvirt: The virtualization API, 2014, http://libvirt.org/.

[12] Frontpage—Gluster Communtity Website, http://www.gluster org/.

[13] MongoDB, 2014, http://www.mongodb.org.

[14] ZigBee, http://www.zigbee.org/.

[15] A. J. Younge, R. Henschel, J. T. Brown, G. Von Laszewski, J. Qiu, and G. C. Fox, "Analysis of virtualization technologies for high performance computing environments," in *Proceedings of the 4th IEEE International Conference on Cloud Computing (CLOUD '11)*, pp. 9–16, Washington, DC, USA, July 2011.

[16] G. Praveen, "Analysis of performance in the virtual machines environment," *International Journal of Advanced Science and Technology*, vol. 32, pp. 53–64, 2011.

[17] J. SeungHwan, Y. E. Gelogo, and B. Park, "Next generation cloud computing issues and solutions," *International Journal of Control and Automation*, vol. 5, no. 1, pp. 63–70, 2012.

[18] Z. Liu, W. Qu, W. Liu, and K. Li, "Xen live migration with slowdown scheduling algorithm," in *Proceeding of the 11th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '10)*, pp. 215–221, Wuhan, China, December 2010.

[19] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Enabling instantaneous relocation of virtual machines with a lightweight VMM extension," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 73–83, May 2010.

[20] "lookbusy—a synthetic load generator," http://www.devin.com/lookbusy/.