Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 5612039, 11 pages
http://dx.doi.org/10.1155/2016/5612039

Research Article

Virtual Machine Placement Algorithm for Both Energy-Awareness and SLA Violation Reduction in Cloud Data Centers

Zhou Zhou,[1] Zhigang Hu,[1] and Keqin Li[2]

[1] *School of Software, Central South University, Changsha 410083, China*
[2] *Department of Computer Science, State University of New York, New Paltz, NY 12561, USA*

Correspondence should be addressed to Zhou Zhou; zhouzhou03201@126.com

Received 7 February 2016; Accepted 10 March 2016

Academic Editor: Laurence T. Yang

The problem of high energy consumption is becoming more and more serious due to the construction of large-scale cloud data centers. In order to reduce the energy consumption and SLA violation, a new virtual machine (VM) placement algorithm named ATEA (adaptive three-threshold energy-aware algorithm), which takes good use of the historical data from resource usage by VMs, is presented. In ATEA, according to the load handled, data center hosts are divided into four classes: hosts with little load, hosts with light load, hosts with moderate load, and hosts with heavy load. ATEA migrates VMs on heavily loaded or little-loaded hosts to lightly loaded hosts, while the VMs on lightly loaded and moderately loaded hosts remain unchanged. Then, on the basis of ATEA, two kinds of adaptive three-threshold algorithm and three kinds of VMs selection policies are proposed. Finally, we verify the effectiveness of the proposed algorithms by CloudSim toolkit utilizing real-world workload. The experimental results show that the proposed algorithms efficiently reduce energy consumption and SLA violation.

1. Introduction

Cloud computing [1, 2] is derived from grid computing. At present, cloud computing is receiving more and more attention, through which people can access resources in a simple way. In contrast to previous paradigms, cloud computing provides infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).

On one hand, the construction of a large-scale virtualized data centers meets the demand of computational power; on the other hand, such data centers consume a great many of electrical energy resources, leading to high energy consumption and carbon dioxide emissions. It has been reported that [3], in 2013, the total electricity consumption of global data center was more than 4.35 gigawatts, and annual growth rate was by 15%. The high energy consumption problem of virtualized data centers causes a series of problems, including energy wastes, low Return on the Investment (ROI), system instability, and more carbon dioxide emissions [4].

However, most hosts in data centers are in a state of low CPU utilization. Barroso and Hölzle [5] took a survey over half a year and found that most hosts in data centers operate at lower than 50% CPU utilization. Bohrer et al. [6] investigated the problem of high energy consumption and came to the same conclusion. Therefore, it is extremely necessary to reduce the energy consumption of data centers while keeping low SLA (Service Level Agreement) violation [7].

In this paper, we put forward a new VM deployment algorithm (ATEA), two kinds of adaptive three-threshold algorithm (KAM and KAI), and three kinds of VM selection policies to reduce energy consumption and SLA violation. We verify the effectiveness of the proposed algorithms through using the CloudSim toolkit.

The main contributions of the paper are summarized as follows:

(i) Proposing a novel VM deployment algorithm (ATEA). In ATEA, hosts in a data center are divided into four classes according to their load. ATEA migrates VMs on heavily loaded or little-loaded hosts to lightly loaded hosts, while the VMs on lightly loaded and moderately loaded hosts remain unchanged.

(ii) Presenting two kinds of adaptive three-threshold algorithm to determine the three thresholds.

(iii) Putting forward three kinds of VMs selection policies and making three paired t-tests.

(iv) Evaluating the proposed algorithms by extensive simulation using the CloudSim toolkit.

The rest of this paper is organized as follows. In Section 2, the related work is discussed. Section 3 presents the power model, cost of VM migration, SLA violation metrics, and energy efficiency metrics. Section 4 proposes ATEA, two kinds of adaptive three-threshold algorithm, VM selection policy, and VM deployment algorithm. Experiments and performance evaluation are presented in Section 5. Section 6 concludes the paper.

2. Related Work

At present, there are various studies focusing on energy efficient resource management in cloud data centers. Constraint energy consumption algorithm [8–10] and energy efficiency algorithm [11–15] are two main types of algorithms for solving the problem of high energy consumption in data centers. The main idea of the constraint energy consumption algorithm is to reduce the energy consumption in data centers, but this type of algorithm focuses a little on (does not consider) the SLA violation. For example, Lee and Zomaya [8] proposed two heuristic algorithms (ECS, ECS + idle) to decrease the energy consumption, but the two algorithms are easy to fall into local optimum and do not consider the SLA violation. Hanson et al. [9] presented Dynamic Voltage and Frequency Scaling (DVFS) policy to save power in data centers. When the task number is large, DVFS policy raises the voltage of the processor in order to deal with the task in time; when the task number is small, DVFS policy decreases the voltage of processor for the purpose of saving power. Kang and Ranka [10] put forward an energy-saving algorithm, and they supposed that overestimated or underestimated execution time of tasks is bad for energy-saving algorithm. For overestimation, the extra available time should be assigned to other tasks in order to reduce energy consumption. Similarly, this energy-saving algorithm does not consider the SLA violation. Therefore, the constraint energy consumption algorithm does not meet the requirement of users because of focusing a little on (not considering) the SLA violation.

The goal of the energy efficiency (energy consumption and SLA violation) algorithm is to decrease the energy consumption and SLA violation in data centers. For example, Buyya et al. [11] raised a virtual machine (VM) placement algorithm (called Single Threshold (ST)) based on the combination of VM selection policies. ST algorithm sets a unified value for all servers' CPU utilization to make sure all servers are below this value. Obviously, ST algorithm can save energy consumption and decrease the SLA violation, but the SLA violation remains at a high level. Beloglazov and Buyya [12] proposed an energy efficient resource management system, which includes the dispatcher, global manager, local manager, and VMM (VM Monitor). In order to improve the energy efficiency, Beloglazov et al. put forward a new VM migration algorithm called Double Threshold (DT) [13]; DT sets two thresholds to keep all hosts' CPU utilization between the two thresholds. However, the energy consumption and SLA violation for DT algorithm need to be further decreased. Later, Beloglazov and Buyya [14, 15] proposed an adaptive double-threshold VM placement algorithm to improve energy efficiency in data centers. However, the energy consumption in data centers remains at a high level.

In our previous study [16], we proposed a three-threshold energy-aware algorithm named MIMT to deal with the energy consumption and SLA violation. However, the three thresholds for controlling host's CPU utilization are fixed. Therefore, MIMT is not suitable for varying workload. Therefore, it is necessary to put forward a novel VM placement algorithm to deal with energy consumption and SLA violation in cloud data centers.

3. The Power Model, Cost of VM Migration, SLA Violation Metrics, and Energy Efficiency Metrics

3.1. The Power Model. Energy consumption by servers in data centers is connected with its CPU, memory, disk, and bandwidth. Recent studies [17, 18] have illustrated that the energy consumption by servers has a linear relationship with its CPU utilization; even DVFS policy is applied. However, with the decrease of hardware price, multicore CPUs and memory with large-capacity are widely equipped in servers, leading to the traditional linear model not being able to accurately describe energy consumption of servers. In order to deal with this problem, we use the real data of energy consumption offered by SPECpower benchmark (http://www.spec.org/power_ssj2008/).

We have chosen two servers equipped with dual-core CPUs. The main configuration of the two servers is as follows: one is HP ProLiant G4 equipped with 1.86 GHz (dual-core), 4 GB RAM; the other is HP ProLiant G5 equipped with 2.66 GHz (dual-core), 4 GB RAM. The energy consumption for the two servers at different load levels is listed in Table 1 [15].

3.2. Cost of VM Migration. Proper VM migration between servers can reduce energy consumption and SLA violation in data centers. However, excessive VM migration could bring negative impact on performance of application which runs on the VMs. Voorsluys et al. [19] investigated the cost problem of VM migration, and the performance degradation caused by VM can be described in

$$\begin{gathered} C = k \cdot \int_{t_0}^{t_0+T_{m_j}} u_j(t)\,dt, \\ T_{m_j} = \frac{M_j}{B_j}, \end{gathered} \tag{1}$$

Table 1: Power consumption by the two servers at different load levels in Watts.

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP G5	93.7	97	101	105	110	116	121	125	129	133	135

where parameter C represents total performance degradation caused by VM j, parameter k is the coefficient of average performance degradation caused by VMs (in terms of the class of web-applications, the value of k could be estimated as approximately 0.1 (10%) of the CPU utilization [19]), function $u_j(t)$ corresponds to the CPU utilization of VM j, parameter t_0 represents start time of migration, T_{m_j} is completion time, M_j corresponds to the total memory used by VM j, and B_j represents the available bandwidth.

3.3. SLA Violation Metrics. SLA violation is extremely important factor for any VM migration algorithm. At present, there are two ways to describe the SLA violations.

(1) PDM [15] (Overall Performance Degradation Caused by VM Migration). It is indicated in

$$\mathrm{PDM} = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}}, \tag{2}$$

where parameter M represents the number of VMs in data center, C_{d_j} means the estimate of the performance degradation caused by VM j migration, and C_{r_j} corresponds to the total CPU capacity requested by VM j during its lifetime.

(2) SLATAH [15] (SLA Violation Time per Active Host). It means the percentage of total SLA violation time, during which active host's CPU utilization has experienced 100%, as indicated in

$$\mathrm{SLATAH} = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}}, \tag{3}$$

where N represents the number of hosts in data center, T_{s_i} corresponds to the total time, during which the CPU utilization of host i has experienced 100% utilization resulting in the SLA violations, T_{a_i} corresponds to the total time of host i being in active state. The reasoning behind the SLATAH is that the active host's CPU utilization has experienced 100% utilization, the VMs on the host could not be provided with the requested CPU capacity.

Both PDM and SLATAH are two effective methods to independently evaluate the SLA violation. Therefore, the SLA violation is defined as in [15]

$$\mathrm{SLA} = \mathrm{PDM} \times \mathrm{SLATAH}. \tag{4}$$

3.4. Energy Efficiency Metric. Energy efficiency includes energy consumption and SLA violation. Improving the energy efficiency means less energy consumption and SLA violation in data centers. Therefore, the metric of energy efficiency is defined as in

$$E = \frac{1}{P \times \mathrm{SLA}}, \tag{5}$$

where E corresponds to the energy efficiency of a data center, P means the energy consumption of a data center, and SLA represents the SLA violation of a data center. Equation (5) shows that the higher the E, the more the energy efficiency.

4. ATEA, Two Kinds of Adaptive Three-Threshold Algorithm, VM Selection Policy, and VM Deployment Algorithm

4.1. ATEA. VM migration is an effective method to improve the energy efficiency in data centers. However, there are several key problems which should be dealt with: (1) when a host is supposed to be heavily loaded, where some VMs from the host should be migrated to another host; (2) when a host is believed to be moderately loaded or lightly loaded, resulting in a decision to keep all VMs on this host unchanged; (3) when a host is believed to be little-loaded, where all VMs on the host must be migrated to another host; (4) selecting a VM or more VMs that should be migrated from the heavily loaded; (5) finding a new host to accommodate VMs migrated from heavily loaded or little-loaded hosts.

In order to solve the above problems, ATEA (adaptive three-threshold energy-aware algorithm) is proposed. ATEA automatically sets three thresholds T_l, T_m, and T_h ($0 \le T_l < T_m < T_h \le 1$), leading to the data center hosts to be divided into four classes: hosts with little load, hosts with light load, hosts with moderate load, and hosts with heavy load. When CPU utilization of a host is less than or equal to T_l, the host is believed to be little-loaded. In order to save energy consumption, all VMs on little-loaded host must be migrated to another host with light load and the host is switched to sleep mode; when CPU utilization of a host is between T_l and T_m, the host is considered as being lightly loaded. In order to avoid high SLA violations, all VMs on lightly loaded host have to be kept unchanged; the reason is that excessive VMs migration leads to the performance degradation and high SLA violations; when CPU utilization of a host is between T_m and T_h, the host is believed to be moderately loaded; all VMs on moderately loaded host have to be kept unchanged for the reason that excessive VMs migration leads to the performance degradation and high SLA violations; when CPU utilization of a host is greater than T_h, the host is considered as being heavily loaded; in order to reduce the SLA violations, some VMs on heavily loaded host must be migrated to another host with light load. Figure 1 shows the flow chart of algorithm ATEA.

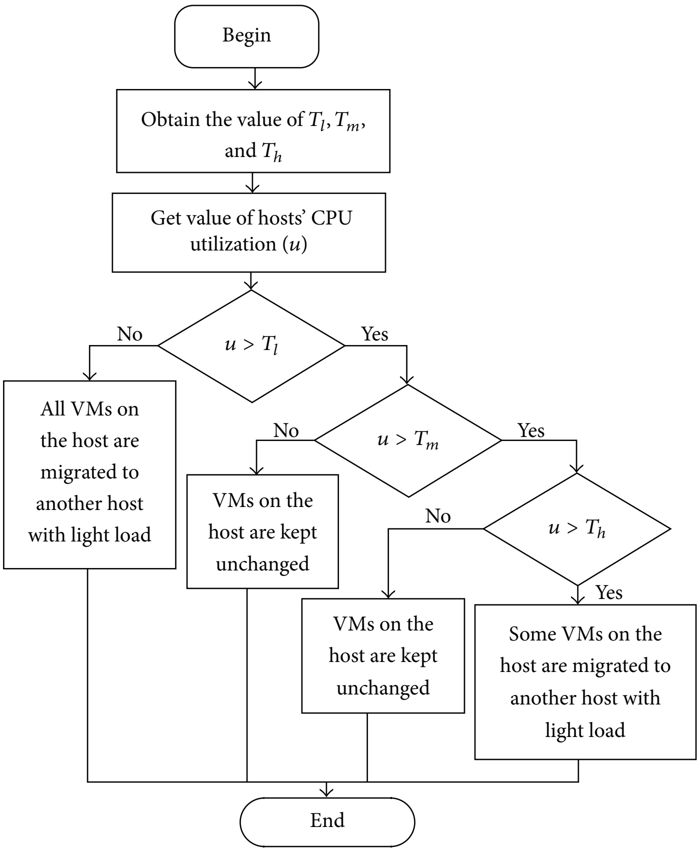

Begin
Obtain the value of T_l, T_m, and T_h
Get value of hosts' CPU utilization (u)
$u > T_l$
No
Yes
All VMs on the host are migrated to another host with light load
$u > T_m$
No
Yes
VMs on the host are kept unchanged
$u > T_h$
No
Yes
VMs on the host are kept unchanged
Some VMs on the host are migrated to another host with light load
End

Figure 1: Flow chart of ATEA.

Different from out previous study [16], where the thresholds are fixed, in this paper, the three thresholds T_l, T_m, and T_h in ATEA are not fixed and these values can be autoadjusted according to workload.

ATEA migrates VMs that must be migrated to other hosts, while keeping some VMs unchanged. In doing so, ATEA improves the migration efficiency of VMs. Therefore, ATEA is a fine-grained algorithm. However, two problems should be solved as for ATEA. Firstly, what are the threshold values of T_l, T_m, and T_h? This problem will be discussed in Section 4.2. Secondly, as mentioned above, some VMs on heavily loaded host must be migrated to another host with light load. Which VM should be migrated? This issue will be discussed in Section 4.3.

The VM placement optimization of ATEA is illustrated in Algorithm 1.

In the first stage, the algorithm inspects each host in host list and decides which host is heavily loaded. If the host is heavily loaded (corresponding to Line 2 in Algorithm 1), the algorithm uses the VM selection policy to choose VMs which must be migrated from the host (corresponding to Line 6 in Algorithm 1). Once VMs list that should be migrated from the heavily loaded is created, the VM deployment algorithm is invoked for the purpose of finding a new host to accommodate the VM (corresponding to Line 7 in Algorithm 1). Function "getNewVmPlacement(vmsToMigrate)" means to find a new host to accommodate the VM. In the second stage, the algorithm inspects each host in host list and decides which host is little-loaded. If the host is little-loaded (corresponding to Line 11 in Algorithm 1), the algorithm chooses all VMs from the host to migrate and finds a placement of the VMs (corresponding to Line 15-Line 16 in Algorithm 1). At last, the algorithm returns the migration map.

4.2. Two Kinds of Adaptive Three-Threshold Algorithm. As discussed in Section 4.1, what are the threshold values of T_l, T_m, and T_h? To solve this problem, two adaptive three-threshold algorithms (KAM and KAI) are proposed.

4.2.1. KAM (K-Means Clustering Algorithm-Average-Median Absolute Deviation). For a univariate data set $V_1, V_2, V_3, \ldots$, V_n (V_i is a host's CPU utilization at time i, and the size of n could be determined by empirical value), the KAM algorithm firstly uses the K-means clustering algorithm to divide the data set $(V_1, V_2, V_3, \ldots, V_n)$ into m groups $(G_1, G_2, \ldots, G_m)$ (the size of m could be obtained by empirical value, and in this paper, $m = 5$), where $G_k = (V_{j_{k-1}+1}, V_{j_{k-1}+2}, \ldots, V_{j_k})$, for all $1 \leq k \leq 5$, and $0 = j_0 < j_1 < j_2 < \cdots < j_5 = n$. Then, KAM gets the average value of each group, formalized as follows:

$$G_{Ak} = \frac{\left(V_{j_{k-1}+1} + V_{j_{k-1}+2} + \cdots + V_{j_k}\right)}{\left(j_k - j_{k-1}\right)}, \qquad (6)$$

```
Require:   hostlist          // hostlist is the set of all hosts
Ensure:   migrationMap
for each host in hostlist do
    if isHostHeavilyLoaded(host) then
        HeavilyLoadedHosts.add(host);   // host is heavily-loaded
    end if
end for
vmsToMigrate = getVmsFromHeavilyLoadedHosts(HeavilyLoadedHosts);
migrationMap.addAll(getNewVmPlacement(vmsToMigrate));
HeavilyLoadedHosts.clear( );
vmsToMigrate.clear( );
for each host in hostlist do
    if isHostLittleLoaded(host) then
        LittleLoadedHosts.add(host); // host is little-loaded
    end if
end for
vmsToMigrate = getVmsToMigrateFromHosts(LittleLoadedHosts);
migrationMap.addAll(getNewVmPlacement(vmsToMigrate));
return migrationMap
```

Algorithm 1: Optimized allocation of VMs.

for all $1 \le k \le 5$. Then, KAM gets the Median Absolute Deviation (MAD) of $G(G_{A1}, G_{A2}, \ldots, G_{A5})$. Therefore, the MAD is defined as follows:

$$\mathrm{MAD} = \mathrm{median}_{Ap}\left(\left|G_{Ap} - \mathrm{median}_{Aq}\left(G_{Aq}\right)\right|\right), \quad (7)$$

where $A1 \le Ap \le A5$ and $\mathrm{median}_{Aq}(G_{Aq})$ is median value of G_{Aq}. Finally, the three thresholds (T_l, T_m, and T_h) in ATEA could be defined as follows:

$$T_l = 0.5\,(1 - r \times \mathrm{MAD}), \quad (8)$$

$$T_m = 0.9\,(1 - r \times \mathrm{MAD}), \quad (9)$$

$$T_h = 1 - r \times \mathrm{MAD}, \quad (10)$$

where $r \in R^+$ represents a parameter of the algorithm that defines how aggressively the system consolidates VMs. For example, the higher r, the more energy consumption, but the less SLA violations caused by VMs consolidation. The complexity of KAM is $O(m \times n \times t)$, where m is the group number, n denotes the data size, and t is the iteration number.

As the value of V_i $(i = 1, 2, 3, \ldots, n)$ varies from time to time, the value of T_l, T_m, and T_h are also variable. Therefore, KAM is an adaptive three-threshold algorithm. When the workloads are dynamic and unpredictable, as compared with a fixed threshold algorithm, KAM generates higher energy efficiency by setting the value of T_l, T_m, and T_h.

4.2.2. KAI (K-Means Clustering Algorithm-Average-Interquartile Range). KAI is another adaptive threshold algorithm. For a univariate data set $V_1, V_2, V_3, \ldots, V_n$ (V_i is a host's CPU utilization at time i, and the size of n could be determined by empirical value), the KAI algorithm firstly uses the K-means clustering algorithm to divide the data set $(V_1, V_2, V_3, \ldots, V_n)$ into m groups $(G_1, G_2, \ldots, G_m)$ (the size of m could be obtained by empirical value, and in this paper, $m = 5$), where $G_k = (V_{j_{k-1}+1}, V_{j_{k-1}+2}, \ldots, V_{j_k})$, for all $1 \le k \le 5$, and $0 = j_0 < j_1 < j_2 < \cdots < j_5 = n$. Then, KAI gets the average value of each group, formalized as follows:

$$G_{Ak} = \frac{\left(V_{j_{k-1}+1} + V_{j_{k-1}+2} + \cdots + V_{j_k}\right)}{\left(j_k - j_{k-1}\right)}, \quad (11)$$

for all $1 \le k \le 5$. Then, KAI gets the Interquartile Range (IR) of $G(G_{A1}, G_{A2}, \ldots, G_{A5})$. Therefore, the IR is defined as follows:

$$\mathrm{IR} = Q_3 - Q_1, \quad (12)$$

where Q_3 is third quartiles of G and Q_1 is first quartiles of G. Finally, the three thresholds (T_l, T_m, and T_h) in ATEA can be defined as follows:

$$T_l = 0.5\,(1 - r \times \mathrm{IR}), \quad (13)$$

$$T_m = 0.9\,(1 - r \times \mathrm{IR}), \quad (14)$$

$$T_h = 1 - r \times \mathrm{IR}, \quad (15)$$

where $r \in R^+$ represents a parameter of the algorithm that defines how aggressively the system consolidates VMs. For example, the higher r, the more energy consumption, but the less SLA violations caused by VMs consolidation. The complexity of KAI is $O(m \times n \times t)$, where m is the group number, n denotes the data size, and t is the iteration number.

As the value of V_i $(i = 1, 2, 3, \ldots, n)$ varies from time to time, the values of T_l, T_m, and T_h are also variable. Therefore, KAI is an adaptive three-threshold algorithm. When the workloads are dynamic and unpredictable, as compared with fixed threshold algorithm, KAI generates higher energy efficiency by setting the value of T_l, T_m, and T_h.

4.3. VM Selection Policies. As described earlier in Section 4.1, some VMs on heavily loaded host must be migrated to another host with light load. Which VM should be migrated? In general, a host's CPU utilization and memory size affect its energy efficiency. Therefore, to solve this problem, three kinds of VM selection policies (MMS, LCU, and MPCM) are proposed in this section.

4.3.1. MMS (Minimum Memory Size) Policy. The migration time of a VM will change, depending on its different memory size. A VM with less memory size means less migration time under the same spare network bandwidth. For example, a VM with 16 GB memory may take 16 times' longer migration time than a VM with 1 GB memory. Clearly, selecting the VM with 16 GB memory or the VM with 1 GB memory greatly affects energy efficiency of data centers. Therefore, if a host is being heavily loaded, MMS policy selects a VM with the minimum memory size to migrate compared with the other VMs allocated to the host. MMS policy chooses a VM u that satisfies the following condition:

$$\mathrm{RAM}(u) \leq \mathrm{RAM}(v), \quad \forall v \in \mathrm{VM}_i, \tag{16}$$

where VM_i means the set of VMs allocated to host i and $\mathrm{RAM}(u)$ is the memory size currently utilized by the VM u.

4.3.2. LCU (Lowest CPU Utilization) Policy. As for energy efficiency in data center, the CPU utilization of a host is also another important factor. Therefore, if a host is being heavily loaded, LCU policy chooses a VM with the lowest CPU utilization to migrate compared with the other VMs allocated to the host. LCU policy chooses a VM u that satisfies the following condition:

$$\mathrm{Utilization}(u) \leq \mathrm{Utilization}(v), \quad \forall v \in \mathrm{VM}_i, \tag{17}$$

where VM_i means the set of VMs allocated to host i and $\mathrm{Utilization}(u)$ is the CPU utilization of VM u allocated to host i.

4.3.3. MPCM (Minimum Product of Both CPU Utilization and Memory Size) Policy. As host's CPU utilization and memory size are two important factors for energy efficiency in data center, if a host is being heavily loaded, MPCM policy selects a VM with the minimum product of both CPU utilization and memory size to migrate compared with the other VMs allocated to the host. MPCM policy chooses a VM u that satisfies the following condition:

$$\left(u_{\mathrm{CPU}} \times u_{\mathrm{memory}}\right) \leq \left(v_{\mathrm{CPU}} \times v_{\mathrm{memory}}\right), \quad \forall v \in \mathrm{VM}_i, \tag{18}$$

where VM_i means the set of VMs allocated to host i and u_{CPU} and u_{memory}, respectively, represent CPU utilization and memory size currently utilized by the VM u.

4.4. VM Deployment Algorithm. VM deployment can be considered as a bin packing problem. In order to tackle the problem, a modification of the Best Fit Decreasing algorithm denoted by Energy-Aware Best Fit Decreasing (EBFD) could be used to deal with it. As described earlier in Section 4.1, VMs on heavily loaded host or VMs on little-loaded host must be migrated to another host with light load (CPU utilization of a host at T_l-T_m interval); VMs on lightly loaded host or VMs on moderately loaded host are kept unchanged.

Algorithm 2 shows the pseudocode of EBFD, where T_l, T_m, and T_h are three thresholds in ATEA (the definition of the three thresholds in Section 4.2). "vmlist" is the set of all VMs. "hostlist" represents all hosts in the data centers. Line 1 (see Algorithm 2) means to sort all VM by CPU utilization in descending order. Line 3 represents that parameter "minimumPower" is assigned a maximum value. Line 6 is to check whether the host is suitable to accommodate the VM (e.g., host's CPU capacity, memory size, and bandwidth). Function getUtilizationAfterAllocation means to obtain host's CPU utilization after allocating a VM. Line 7 to Line 9 (see Algorithm 2) are to keep a host with light load (CPU utilization of a host at T_l-T_m interval). Function getPowerAfterVM is to obtain the increasing of host's energy consumption after allocating a VM. Line 11 to Line 15 are to find the host which owns the least increasing of power consumption caused by VM allocation. Line 19 is to return the destination hosts for accommodating VMs. The complexity of the EBFD is $O(m \times n)$; parameter m represents the number of hosts, whereas parameter n corresponds to the number of VMs which should be allocated.

5. Experiments and Performance Evaluation

5.1. Experiment Setup. Due to the advantages of CloudSim toolkit [20, 21] such as supporting on demand dynamic resource provisioning and modeling of virtualized environments and so on, we choose it as the simulation toolkit for our experiments.

We have simulated a data center which includes 800 heterogeneous physical nodes, half of which consists of HP ProLiant G4 (Intel Xeon 3040, 2 cores 1860 MHz, 4 GB), and the other half are HP ProLiant G5 (Intel Xeon 3075, 2 cores 2660 MHz, 4 GB). There are 1052 VMs and four kinds of VM types (High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB)) in the data center. The characteristics of the VMs correspond to Amazon EC2 instance types.

5.2. Workload Data. Using real workload to do experiments is extremely significant for VM placement. In this paper, we utilize the workload coming from a CoMon project, which mainly monitors infrastructure for PlanetLab [22]. We obtain the data derived from more than a thousand VMs' CPU utilization and the VMs placed at more than 500 places across the world. Table 2 [15] shows the characteristics of the data.

5.3. Experimental Results and Analysis. By using the workload data (described in Section 5.2), two kinds of adaptive three-threshold algorithm (KAM and KAI) and three kinds of VM selection policies (MMS, LCU, and MPCM) are simulated. In addition, for the two adaptive three-threshold

```
Require:   $T_l$, $T_m$, $T_h$, vmlist, hostlist
Ensure:   migrationMap
vmList.sortByCpuUtilization( );
// sorted by CPU utilizatioin in descending order
for each vm in vmlist do
    minimumPower = maximum;
    // minimunPower is assigned a maximum value
    allocatedHost = null;
    for each host in hostlist do
        if (host is Suitable for Vm (vm)) then
            utilization = getUtilizationAfterAllocation(host, vm);
            if ((utilization < $T_l$) || (utilization > $T_m$)) then
                continue;
            end if
            EnergyConsumption = getPowerAfterVM(host, vm);
            if (EnergyConsumption < minimumPower) then
                minimumPower = EnergyConsumption;
                allocatedHost = host;
            end if
        end if
    end for
end for
return allocationHost.
```

Algorithm 2: Energy-Aware Best Fit Decreasing (EBFD).

Table 2: Workload data characteristics (CPU utilization).

Date	Number of VMs	Mean	St. dev.	Quartile 1	Median	Quartile 3
03/03/2011	1052	12.31%	17.09%	2%	6%	15%

algorithm we have varied the parameters (r (7)–(9)) and ((11)–(13)) form 0.5 to 3.0 increasing by 0.5. Therefore, these variations have led to 36 combinations of the algorithms and parameters. For example, for KAM, there are three VM selection policies (MMS, LCU, and MPCM) denoted by KAM-MMS, KAM-LCU, and KAM-MPCM, respectively. Similarly, for KAI, there are also three VM selection policies (MMS, LCU, and MPCM) denoted by KAI-MMS, KAI-LCU, and KAI-MPCM, respectively. In the following section, we will discuss the energy consumption, SLATAH, PDM, SLA violations, and energy efficiency for the algorithms with different parameter.

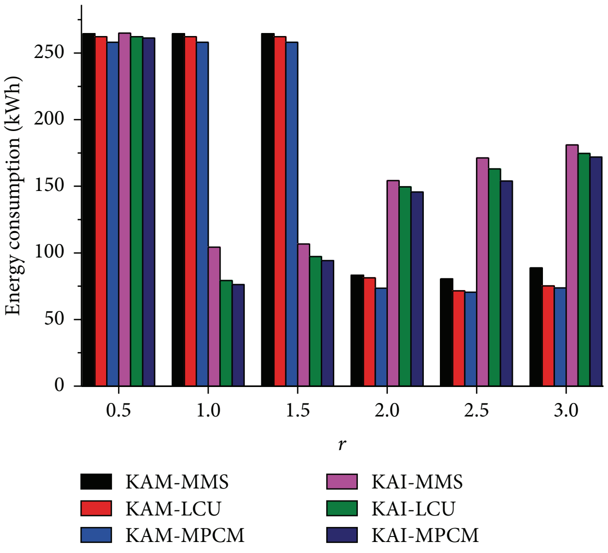


Figure 2: The energy consumption of the six algorithms.

(1) Energy Consumption. For the six algorithms (KAM-MMS, KAM-LCU, KAM-MPCM, KAI-MMS, KAI-LCU, and KAI-MPCM) with different parameter (0.5 to 3.0 increasing by 0.5), the energy consumption is shown in Figure 2, which shows the energy consumption for the six algorithms. As for KAM-MMS, KAM-LCU, and KAM-MPCM, KAM-MPCM leads to the least energy consumption, KAM-LCU the second energy consumption, and KAM-MMS the most energy consumption. The reason is that KAM-MPCM considers both CPU utilization and memory size when a host is with heavy load. Compared with KAM-MMS, KAM-LCU leads to less energy consumption. This can be explained by the fact that the processor (CPU) of a host consumes much more energy than its memory. Similarly, as for KAI-MMS, KAI-LCU, and KAI-MPCM, KAI-MPCM leads to the least energy consumption, KAI-LCU the second energy consumption, and KAI-MMS the most energy consumption. The reason is that KAI-MPCM considers both CPU utilization and memory size when a host is with heavy load. Compared with

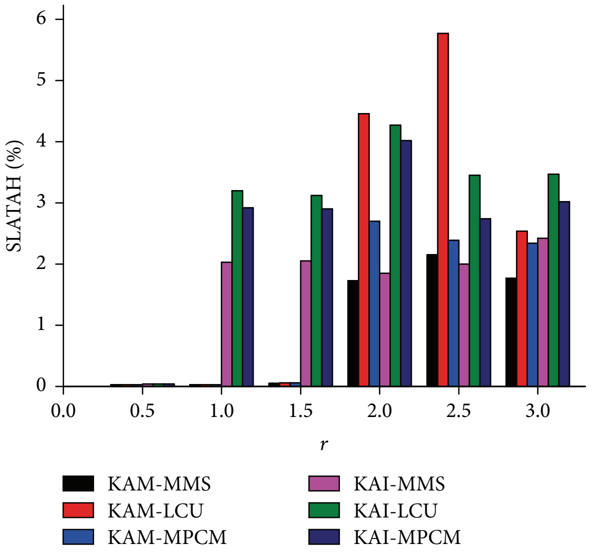


Figure 3: The SLATAH of the six algorithms.

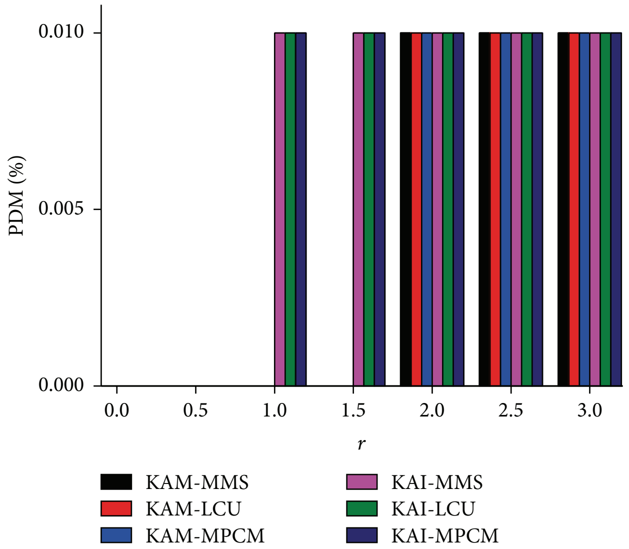


Figure 4: The PDM of the six algorithms.

KAI-MMS, KAI-LCU leads to less energy consumption. This can be also explained by the fact that the processor (CPU) of a host consumes much more energy than its memory.

(2) SLATAH. The SLATAH is described in Section 3.3 (see (3)). For the six algorithms (KAM-MMS, KAM-LCU, KAM-MPCM, KAI-MMS, KAI-LCU, and KAI-MPCM) with different parameter (0.5 to 3.0 increasing by 0.5), the SLATAH is shown in Figure 3, which shows the SLATAH for the six algorithms. In terms of KAM-MMS, KAM-LCU, and KAM-MPCM, KAM-MMS contributes to the least SLATAH, KAM-MPCM the second, and KAM-LCU the most. The reason is as follows: when a host is with heavy load, KAM-MMS selects a VM with the minimum memory size to migrate leading to less migration time. Therefore, KAM-MMS leads to the least SLATAH compared with KAM-LCU and KAM-MPCM. Compared with KAM-MPCM, KAM-LCU contributes to much more SLATAH. This could be explained by the fact that SLATAH mainly depends on memory size but not CPU utilization. Similarly, as for KAI-MMS, KAI-LCU, and KAI-MPCM, KAI-MMS contributes to the least SLATAH, KAI-MPCM the second, and KAI-LCU the most. The reason is that KAI-MMS causes the least migration time leading to the least SLATAH. Compared with KAI-MPCM, KAI-LCU leads to much more SLATAH. This could also be explained by the fact that SLATAH mainly depends on memory size but not CPU utilization.

(3) PDM. The PDM is described in Section 3.3 (see (2)). For the six algorithms (KAM-MMS, KAM-LCU, KAM-MPCM, KAI-MMS, KAI-LCU, and KAI-MPCM) with different parameter (0.5 to 3.0 increasing by 0.5), the PDM is shown in Figure 4, which illustrates the PDM for the six algorithms. As for KAM-MMS, KAM-LCU, and KAM-MPCM, the PDM is the same. This could be explained by the fact that the overall performance degradation caused by VMs due to migration is the same. Furthermore, when parameter $r = 0.5$, 1.0, and 1.5, respectively, the corresponding PDM of the three algorithms (KAM-MMS, KAM-LCU, and KAM-MPCM) are 0. As for KAI-MMS, KAI-LCU, and KAI-MPCM, the PDM is also the same. This could also be explained by the fact that the overall performance degradation caused by VMs due to migration is the same. At the same time, when parameter $r = 0.5$, the PDM of the three algorithms (KAI-MMS, KAI-LCU, and KAI-MPCM) are 0.

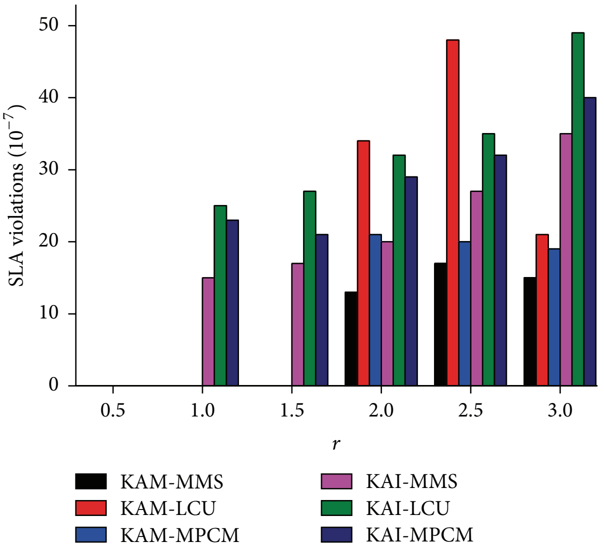


Figure 5: The SLA violations of the six algorithms.

(4) SLA Violations. The SLA violations are described in Section 3.3 (see (4)). For the six algorithms (KAM-MMS, KAM-LCU, KAM-MPCM, KAI-MMS, KAI-LCU, and KAI-MPCM) with different parameter (0.5 to 3.0 increasing by 0.5), the SLA violations are shown in Figure 5, which shows the SLA violations for the six algorithms. From (4), the SLA

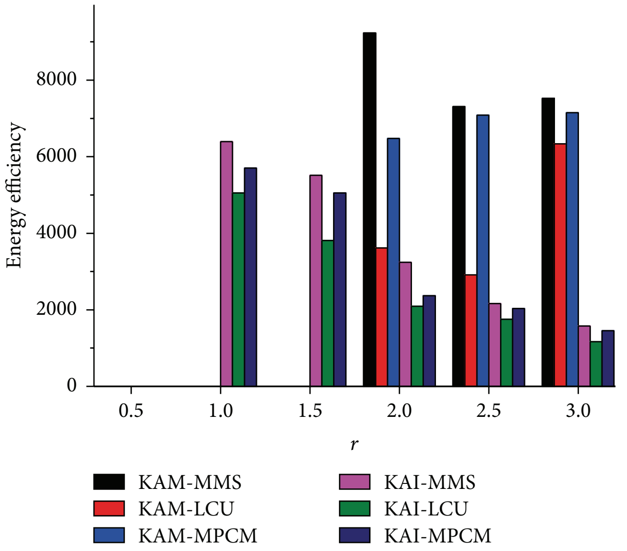


Figure 6: The energy efficiency of the six algorithms.

violations depend on SLATAH (Figure 3) and PDM (Figure 4). As for KAM-MMS, KAM-LCU, and KAM-MPCM, the PDM is the same. Therefore, SLA violations depend on SLATAH. In terms of KAM-MMS, KAM-LCU, and KAM-MPCM, KAM-MMS contributes to the least SLATAH, KAM-MPCM the second, and KAM-LCU the most. So, KAM-MMS contributes to the least SLA violations, KAM-MPCM the second, and KAM-LCU the most. Furthermore, when parameter $r = 0.5$, 1.0, and 1.5, respectively, the corresponding PDM of the three algorithms (KAM-MMS, KAM-LCU, and KAM-MPCM) are 0. Therefore, the corresponding SLA violations of the three algorithms (KAM-MMS, KAM-LCU, and KAM-MPCM) are 0. By the same way, as for KAI-MMS, KAI-LCU, and KAI-MPCM, KAI-MMS contributes to the least SLA violations, KAI-MPCM the second, and KAI-LCU the most. At the same time, when parameter $r = 0.5$, the PDM of the three algorithms (KAI-MMS, KAI-LCU, and KAI-MPCM) are 0. Therefore, the SLA violations of the three algorithms (KAI-MMS, KAI-LCU, and KAI-MPCM) are 0.

(5) Energy Efficiency. For the six algorithms (KAM-MMS, KAM-LCU, KAM-MPCM, KAI-MMS, KAI-LCU, and KAI-MPCM) with different parameter (0.5 to 3.0 increasing by 0.5), the energy efficiency (E) is shown in Figure 6, which shows the energy efficiency of the six algorithms. As discussed in Section 3.4, (5) depends on energy consumption (Figure 2) and SLA violation (Figure 5). Compared with KAM-LCU and KAM-MPCM, the energy efficiency of KAM-MMS is the most. The reason is that KAM-MMS reduces the migration time of VMs. In terms of energy efficiency, KAM-MPCM is better than KAM-LCU. This can be explained by the fact that KAM-MPCM considers both CPU utilization and memory size. As (5) is related to energy consumption (Figure 2) and SLA violation (Figure 5), when parameter $r = 0.5$, 1.0, and 1.5, respectively, the corresponding SLA violations of the three algorithms (KAM-MMS, KAM-LCU, and KAM-MPCM) are 0. Therefore, the corresponding energy efficiency of the three algorithms (KAM-MMS, KAM-LCU, and KAM-MPCM) is 0. Similarly, compared with KAI-LCU and KAI-MPCM, the energy efficiency of KAI-MMS is the most; the reason is that KAI-MMS reduces the migration time of VMs. In terms of energy efficiency, KAI-MPCM is better than KAI-LCU. This can be explained by the fact that KAI-MPCM considers both CPU utilization and memory size. As (5) is related to energy consumption (Figure 2) and SLA violation (Figure 5), when parameter $r = 0.5$, the corresponding SLA violations of the three algorithms (KAI-MMS, KAI-LCU, and KAI-MPCM) are 0. Therefore, the corresponding energy efficiency of the three algorithms (KAI-MMS, KAI-LCU, and KAI-MPCM) is 0.

Table 3: Comparison of the VMs select policies using paired t-tests.

Policy 1	Policy 2	Difference	P value
MMS (3579.5)	LCU (2228.3)	1531.2	P value < 0.05
MMS (3579.5)	MPCM (3109.9)	469.6	P value > 0.05
MPCM (3109.9)	LCU (2228.3)	881.6	P value < 0.05

Considering energy efficiency, we choose the parameter of six algorithms to maximize the energy efficiency. Figure 6 illustrates that parameter $r = 2.0$ for KAM-MMS is the best (denoted by KAM-MMS-2.0), parameter $r = 3.0$ for KAM-LCU is the best (denoted by KAM-LCU-3.0), parameter $r = 3.0$ for KAM-MPCM is the best (denoted by KAM-MPCM-3.0), parameter $r = 1.0$ for KAI-MMS is the best (denoted by KAI-MMS-1.0), parameter $r = 1.0$ for KAI-LCU is the best (denoted by KAI-LCU-1.0), and parameter $r = 1.0$ for KAI-MPCM is the best (denoted by KAI-MPCM-1.0).

For the two kinds of adaptive three-threshold algorithms (KAM and KAI), there are three VMs select policies which could be provided. Does a policy that is the best compared with other two policies in regard to energy efficiency exist? If it exists, which VM select policy is the best? In order to solve this problem, we have made three paired t-tests to determine which VM policy is the best in regard to energy efficiency. Before using the three paired t-tests, according to Ryan-Joiner's normality test, we verify the three VM select policies (MMS, LCU, and MPCM) with different parameter (r) that maximization energy efficiency follows a normal distribution with P value $= 0.2 > 0.05$. The paired t-tests results are showed in Table 3, which shows that MMS leads to a statistically significantly upper value of energy efficiency with P value < 0.05 compared with LCU and MPCM. In other words, in terms of energy efficiency, MMS is the best VM select policy compared with LCU and MPCM. Therefore, KAM-MMS-2.0 and KAI-MMS-1.0 are the best combination in regard to energy efficiency. In the following section, we use KAM-MMS-2.0 and KAI-MMS-1.0 to make comparison with other energy-saving algorithms.

(6) Comparison with Other Energy-Saving Algorithms. In this section, the NPA (Nonpower Aware), DVFS [9], THR-MMT-1.0 [15], THR-MMT-0.8 [15], MAD-MMT-2.5 [15], IQR-MMT-1.5 [15], and MIMT [16] are chosen to make comparison in regard to energy efficiency. The related experimental results are shown in Table 4.

Table 4: The comparison of the energy-saving algorithms.

Algorithm	Energy efficiency (KWh)	Energy consumption ($\times 10^{-7}$)	SLA violations	SLATAH (%)	PDM (%)	Number of VM migrations
NPA	—	2410.8	—	—	—	—
DVFS	—	803.91	—	—	—	—
THR-MMT-1.0	38	99.95	2613	26.97	0.10	19852
THR-MMT-0.8	170	119.40	492	4.99	0.10	26567
MAD-MMT-2.5	169	114.27	518	5.24	0.10	25923
IQR-MMT-1.5	166	117.08	514	5.08	0.10	26420
MIMT (0–40%–80%)	5420	108.53	17	2.86	0.01	8418
MIMT (5%–45%–85%)	4949	112.26	18	3.22	0.01	8687
KAM-MMS-2.0	9231	83.33	13	1.73	0.01	6808
KAI-MMS-1.0	6393	104.28	15	2.03	0.01	7519

Table 4 illustrates the energy efficiency, energy consumption, and SLA violations for the energy-saving algorithms. As the NPA algorithm does not take any energy-saving policy leading to 2410.8 KWh, DVFS algorithm reduces this value to 803.91 KWh by adjusting its CPU frequency dynamically. Because NPA and DVFS algorithm have no VMs migration, the symbol "—" is used to represent the energy efficiency, SLA violations, SLATAH, PDM, and number of VMs' migration. In terms of energy efficiency, THR-MMT-1.0 and THR-MMT-0.8 algorithms are better than DVFS. The reason is that THR-MMT-1.0 and THR-MMT-0.8 algorithms set the fixed threshold to migrate VMs leading to upper energy efficiency.

MAD-MMT-2.5 and IQR-MMT-1.5 are two kinds of adaptive threshold algorithm, which (MAD-MMT-2.5 and IQR-MMT-1.5) are better than THR-MMT-1.0 in regard to energy efficiency. This could be described by the fact that MAD-MMT-2.5 and IQR-MMT-1.5 dynamically set two thresholds to improve the energy efficiency. But compared with THR-MMT-0.8, the three algorithms (MAD-MMT-2.5, IQR-MMT-1.5, and THR-MMT-0.8) are at the same level in regard to energy efficiency. This could be explained by the fact that the 0.8 threshold value is a proper value for the THR-MMT algorithm.

KAM-MMS-2.0 and KAI-MMS-1.0 algorithms are better than MIMT (0–40%–80%) and MIMT (5%–45%–85%). This could be explained by the fact that MIMT (0–40%–80%) and MIMT (5%–45%–85%) set the fixed threshold to control the migration of VMs, while KAM-MMS-2.0 and KAI-MMS-1.0 autoadjust threshold to control the migration of VMs according to the workload.

Table 4 also shows that KAM-MMS-2.0 and KAI-MMS-1.0 algorithms, respectively, improve the energy efficiency compared with IQR-MMT-1.5, MAD-MMT-2.5, THR-MMT-0.8, and THR-MMT-1.0 and maintain the low SLA violation of 13×10^{-7}. In addition, KAM-MMS-2.0 and KAI-MMS-1.0 algorithms, respectively, generate 2-3 times fewer VMs migrations than IQR-MMT-1.5, MAD-MMT-2.5, THR-MMT-0.8, and THR-MMT-1.0. The reason is as follows. On one hand, KAM-MMS-2.0 and KAI-MMS-1.0 are two kinds of adaptive three-threshold algorithm, which dynamically set three thresholds, leading to the data center hosts to be divided into four classes (hosts with little load, hosts with light load, hosts with moderate load, and hosts with heavy load), and migrate the VMs on heavily loaded and little-loaded hosts to the host with light load, while the VMs on lightly loaded and moderately loaded hosts remain unchanged. Therefore, KAM-MMS-2.0 and KAI-MMS-1.0 algorithm reduce the migration time and improve the migration efficiency compared with other energy-saving algorithms. On the other hand, KAM-MMS-2.0 and KAI-MMS-1.0, respectively, select the best VMs select policy (MMS) combination with the best parameter (r) compared with other energy-saving algorithms.

6. Conclusions

This paper proposes a novel VMs deployment algorithm based on the combination of adaptive three-threshold algorithm and VMs selection policies. This paper shows that dynamic thresholds are more energy efficient than fixed threshold. The proposed algorithms are expected to be applied in real-world cloud platforms, aiming at reducing the energy costs for cloud data centers.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Natural Science Foundation (nos. 61572525 and 61272148), the Ph.D. Programs Foundation of Ministry of Education of China (no. 20120162110061), the Fundamental Research Funds for the Central Universities of Central South University (no. 2014zzts044), the Hunan Provincial Innovation Foundation for Postgraduate (no. CX2014B066), and China Scholarship Council.

References

[1] H. Khazaei, J. Mišić, V. B. Mišić, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 849–861, 2013.

[2] J. Carretero and J. G. Blas, "Introduction to cloud computing: platforms and solutions," *Cluster Computing*, vol. 17, no. 4, pp. 1225–1229, 2014.

[3] L. Wang and S. U. Khan, "Review of performance metrics for green data centers: a taxonomy study," *Journal of Supercomputing*, vol. 63, no. 3, pp. 639–656, 2013.

[4] D. Rincón, A. Agustí-Torra, J. F. Botero et al., "A novel collaboration paradigm for reducing energy consumption and carbon dioxide emissions in data centres," *The Computer Journal*, vol. 56, no. 12, pp. 1518–1536, 2013.

[5] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[6] P. Bohrer, E. N. Elnozahy, T. Keller et al., "The case for power management in web servers," in *Power Aware Computing*, Computer Science, pp. 261–289, Springer, Berlin, Germany, 2002.

[7] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[8] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1374–1381, 2011.

[9] H. Hanson, S. W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rubio, "Thermal response to DVFS: analysis with an Intel Pentium M," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '07)*, pp. 219–224, August 2007.

[10] J. Kang and S. Ranka, "Dynamic slack allocation algorithms for energy minimization on parallel machines," *Journal of Parallel and Distributed Computing*, vol. 70, no. 5, pp. 417–430, 2010.

[11] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities," in *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS '09)*, pp. 1–11, Leipzig, Germany, June 2009.

[12] 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, IEEE, Melbourne, Australia, May 2010.

[13] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[14] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the ACM 8th International Workshop on Middleware for Grids, Clouds and e-Science (MGC '10)*, pp. 1–6, Bangalore, India, December 2010.

[15] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Computation Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[16] Z. Zhou, Z.-G. Hu, T. Song, and J.-Y. Yu, "A novel virtual machine deployment algorithm with energy efficiency in cloud computing," *Journal of Central South University*, vol. 22, no. 3, pp. 974–983, 2015.

[17] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*, pp. 13–23, ACM, June 2007.

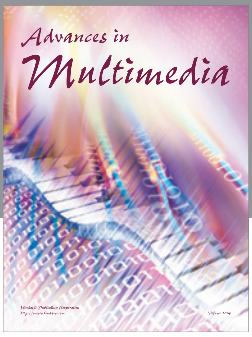
[18] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Computing*, vol. 12, no. 1, pp. 1–15, 2009.

[19] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: a performance evaluation," in *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1–4, 2009. Proceedings*, vol. 5931 of *Lecture Notes in Computer Science*, pp. 254–265, Springer, Berlin, Germany, 2009.

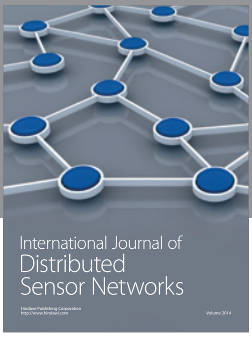
[20] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

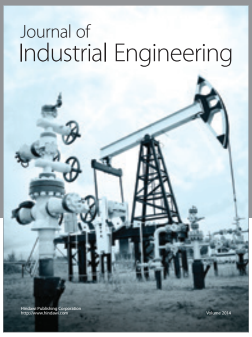
[21] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA '10)*, pp. 446–452, IEEE, Perth, Australia, April 2010.

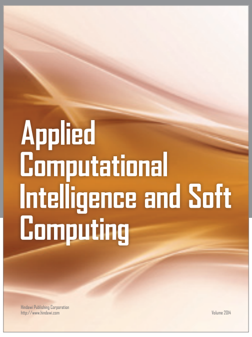
[22] K. S. Park and V. S. Pai, "CoMon: a mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.

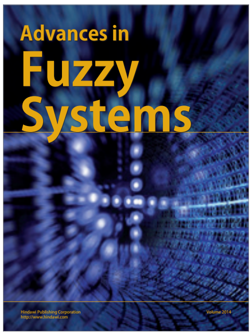

Advances in
Multimedia

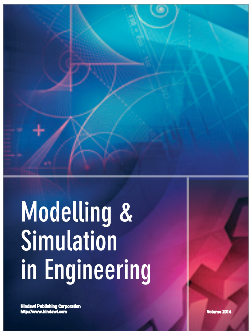
The Scientific
World Journal

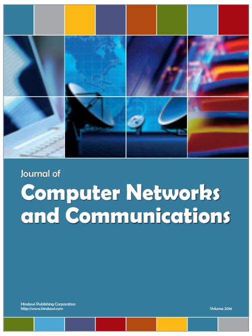

International Journal of
Distributed
Sensor Networks

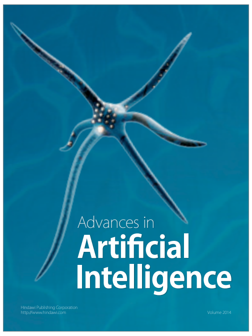

Journal of
Industrial Engineering

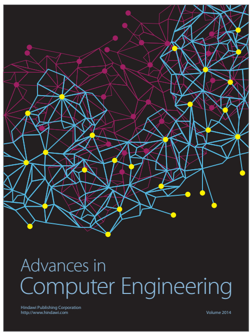

Applied
Computational
Intelligence and Soft
Computing


Advances in
Fuzzy
Systems

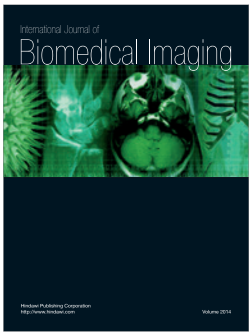

Modelling &
Simulation
in Engineering

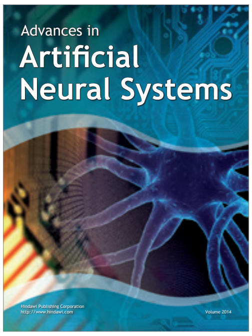

Journal of
Computer Networks
and Communications

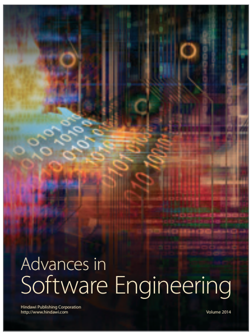

Advances in
Artificial
Intelligence


Advances in
Computer Engineering

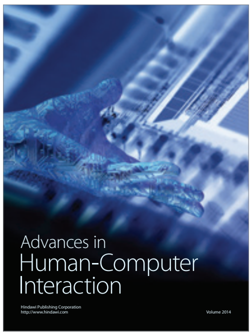
International Journal of
Computer Games
Technology

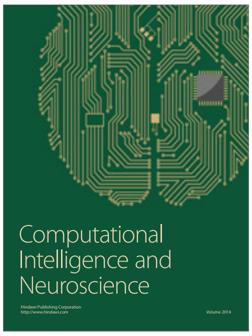

International Journal of
Biomedical Imaging


Advances in
Artificial
Neural Systems


Advances in
Software Engineering

Journal of
Robotics


Advances in
Human-Computer
Interaction


Computational
Intelligence and
Neuroscience

International Journal of
Reconfigurable
Computing

Journal of
Electrical and Computer
Engineering