Future Generation Computer Systems 86 (2018) 836–850

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms

Zhou Zhou [a,b], Jemal Abawajy [c,*], Morshed Chowdhury [d], Zhigang Hu [e], Keqin Li [f], Hongbing Cheng [g], Abdulhameed A. Alelaiwi [h], Fangmin Li [a]

[a] Department of Mathematics and Computer Science, Changsha University, Changsha, China
[b] Department of Computer Science, Hunan University, Changsha, China
[c] School of Information Technology, Deakin University, Geelong, Australia
[d] School of Information Technology, Deakin University, Melbourne, Australia
[e] School of Software, Central South University, Changsha, China
[f] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
[g] College of Computer, Zhejiang University of Technology, 310032, Hangzhou, China
[h] Software Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

HIGHLIGHTS

- Addressed the problem of reducing Cloud datacenter high energy consumption with minimal Service Level Agreement (SLA) violation.
- We propose two novel adaptive energy-aware algorithms for maximizing energy efficiency and minimizing SLA violation rate in Cloud datacenters.
- The proposed energy-aware algorithms take into account the application types as well as the CPU and memory resources during the deployment of VMs.
- We performed extensive experimental analysis using real-world workload.

ARTICLE INFO

Article history:
Received 4 November 2016
Received in revised form 22 June 2017
Accepted 21 July 2017
Available online 10 August 2017

Keywords:
Cloud computing
Energy efficiency
CPU intensive task
I/O intensive task
VM deployment
Service Level Agreement
Data center

ABSTRACT

In this paper, we address the problem of reducing Cloud datacenter high energy consumption with minimal Service Level Agreement (SLA) violation. Although there are many energy-aware resource management solutions for Cloud datacenters, existing approaches focus on minimizing energy consumption while ignoring the SLA violation at the time of virtual machine (VM) deployment. Also, they do not consider the types of application running in the VMs and thus may not really reduce energy consumption with minimal SLA violation under a variety of workloads. In this paper, we propose two novel adaptive energy-aware algorithms for maximizing energy efficiency and minimizing SLA violation rate in Cloud datacenters. Unlike the existing approaches, the proposed energy-aware algorithms take into account the application types as well as the CPU and memory resources during the deployment of VMs. To study the efficacy of the proposed approaches, we performed extensive experimental analysis using real-world workload, which comes from more than a thousand PlanetLab VMs. The experimental results show that, compared with the existing energy-saving techniques, the proposed approaches can effectively decrease the energy consumption in Cloud datacenters while maintaining low SLA violation.

1. Introduction

Cloud computing [1,2] has fundamentally transformed the way IT infrastructure is delivered to meet the IT needs of businesses and consumers. Generally, cloud computing delivery models are classified into software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) [3]. By allowing on-demand IT infrastructure provisioning model, Cloud computing enables organizations to automatically scale up and down IT resources usage based on their current and future needs. It also

* Corresponding author.

E-mail addresses: zhouzhou03201@126.com (Z. Zhou), jemal.abawajy@deakin.edu.au (J. Abawajy), morshed.chowdhury@deakin.edu.au (M. Chowdhury), zghu@csu.edu.cn (Z. Hu), lik@newpaltz.edu (K. Li), chenghb@zjut.edu.cn (H. Cheng), aalelaiwi@KSU.EDU.SA (A.A. Alelaiwi), lifangmin@whut.edu.cn (F. Li).

http://dx.doi.org/10.1016/j.future.2017.07.048
0167-739X/

enables great improvement in business or mission proficiencies without increasing in the corresponding resource (time, people or money) requirements. Moreover, by allowing pay-as-you-go service model, it eliminates high initial acquisition costs, maintenance costs and costs associated with licensing of software.

Although Cloud computing enables organizations to realize great benefits by minimizing operational and administrative costs, it suffers from the problem of high energy consumption that could negate its benefits [4]. For example, an average datacentre consumes energy as much as 25 000 households' energy consumptions [5]. Such high energy consumption can lead to an increasing Operational Cost (OC) and consequently reduce the Return on Investment (ROI). The high energy consumption, apart from the high OC and diminished ROI, results in much carbon dioxide (CO_2) emissions, which contributes to the global warming. Although advances in physical infrastructure have partly addressed the high energy consumption of datacentres issue, effective resource management is vital in further decreasing the high energy consumption of datacentres.

An important question is how to minimize datacentre energy consumption while ensuring the Quality of Service (QoS) delivered by the Cloud system. QoS is an important factor in Cloud environment and can be defined in the form of SLA (Service Level Agreement) [6,7]. The need to make datacentres efficient not only in regards to performance factors but also in both energy and emissions reduction have motivated a flurry of research recently. Although a remarkable improvements in the hardware infrastructure has enabled techniques to improve energy consumption, there are still lots of room for improvement. For instance, hosts in datacentres operate only at 10%–50% utilization for most of the time [8]. As the low utilization of hosts in datacentres results in huge amount of energy wastage, improving host utilization level in the datacentres can help decrease the energy consumption. However, naively improving host utilization level can affect the QoS delivered by the system. One way to effectively improve host utilization in Cloud datacentres is by using dynamic consolidation of VMs [9–11]. Dynamic VMs consolidation enables VMs to be reallocated periodically from overloaded hosts by utilizing VMs migration as well as minimize the number of hosts in datacentres by switching idle hosts to low-power mode to save energy consumption. Although dynamic consolidation of VMs have been shown to be NP-hard problem [12,13], it has been shown to be effective in minimizing energy consumptions [6].

Dynamic VMs consolidation generally involves the detection of overloaded and underloaded hosts in the datacenter (i.e., overloaded hosts detection), choose VMs to be reallocated from an overloaded host (i.e., VM selection) and select the receiving hosts for the VMs marked for relocation from overloaded host (i.e., VM deployment) [6]. Existing approaches only focus on one of the sub-components (i.e., host overload detection algorithm or VM selection algorithm or VM deployment algorithm). Moreover, they only consider the CPU in the dynamic VMs consolidation process and assume that the other system resources are not significant thus leading to wrong VM allocation. In addition, the previous works could not leverage the combination of energy efficiency (energy consumption and SLA violations) and placement of VM. For example, when VM is reallocated or migrated to another host, the existing algorithms only consider minimizing the energy consumption. But actually, SLA violation should also be considered during the process of VM reallocation and migration. Furthermore, previous approaches do not take into account the types of application running in the VMs and thus may not sufficiently reduce energy consumption of the Cloud datacentre and minimize SLA violation rate under a different workload.

In this paper, we propose a novel adaptive energy-aware algorithm *for maximizing energy efficiency and minimizing SLA violation rate in Cloud datacenters*. In order to adapt well to the dynamic and unpredictable workload commonly running in Cloud datacentres, the proposed energy-aware algorithm uses an adaptive three-threshold framework for the classification of Cloud datacentre hosts into four different classes (i.e., less loaded hosts, little loaded hosts, normally loaded hosts, and overloaded hosts). It also uses two VM selection policies for selecting VMs to migrate from an overloaded hosts. These methods consider both CPU and memory in the course of VM selection and deployment decision making. Finally, a VM deployment policy that leverages the combination of energy efficiency (energy consumption and SLA violations) and placement of VM is presented. It also takes into account both CPU and memory utilization during VM deployment. All in all, the main contributions of the paper can be summarized as follows:

(1) A framework that divides hosts in the datacentres, according to the different workload running on the hosts, into less loaded hosts, little loaded hosts, normally loaded hosts, and overloaded hosts.
(2) We put forward an adaptive three-threshold framework, which is different from the existing two-threshold framework. The adaptive three-threshold can adapt well to the dynamic and unpredictable workload commonly running in Cloud datacentres. A new algorithm, based on the adaptive three-threshold framework, for host state (e.g., overloaded) detection is presented.
(3) To handle the CPU intensive task and I/O intensive task, we present two VM selection methods from the overloaded hosts. The methods consider both CPU and memory utilization in decision making.
(4) We present VM selection methods for VM migration from the overloaded hosts. The methods consider both CPU and memory utilization in decision making.
(5) A new VM deployment policy that maximizes energy efficiency (i.e., energy consumption and SLA violation) is presented.
(6) We evaluate the algorithms proposed in this paper by using real-world workload, which comes from more than a thousand PlanetLab VMs hosted on 800 hosts located in more than 500 places across the world.

The rest of the paper is organized as follows: In Section 2, we present the related work. Adaptive three-threshold VM placement framework is proposed in Section 3. Experimental results and performance analysis are presented in Section 4. Section 5 concludes the paper.

2. Related work

The prior works concerning the energy consumption management in data centers can be broadly divided into four categories: dynamic performance scaling (DPS) [14–22], threshold-based heuristics [4,6,23–31], decision-making based on statistical analysis of historical data [32–35] and other methods [36–38]. In DPS [14–22], the system components are capable of dynamically adjusting their performance to save energy consumption. For example, the clock frequency of CPU can be gradually reduced or increased along with the change of the supply voltage. DPS can save substantial energy consumption when the resource is not fully utilized. A case in point of adopting DPS is DVFS (Dynamic Voltage and Frequency Scaling) technique. DVFS can be divided into three categories [14]: interval-based [15–18], inter-task [19,20] and intra-task methods [21,22]. Interval-based methods leverage the knowledge of past periods of CPU utilization to predict the utilization in the future, thus adjusting the processor frequency. Prior works [17,18] investigated the speed scaling methods in multi-core computer system and found that they were better

than the optimal offline algorithm. In contrast to interval-based methods, inter-task methods distinguish different tasks and assign them different speed of a processor (CPU). The method is simple and effective, especially when the workload is known in advance. However, it is not suitable for using the methods when workload is irregular. Different from inter-task methods, intra-task methods utilize the knowledge of structure of programs, consequently adjusting the processor frequency within the tasks to save energy consumption. Existing DSP-based works depend on using DVFS technique to save energy consumption. Although the method has improved consumption of energy, the overall datacenter energy consumption remain high.

Threshold-based heuristics [4,6,23–31] improve the utilization rate of resources by setting threshold, thus reducing the energy consumption and meeting the QoS delivered by cloud system. For example, Buyya et al. [23] proposed a method named Single Threshold (ST) algorithm. ST sets a utilization threshold to keep all hosts' CPU utilization in data centers below this value, consequently control the migration of VM. ST algorithms can save more energy compared to DVFS algorithms. Later, Beloglazov and Buyya [24] presented Double Thresholds (DT) algorithm, which sets two thresholds (one is low utilization threshold and the other is upper utilization threshold) to control all hosts' CPU utilization between the two thresholds. When some hosts' CPU utilization does not fall into these two thresholds, these hosts should be migrated to other hosts to reduce energy consumption and SLA violations. Beloglazov and Buyya [25] put forward a decentralized architecture of energy saving management system composed of three stages for optimizing VM placement, selection of VMs for migration, finding new hosts to accommodate the VMs, and optimizing the virtual network topologies. Beloglazov and Buyya [4] proposed an energy-saving allocation heuristics to improve the energy efficiency. They also presented three VMs selection policies: (1) minimization of migrations approach; (2) the highest potential growth approach; and (3) the random choice approach. Beloglazov and Buyya [6] focused on the overloaded host detection. For any known stationary workload, authors utilized maximizing the mean intermigration time to tackle it based on Markov chain model. As for the unknown non-stationary workloads, authors used the Multisize Sliding Window workload estimation method to address it.

Li et al. [26] designed a new VM placement algorithm called Modified Particle Swarm Optimization (MPSO) based on DT with multi-resource utilization. The MPSO approach avoids local optimization. Experimental results show that MPSO decreases both the number of active hosts and VMs migration. Fu and Zhou [27] investigated the problem of energy efficiency in data centers and put forward a novel VM selection method based on DT. Their algorithm considers the CPU utilization as well as the degree of resource satisfaction. Zhu et al. [28] explored the problem of VMs consolidation and set a static upper threshold of 85% for CPU utilization. The threshold value of 85% is mainly used to judge whether a host is overloaded. The static upper threshold of 85% for CPU utilization is put forward and justified for the first time in [29]. Similarly, VDPM (VMware Distributed Power Management) set a static utilization threshold of 81% [30] for CPU utilization. Jung et al. [14] studied the problem of consolidation of VMs, which run multitier web application to optimize the global utility function, while ensuring the QoS. In the paper, the QoS is defined by response time for computing each transaction type of the applications. The method proposed in the paper can yield significant saving in power consumption despite the workload is specific. In our previous study [31], we proposed a static three-threshold algorithm for VM placement, and justified the optimal threshold interval is 40% considering energy efficiency (energy consumption and SLA violations). All these works based on threshold heuristics can bring substantial saving in energy consumption. However, they are not suitable for variable and unknown workload.

To further save on energy consumption, decision-making based on statistical analysis of historical data [32–35] is an effective method to improve the energy efficiency in data centers. Specifically, authors in [32] developed an adaptive double-threshold algorithm based on characteristics of the distribution over some recent period. Although the method is promising, the effect of saving energy consumption in data centers still need to be improved. Later, authors [33] put forward some modification to their adaptive double-threshold algorithms by using the historical data of CPU utilization. Guenter et al. [34] leveraged the weighted linear regression to predict the future workload and optimize the resource allocation. The method is similar to Local Regression (LR) proposed in [33]. In our previous study [35], we investigated the problem of VM placement to improve the energy efficiency in data centers. However, the approach proposed in the paper did not consider the workload type such as I/O bound tasks. Moreover, during the process of VM placement, the approach proposed in the paper only considers minimizing energy consumption, while ignoring the energy efficiency (energy consumption and SLA violations). All these works based on statistical analysis of historical data can bring considerable saving in energy consumption. However, they did not consider the workload type and only consider minimizing the energy consumption during the allocation and placement of VMs. However, another factor such as SLA violations should also be considered.

Other methods [36–38] include optimizing energy consumption of disk subsystems [36,37], and data management system [38] to reduce energy consumption in data centers, while ensuring the QoS delivered by cloud system. Specifically, to deal with the missing data blocks led to the problem of performance degradation, Higai et al. [36] put forward two replica reconstruction approaches to balance the workloads of replication processes. For the problem of read-performance of duplicate, Mao et al. [37] proposed a new method to build a special data region by using the SSD (solid-state drive), aiming at improving the read performance of the deduplication-based storage systems in the cloud. A VM scheduling algorithm based on datacenter thermal model that considers the temperature distribution of airflow and server CPU is discussed in [39]. An algorithm for VM consolidation with multiple usage prediction based on historical data of the long-term utilization of datacenter resources is discussed in [40].

3. Adaptive three-threshold VM placement framework

In this section, we present the proposed resource management algorithm along with its components for detecting and relieving overloaded hosts by reallocating some Virtual Machines (VMs), detecting underloaded hosts and perform consolidation, and allocating the VMs selected for relocation to other hosts in the datacenter in a holistic manner. The main notations and their meanings used in throughout the paper are listed in Table 1.

3.1. Adaptive three-threshold framework

To solve the problem of high energy consumption, we divide the hosts into classes based on the CPU utilization. As shown in Fig. 1, we set three thresholds of CPU utilization, T_a, T_b, and T_c ($0 \leq T_a < T_b < T_c \leq 1$), resulting in the hosts of a data center divided into four classes: *less loaded hosts, little loaded hosts, normally loaded hosts, and overloaded hosts*. The CPU utilization of the less loaded hosts is between "0" and "T_a"; the CPU utilization of

Table 1
Main symbols and their meaning.

Symbol	Meaning
T_a, T_b, T_c	Threshold values
U_i	The CPU utilization of host i
M	The number of hosts
$D = \{U_1, U_2, U_3, \ldots, U_n\}$	CPU utilization at different time
k	The number of cluster
$R_i\ (1 \leqslant i \leqslant k)$	Number of the i cluster
$MR = \{MR_1, MR_2, \ldots, MR_k\}$	The set of midrange
IQR	Interquartile range of the MR
TQ_3	The third quartiles of MR
TQ_1	The first quartile of MR
d	Denotes the level of VMS consolidation
$C^i_{VM}\ (i = 1, 2, \ldots, S_{VM})$	The CPU utilization of the VM i
S_{VM}	The set of VMs of a host
$M^i_{VM}\ (i = 1, 2, \ldots, S_{VM})$	The memory utilization of the VM i

the little loaded hosts is between "T_a" and "T_b"; the CPU utilization of the normally loaded hosts is between "T_b" and "T_c"; and the CPU utilization of the overloaded hosts is between "T_c" and "1".

Fig. 2 shows the adaptive three-threshold framework (ATF). The main idea of the ATF is as follows: for every host in the data center, ATF firstly obtains its CPU utilization (U_i). It then classifies the host as follows: (1) if $U_i \geqslant T_c$, the host is considered to be an overloaded host. To avoid the high SLA violations, some VMs on the host should be migrated to a little loaded host with the highest energy efficiency; (2) if $T_b \leq U_i < T_c$, the host is considered to be a normally loaded host. Since excessive VMs migration results in much more energy consumption and SLA violations [41], all VMs on the host are kept unchanged; (3) if $T_a \leq U_i < T_b$, the host is considered to be a little loaded host. Based on the reason that excessive VMs migration leads to much more energy consumption and SLA violations [41], all VMs on the host are kept unchanged; (4) if $U_i < T_a$, based on the reason of saving power, all VMs on the host are migrated to a little loaded host with the highest energy efficiency, the idle host are switched to low-power mode, thus reducing the static energy consumption. When the demand increases, the host can be reactivated within a short time to process tasks.

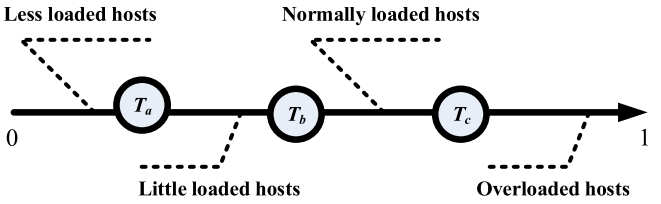


Fig. 1. Classification of the host.

However, there are two problems that should be addressed in ATF. The first one is how to determine the value of the three thresholds (T_a, T_b, and T_c), and the issue will be handled in Section 3.2. The second one is how to select VMs from the overloaded hosts, especially for processing the CPU intensive task or I/O intensive task, and the issue will be addressed in Section 3.3.

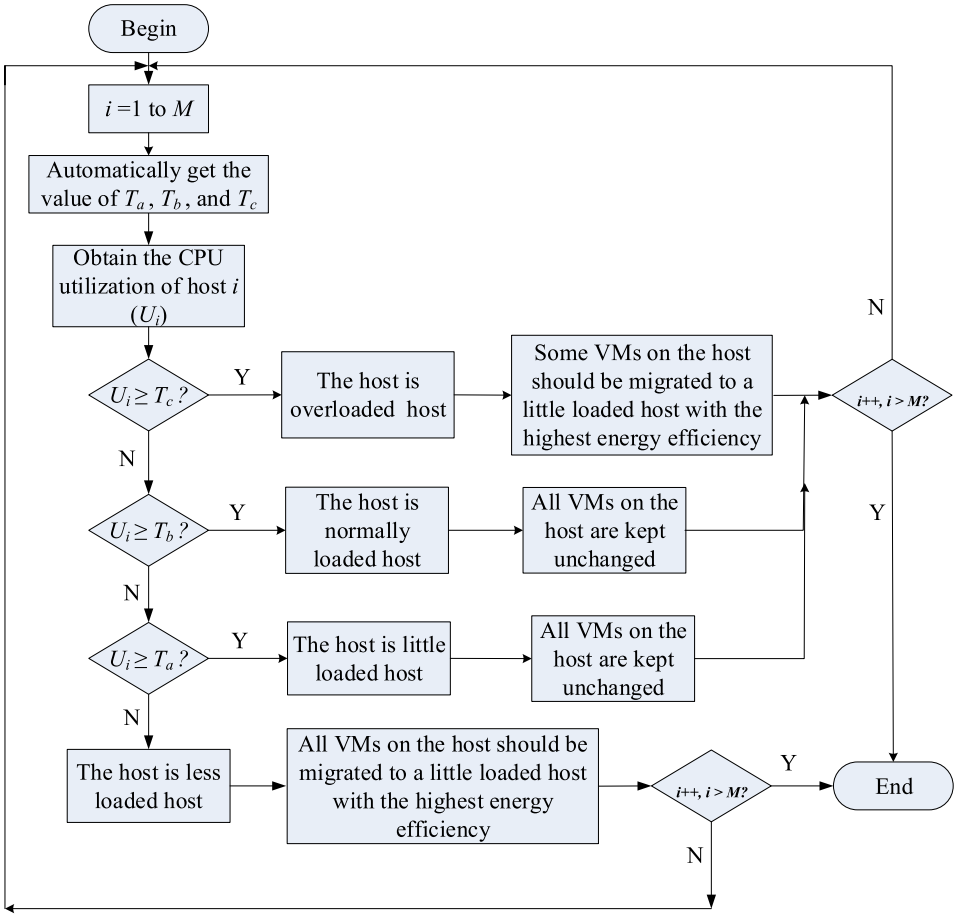


Fig. 2. Adaptive three-threshold framework (ATF).

3.2. Determining threshold values

To determine the value of the three thresholds (T_a, T_b, and T_c), we use a novel algorithm which refer to as a KMI (K-Means clustering algorithm-Midrange-Interquartile range). Let $H = \{h_1, h_2, \ldots, h_M\}$ be a set of M hosts in the Cloud datacenter. Let $D = \{U_1, U_2, \ldots, U_n\}$ be a dataset such that $U_t \in D\,(1 \leq t \leq n)$ is the CPU utilization of a host $h_f \in H$ at time t. Algorithm 1 shows the pseudo-code of KMI.

Algorithm 1: KMI

Input: D, k, d

Output: T_a, T_b, and T_c

```
R = Cluster (D, k)        // K-Means clustering algorithm
for i = 1 to R.length do
        MR_i = (Max(R_i) + Min(R_i))/2
        MR [i] = MR_i
end for
IQR = TQ_3 (MR) - TQ_1 (MR)
Define T_a, T_b, and T_c      // According to Equation (3) – (5)
```

END

In the algorithm, the function Cluster (D, k) is a K-Means clustering algorithm that partitions D into k clusters: $R_1, R_2, \ldots, R_k$ such that:

- $R_i = \{U_{P_{i-1}+1}, U_{P_{i-1}+2}, \ldots, U_{P_i}\} \in D$,
- $0 = P_0 < P_1 < P_2 < P_3 < P_4 < \ldots < P_i = n$, and
- $R_i \cap R_j = \Phi$ for $1 \leqslant i, j \leqslant k$ (the value of k can be set by using empirical approach).

For each cluster R_i, the algorithm obtains the midrange as follows:

$$MR_i = \frac{Max\,(R_i) + Min\,(R_i)}{2}, \quad 1 \leq i \leq k. \tag{1}$$

In Eq. (1), the parameter $Max\,(R_i)$ refers to the maximum value of the R_i cluster and the parameter $Min\,(R_i)$ is the minimum value of the R_i cluster. The function of Eq. (1) is to obtain the midrange value (MR_i) of the i cluster. This step will produce $MR = \{MR_1, MR_2, MR_3, \ldots, MR_i, \ldots, MR_k\}$. The algorithm then obtains the Interquartile Range (IQR) of the set $MR = \{MR_1, MR_2, MR_3, \ldots, MR_i, \ldots, MR_k\}$ as follows:

$$IQR = TQ_3 - TQ_1 \tag{2}$$

where parameter TQ_3 is the third quartiles of MR while parameter TQ_1 refers to the first quartile of MR. The aim of Eq. (2) is to get IQR of the set MR. The adaptive three thresholds (T_a, T_b, and T_c) in ATF are defined as follows:

$$T_c = (1 - d \cdot IQR) \tag{3}$$

$$T_b = \frac{9}{10}\,(1 - d \cdot IQR) \tag{4}$$

$$T_a = \frac{5}{10}\,(1 - d \cdot IQR) \tag{5}$$

where the parameter d corresponds to how aggressively the system consolidates VMs. For instance, the smaller the value of parameter d the less the energy consumption and high SLA violations, and vice versa. The value of the parameter d is addressed in Sections 4.6.2 and 4.6.3. By defining Eqs. (3)–(5), the adaptive three thresholds (T_a, T_b, and T_c) can be obtained. Therefore, the three thresholds (T_a, T_b, and T_c) have been determined by using KMI algorithm. The complexity of KMI is $O(k \times n \times t)$, parameter k is the number of cluster; parameter n is the data size and parameter t is the number of iterations.

3.3. VM selection methods from overloaded hosts

We now present our approach for selecting VMs from the overloaded hosts for migration. We assume that the workload is either CPU intensive (i.e., CPU-bound tasks) or I/O intensive (i.e., I/O bound tasks). An application is considered CPU intensive when the time for it to complete a task is determined principally by the speed of the central processor. Different from the CPU intensive task, the I/O intensive refers to a condition in which the time it takes to complete a computation is determined principally by the period spent waiting for input/output operations to be completed. In other words, more time is spent requesting (waiting) for data than processing it. In this case, the computer component such as both CPU and memory consume substantial energy consumption.

Let S_{VM} be the set of VMs currently running in a given overloaded host. Let U^i_{VM} and M^i_{VM} represent the CPU and memory utilization of i VM respectively, where $1 \leq i \leq S_{VM}$. Let C^e_{VM} *and* M^e_{VM} refer to CPU and memory utilization of any VM e ($1 \leq e \leq S_{VM}$ *and* $e \neq i$), respectively.

MRCU method: We propose a new VMs selection method named MRCU (Maximum ratio of CPU utilization to memory utilization) to select VMs for migration when a host is overloaded by CPU intensive tasks. The MRCU method selects a VM v for migration from the host that meets the following condition:

$$\frac{C^v_{VM}}{M^v_{VM}} > \frac{C^e_{VM}}{M^e_{VM}}, \quad v \in S_{VM} \text{ and } \forall e \in S_{VM} \text{ and } v \neq e. \tag{6}$$

For the case where the host is overloaded by CPU intensive tasks, the energy consumption of CPU accounts for the most part of the total energy consumption relative to other components (such as memory). Eq. (6) shows that the higher the C^v_{VM} value and the lower M^v_{VM} value, the higher the $\frac{C^v_{VM}}{M^v_{VM}}$ value is. Therefore, Eq. (6) selects a VM with the highest value of $\frac{C^v_{VM}}{M^v_{VM}}$ to migrate, because higher CPU utilization means consuming more energy, while lower memory utilization means less energy consumption of by VM migration. MRCU method considers both CPU factor and memory factor during migrating potential VMs.

MPCU method: When a host is overloaded by I/O intensive task, to reduce the potential SLA violations, we propose a novel VMs selection method named MPCU (Minimum the product of a CPU utilization and a memory utilization) to deal with it. The MPCU method selects a VM v that meets the following condition for migration to another host:

$$\left(C^v_{VM} \times M^v_{VM}\right) < \left(C^e_{VM} \times M^e_{VM}\right) \mid v \in S_{VM} \text{ and } \forall e \in S_{VM} \text{ and } v \neq e. \tag{7}$$

For the host is overloaded by I/O intensive task, both CPU and memory consume large amounts of energy consumption, that is to say CPU factor and memory factor are equally important. Eq. (7) selects the VM with the minimum product of CPU utilization and memory utilization to migrate, because the minimum product of CPU utilization and memory utilization means the less energy consumption of migration, the corresponding performance degradation caused by VM migration will decrease, thus reducing the potential SLA violations. MPCU method considers both CPU factor and memory factor during migrating potential VMs.

3.4. VM placement with maximizing energy efficiency

In this subsection, we describe a new VM placement policy that takes into account both the least increase of energy consumption and SLA violation. We refer to this policy as VM placement with maximizing energy efficiency (VPME). Algorithm 2 shows the pseudo-code of VPME. In the algorithm, the "vmlist" is the set of VMs while the "hostlist" refers to the set of hosts in the datacentre. The parameters T_a, T_b, and T_c are the three thresholds previously discussed. We also define a variable "minEnergyEfficiency" and assign a minimum value to it.

VPME firstly sorts all VMs in decreasing order according to their CPU utilization (Line 1). Then for every host in the datacenter (Line 5), we check whether the current host meets requirement of the VM such as available CPU capacities and memory size (Line 6). If it is, we get the host's CPU utilization after an allocation of the VM (see variable CpuUtilize in Line 7). Line 8 is to keep the host with little load. Line 9–Line 11 is to get the power difference before and after an allocation of the VM. Line 12–Line 14 determine the first SLA violations difference before and after an allocation of the VM (for the definition of the first SLA violation see Eq. (8) in Section 4.4).

Line 15–Line 17 refer to the second SLA violations difference before and after an allocation of the VM (for the definition of the second SLA violation sees Eq. (9) in Section 4.4). Line 18 is to define the SLA violation (the definition of the SLA violation sees in Section 4.4), while Line 19 means to define the energy efficiency (the definition sees Eq. (11) in Section 4.5). Line 20–Line 23 find a host with the highest energy efficiency due to the allocation of a VM, while Line 25 is to complete the allocation of the VM. The complexity of VPME is $O(M \times N)$, variable M is the number of hosts, while variable N refers to the number of VMs.

3.5. VM placement optimization

VM placement optimization is also very important to improve the utilization of resources in data centers, thus reducing the energy consumption and SLA violation delivered by cloud system. The pseudo-code of VM placement algorithm is shown in algorithm 3. The input to the algorithm are the three thresholds (T_a, T_b, and T_c), the VM and host lists. The algorithm first obtains the CPU utilization of the hosts (Line 2) and determines if a given host is overloaded. Line 3–Line 6 choose VMs through using VM selection policy (see Section 3.3) from the host, and migrates the selected VMs to another host with the highest energy efficiency. Line 7–Line 10 refer to all VMs on normally loaded host or little loaded host keeping unchanged. Line 11–Line 16 are to select all VMs on the less loaded host, and migrate them to another host with the highest energy efficiency. The complexity of Algorithm 3 is $O(M)$, variable M is the number of hosts.

4. Experimental results and performance analysis

In this section, after presenting some basic definitions (including energy consumption model, SLA violation metrics, and energy efficiency metric), we will focus on the experimental results and performance analysis. The experiment in the paper includes three parts: (1) the first part only evaluates the performance of VPME algorithm (see Section 3.4), which places VM with the aims of maximizing energy efficiency. However, the existing algorithms only place VM with minimizing energy consumption. Since VM placement is very important and is a part of an energy-aware algorithm, we will discuss it in Section 4.6.1; (2) the second part is to evaluate the energy-aware algorithm proposed in this paper for processing CPU intensive task, we will deal with it in Section 4.6.2; (3) the third part is to evaluate the energy-aware algorithm proposed in this paper for processing I/O intensive task, we will address it in Section 4.6.3. Each experiment was repeated at least 10 times and the average of the 10 results are reported.

Table 2
Configuration of hosts.

Hosts	CPU type	Frequency (GHz)	Cores	RAM (GB)
HP Proliant G4	Intel Xeon 3040	1.86	2	4
HP Proliant G5	Intel Xeon 3075	2.66	2	4

Table 3
Four kinds of VM types.

VM type	CPU (MIPS)	RAM (GB)
High-CPU medium instance	2500	0.85
Extra large instance	2000	3.75
Small instance	1000	1.70
Micro instance	500	0.61

Table 4
Characteristics of the workload data (CPU utilization).

Date	Number of VMs	Mean	Quartile 1	Quartile 3
03/March/2011	1052	12.31%	2%	15%
06/March/2011	898	11.44%	2%	13%
09/March/2011	1061	10.70%	2%	13%
22/March/2011	1516	9.26%	2%	12%
25/March/2011	1078	10.56%	2%	14%
03/April/2011	1463	12.39%	2%	17%
09/April/2011	1358	11.12%	2%	15%
11/April/2011	1233	11.56%	2%	16%
12/April/2011	1054	11.54%	2%	16%
20/April/2011	1033	10.43%	2%	12%

4.1. Experiment setup

It is essential to evaluate and compare the proposed algorithms on a large-scale virtualized data center infrastructure. Considering the repeatability of experiments and the advantages of the CloudSim toolkit [42] such as allowing the modeling the virtualized environments and supporting on-demand resource provisioning, the CloudSim toolkit is chosen to implement and analyze the proposed algorithm. We simulate a data center which includes 800 physical hosts, half of which is HP Proliant G4, the other half is HP Proliant G5. The specific parameters of these hosts are listed in Table 2. Characteristics of the VMs correspond to Amazon EC2 VM types [43], which is depicted in Table 3. There are four kinds of VMs in Table 3, that is High-CPU Medium Instance, Extra Large Instance, Small Instance, and Micro Instance.

4.2. Workload data

It is necessary and important to do experiments using real workload data. In our experiment, we use the workload derived from a CoMon project. The function of CoMon is to monitor infrastructure for PlanetLab [44]. We use the data from more than a thousand VMs' CPU utilization and the VMs were placed at more than 500 places throughout the world. Table 4 depicts the characteristics of the data [33].

4.3. Energy consumption model

Recent studied [45,46] show that the energy is consumed by hosts in datacentres related to its CPU and memory utilization, even when the DVFS technique is applied. However, with the equipment of multicore CPUs, large-capacity memory, and big hard disk, the traditional linear model is not capable of depicting the energy consumption of a host accurately. To accurately describe the energy consumption, we use the real data of energy consumption, all of which derive from SPECpower benchmark (http://www.spec.org/powerssj2008/). We have selected two hosts equipped with dual-core CPU, one of which is named HP Proliant G4 with 1.86 GHz (dual-core), 4 GB RAM, the other is HP Proliant G5 with 2.66 GHz (dual-core), 4 GB RAM. The details of energy consumption for the two hosts under different load is illustrated in Table 5 [33].

Algorithm 2. The VPME algorithm

Input: vmlist, hostlist, T_a, T_b, and T_c

Output: allocation of VMs

```
1:  vmlist.sortByDecreasingUtilization( )
2:  for (vm : vmlist) do
3:      minEnergyEfficiency = minimum
4:      allocatedHost = null
5:      for (host: hostlist) do
6:          if (host meets the requirements of vm) then
7:              CpuUtilize = getUtilizationAfterVm (host, vm)
8:              if (Ta ≤ CpuUtilize ≤ Tb ) then
9:                  power = host.getPower ()
10:                 powerAfterAllocation = getPowerAfterAllocation (host, vm)
11:                 powerDiff = powerAfterAllocation - power
12:                 firstSlaBefore = getSlaTimePerActiveHost (host)
13:                 firstSlaAfterVM = getSlaTimeAfterAllocation (host, vm)
14:                 firstSla = firstSlaAfterVM - firstSlaBefore
15:                 secondSlaBefore = getSlaMetric (host.getVmList ())
16:                 secondSlaAfterVm = getSlaMetricAfterAllocation (host.getVmList (), vm)
17:                 secondSla= secondSlaAfterVm - secondSlaBefore
18:                 SLA = firstSla × secondSla
19:                 EnergyEfficiency = 1 / ( powerDiff × SLA)
20:                 if (EnergyEfficiency > minEnergyEfficiency) then
21:                     minEnergyEfficiency = EnergyEfficiency
22:                     allocatedHost = host
23:                 end if
24:                 if (allocatedHost ≠ null) then
25:                     allocate the VM to host
26:                 end if
27:             end if
28:         end if
29:     end for
30  end for
END
```

4.4. SLA violations metrics

SLA violations is an important factor for any VMs deployment. There are two basic methods to depict an SLA violations [33]:

SVTAH (SLA Violation Time per Active Host): SVTAH represents that the SLA violation time accounts for the total time of active host. The SLA violation time means the CPU utilization of active host has reached 100% during the time. All in all, the SVTAH can be defined as follows:

$$SVTAH = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{V_i}}{T_{a_i}} \tag{8}$$

where parameter M is the number of hosts in a data center; parameter T_{V_i} represents total time during which the CPU utilization of host i has reached 100%, parameter T_{a_i} means the total time of host i being in active state (that is to say host i provides service to its VMs). Eq. (8) illustrates that when a host's CPU utilization has reached 100%, the VMs belong to the host cannot get the requested CPU capacity, thus resulting in the SLA violations.

Table 5
Energy consumption at different CPU utilization.

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

PDCVM (Performance Degradation Caused by VM Migration): VM migration can bring negative impact on the performance of

Algorithm 3. The optimization of VM placement

Input: T_a, T_b, and T_c , hostlist

Output: migrationMap

```
for (host: hostlist) do
    utilization = host.getCpuUtilization ()
    if (utilization ⩾ T_c) then
        overUtilizedHosts = getOverUtilizedHosts ()
        vmsToMigrate = getVmsToMigrateFromHosts (overUtilizedHosts)
        migrationMap = getNewVmPlacement (vmsToMigrate)
    else if (utilization < T_a) then
        lowUtilizedHosts = getLowUtilizedHost ()
        vmsToMigrateB = getVmsToMigrateFromLowHosts (lowUtilizedHosts)
        migrationMapB = getNewVmPlacement (vmsToMigrateB)
        migrationMap.addAll (migrationMapB)
    end if
end for
return migrationMap
END
```

application that runs on a VM with a short time. The performance degradation caused by VM j can be defined as follows:

$$PDCVM = \frac{1}{N}\sum_{i=1}^{N}\frac{P_{d_j}}{p_{r_j}} \tag{9}$$

where parameter N corresponds to the number of VMs; parameter P_{d_j} represents the estimate of the performance degradation caused by VM j; parameter p_{r_j} means that the total CPU capacity requested by VM j during its lifetime. Since both SVTAH and PDCVM metrics are of equal important to independently measure the SLA violations, the SLA violation can be defined as follows:

$$SLA = SVTAH \times PDCVM. \tag{10}$$

4.5. Energy efficiency metric

Energy efficiency metric includes two aspects: energy consumption and SLA violation. Based on the previous definition, the energy efficiency (EE) can be defined as follows:

$$EE = \frac{1}{p_{power} \times SLA} \tag{11}$$

where parameter p_{power} represents the energy consumption, while parameter SLA means the SLA violations. Eq. (11) illustrates that the higher the EE value, the more the energy efficiency.

4.6. Discussion and analysis

4.6.1. The evaluation and analysis of VPME

Since VM placement is very important and is a part of an energy-aware algorithm, this part we will evaluate the performance of VPME algorithm (see Section 3.4). Different from the existing algorithms only place VM with minimizing energy consumption, VPME places VM with maximizing energy efficiency (including both energy consumption and SLA violation). The selected data set is "03/March/2011" (see Table 4 in Section 4.2), and the comparison algorithms are KAM-MMS-2.0 [35] and KAI-MMS-1.0 [35]. As both KAM-MMS-2.0 and KAI-MMS-1.0 place VM aiming at minimizing energy consumption, we put the combination of KAM-MMS-2.0 and VPME denoted by KAM-MMS-2.0-EE (KAM-MMS-2.0-EE = KAM-MMS-2.0 + VPME); that is to say, KAM-MMS-2.0-EE places VM aiming at maximizing energy efficiency, while KAM-MMS-2.0 places VM aiming at minimizing energy consumption. By the same method, we put the combination of KAI-MMS-1.0 and VPME denoted by KAI-MMS-1.0-EE (KAI-MMS-1.0-EE = KAI-MMS-1.0 + VPME).

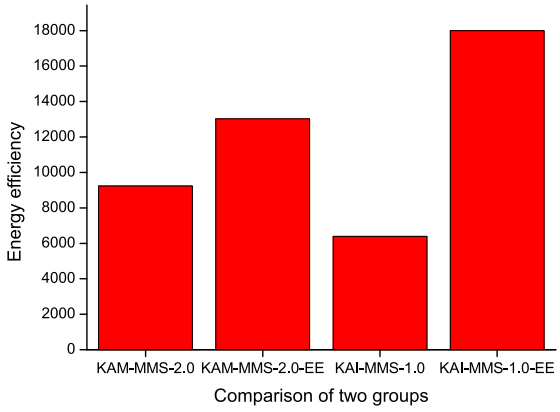


Fig. 3. The comparison of energy efficiency.

Figs. 3 and 4 show the relative performance of the VM placement algorithm with respect to energy efficiency and energy consumption respectively. Fig. 3 shows that KAM-MMS-2.0-EE is better than KAM-MMS-2.0. This can be explained by the fact that KAM-MMS-2.0-EE optimizes the deployment of VM to maximize the energy efficiency, while KAM-MMS-2.0 considers only minimizing the energy consumption and thus ignoring the SLA violation during the deployment of VMs. For KAI-MMS-1.0-EE and KAI-MMS-1.0, KAI-MMS-1.0-EE has a better performance than KAI-MMS-1.0 for the same reason as above.

Figs. 4–6 display energy consumption, SLA violations and SVTAH. The results show that KAM-MMS-2.0-EE has better performance than KAM-MMS-2.0. The reason is that KAM-MMS-2.0-EE

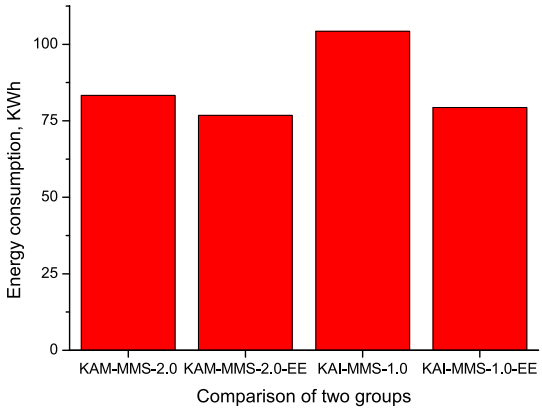


Fig. 4. The comparison of energy consumption.

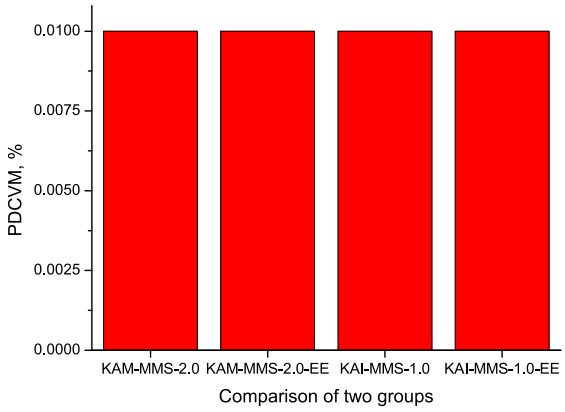


Fig. 7. The comparison of PDCVM.

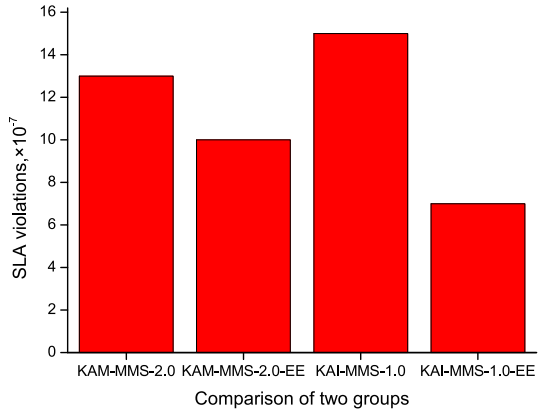


Fig. 5. The comparison of SLA violations.

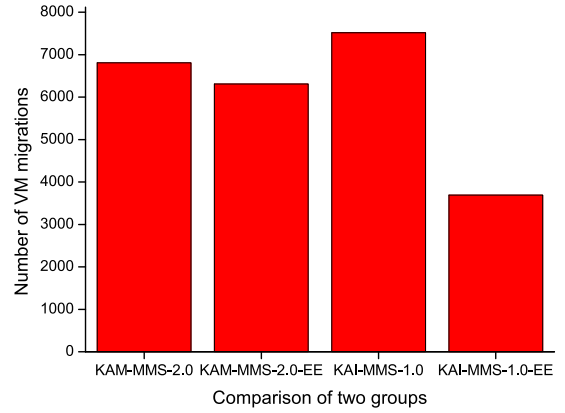


Fig. 8. The number of VM migrations.

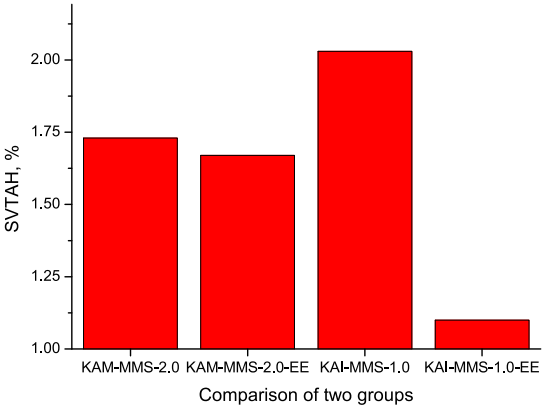


Fig. 6. The comparison of SVTAH.

improves the efficiency of VM migration, decreasing the number of "thrashing of VM migration" ("thrashing of VM migration" means a VM frequently migrated from a host to another host), thus reducing the energy consumption, SLA violations, and SVTAH. By the same reason, in terms of energy consumption, SLA violations and SVTAH, KAI-MMS-1.0-EE has better performance than KAI-MMS-1.0.

Figs. 7 and 8 show the relative performance of the VM placement algorithm with respect to PDCVM and the number of VM migrations respectively. Regarding PDCVM, Fig. 7 show that the two group algorithms (KAM-MMS-2.0-EE and KAM-MMS-2.0, KAI-MMS-1.0-EE and KAI-MMS-1.0) lead to the same result. As for the number of VM migration in Fig. 8, the results show that KAM-MMS-2.0-EE has better performance than KAM-MMS-2.0. The reason is that KAM-MMS-2.0-EE improves the efficiency of VM migration, thus reducing the number of VM migration. In terms of the number of VM migration, KAI-MMS-1.0-EE is better than KAI-MMS-1.0 since KAI-MMS-1.0-EE improves the efficiency of VM migration, thus reducing the number of VM migration.

Regardless of the energy efficiency, energy consumption, SLA violations, SVTAH, and the number of VM migration, Figs. 3–8 show that the algorithm with maximizing the energy efficiency has the better performance than the algorithm with minimizing the energy consumption.

4.6.2. Analysis of CPU intensive task

Considering the characteristics of data set in Table 4 in Section 4.2, we choose "03/March/2011" data set as the CPU intensive task. The comparison algorithms are NPA (None Power Aware) [42], DVFS [15], THR-MMT-1.0 [33], THR-MMT-0.8 [33], MAD-MMT-2.5 [33], IQRMMT-1.5 [33], KAM-MMS-2.0 [35], and KAI-MMS-1.0 [35]. Before making a comparison among these energy-aware algorithms, we need to determine the value of parameter d for KMI (see Section 3.2) to process the CPU intensive task. How to determine the value of parameter d? We put the combination of ATF (see Section 3.1), KMI (see Section 3.2), MRCU (see Section 3.3), VPME (see Section 3.4), and parameter d denoted by KMI-MRCU-d . By the same reason, we put the combination of ATF (see Section 3.1), KMI (see Section 3.2), MPCU (see Section 3.3), VPME (see Section 3.4), and parameter d denoted by KMI-MPCU-d. KMI-MRCU-d is for CPU intensive task, while KMI-MPCU-d is for I/O

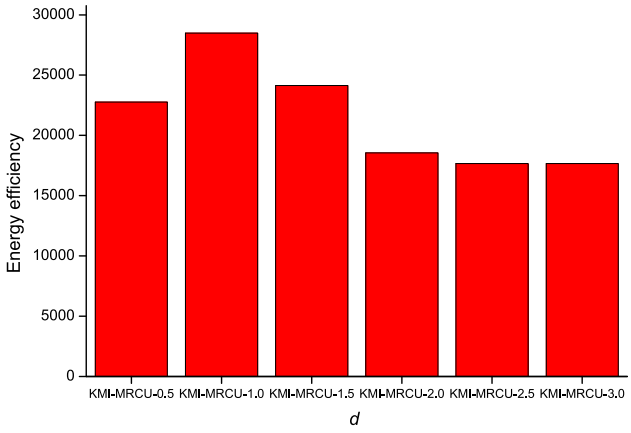


Fig. 9. Energy efficiency under different value of *d*.

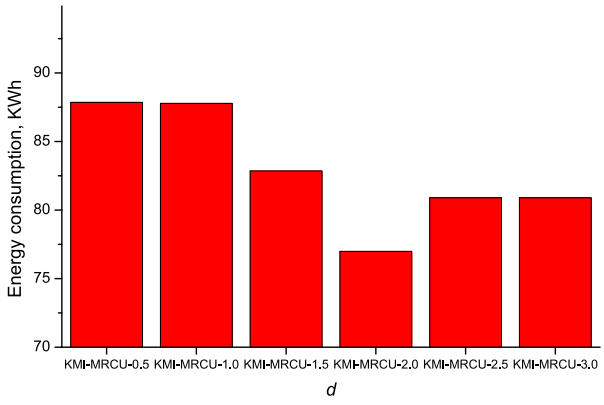


Fig. 10. Energy consumption under different value of *d*.

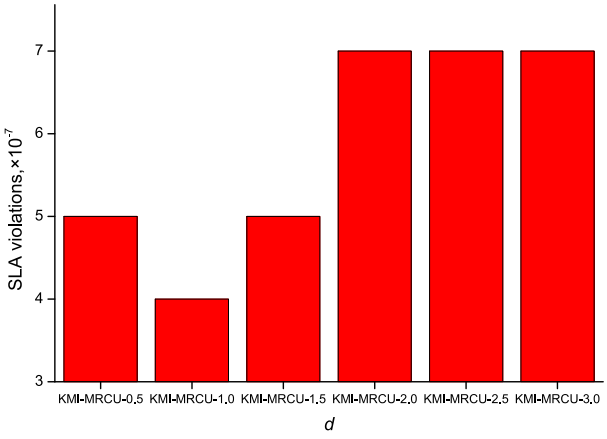


Fig. 11. SLA violations under different value of *d*.

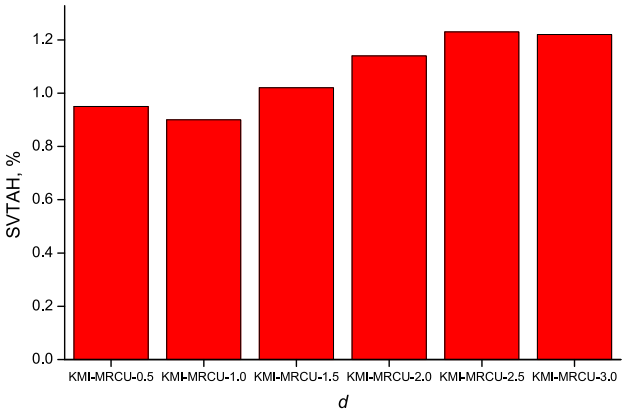


Fig. 12. SVTAH under different value of *d*.

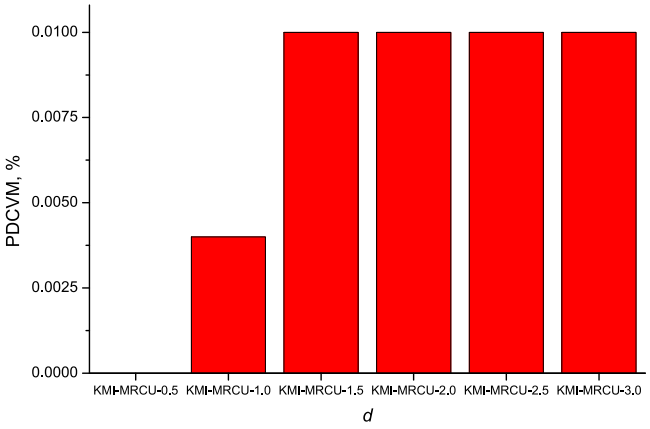


Fig. 13. PDCVM under different value of *d*.

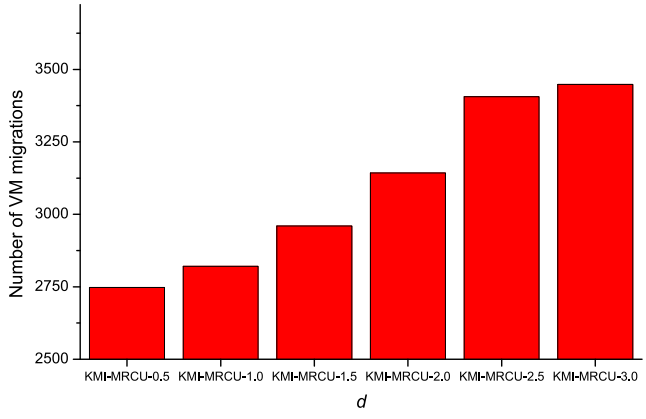


Fig. 14. Number of VM migration under different value of *d*.

intensive task (the evaluation of KMI-MPCU-*d* will be presented in Section 4.6.3). In the experiments, we varied the value of parameter *d* for KMI-MRCU-*d* from 0.5 to 3.0 increased by 0.5. Figs. 9–14 show the experimental results under different value of *d*.

Figs. 9–14 illustrate the energy efficiency (Eq. (11) in Section 4.5), energy consumption, SLA violations (Eq. (10) in Section 4.4), SVTAH (Eq. (8) in Section 4.4), PDCVM (Eq. (9) in Section 4.4) and number of VM migrations under different value of *d*, respectively. In terms of energy consumption in Fig. 10, the minimum value is obtained when parameter $d = 2.0$. However, Fig. 11 shows that a minimum value of SLA violations is obtained when parameter $d = 1.0$. Considering both energy consumption and SLA violations (that is energy efficiency), the optimum value for energy efficiency can be got when parameter $d = 1.0$ (see Fig. 9). Therefore, we use KMI-MRCU-1.0 to make a comparison with other energy-aware algorithms. Table 6 shows the experimental results compared with the other baseline algorithms.

Table 6 illustrates the energy efficiency, energy consumption, SLA violations, SVTAH, PDCVM, and number of VM migrations for

Table 6
Comparison with other energy-aware algorithms.

Algorithms	Energy efficiency	Energy consumption (kWh)	SLA violations (10^{-7})	SVTAH (%)	PDCVM (%)	Number of VM migrations
NPA	–	2410.8	–	–	–	–
DVFS	–	803.91	–	–	–	–
THR-MMT-1.0	38	99.95	2613	26.97	0.10	19 852
THR-MMT-0.8	170	119.40	492	4.99	0.10	26 567
MAD-MMT-2.5	169	114.27	518	5.24	0.10	25 923
IQR-MMT-1.5	166	117.08	514	5.08	0.10	26 420
KAM-MMS-2.0	9 231	83.33	13	1.73	0.01	6 808
KAI-MMS-1.0	6 393	104.28	15	2.03	0.01	7 519
KMI-MRCU-1.0	28 484	87.77	4	0.90	0.004	2 821

NPA (None Power Aware) [42], DVFS [15], THR-MMT-1.0 [33], THR-MMT-0.8 [33], MAD-MMT-2.5 [33], IQRMMT-1.5 [33], KAM-MMS-2.0 [35], KAI-MMS-1.0 [35], and KMI-MRCU-1.0. In terms of energy efficiency, the higher the value of the energy efficiency, the better. As for energy consumption, SLA violations, SVTAH, and PDCVM, the less the value the better. For the number of VM migrations, excessive or too little VM migration is bad for energy efficiency. NPA does not take any measure during processing tasks. It consumes 2410.8 kWh. DVFS takes dynamic voltage and frequency scaling technique to reduce energy consumption, it consumes 803.91 kWh. Compared with NPA, DVFS is effective. However, both NPA and DVFS do not involve the migration of VM, we use notation "—" to express nonexistence energy efficiency, SLA violations, SVTAH, PDCVM, and number of VM migrations.

Compared with THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5, algorithm KMI-MRCU-1.0 improves energy efficiency by more than 10 times, reduces energy consumption by more than 10%, SLA violations by more than 10 times, SVTAH by more than 5 times, PDCVM by more than 10 times, number of VM migration by more than 9 times. The reason can be explained as follows: (1) the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) are based on two-threshold framework, while algorithm KMI-MRCU-1.0 is based on adaptive three-threshold framework. In our previous study such as [31], we have identified that the adaptive three-threshold algorithm is more effective than two-threshold algorithm; (2) the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) consider minimizing the energy consumption during the VM placement, while algorithm KMI-MRCU-1.0 considers maximum energy efficiency. The experimental results in Section 4.6.1 have been proved that the latter has better performance than the former; (3) for the overloaded host detection algorithm, the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) leverage the statistical analysis of historical data, while algorithm KMI-MRCU-1.0 utilizes K-Means cluster algorithm before using the historical data; (4) during selection of VMs, the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) consider single factor such as CPU or memory utilization, while algorithm KMI-MRCU-1.0 considers both CPU and memory utilization.

Compared with KAM-MMS-2.0 and KAI-MMS-1.0, algorithm KMI-MRCU-1.0 improves energy efficiency by more than 3 times by keeping almost equal energy-consumption, reduces SLA violations by more than 3 times, SVTAH by more than 2 times, PDCVM by more than 2 times, and the number of VM migration by more than 2 times. The reason can be concluded as follows: (1) the two algorithms (KAM-MMS-2.0 and KAI-MMS-1.0) consider minimizing the energy consumption during the VM placement, while algorithm KMI-MRCU-1.0 considers maximum energy efficiency. The experimental results in Section 4.6.1 have been proved that the latter has better performance than the former; (2) for processing CPU intensive task, the host overloaded algorithm and VM selection policy of KMI-MRCU-1.0 is more effective than KAI-MMS-1.0 and KAM-MMS-2.0, the reason can be explained by the fact that KMI-MRCU-1.0 considers both CPU and memory factors. Based on Table 6, a conclusion can be drawn that KMI-MRCU-1.0 is more effective than other energy-aware algorithms.

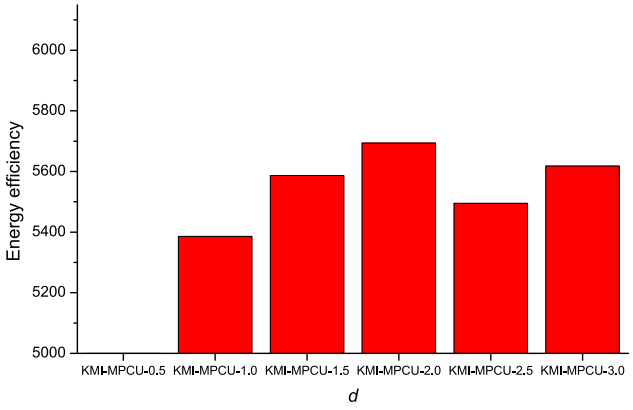


Fig. 15. Energy efficiency under different value of d.

4.6.3. Analysis of I/O intensive task

Taking into account the characteristics of data set at Table 4 in Section 4.2, for example, the mean, quartile 1, and quartile 3 of CPU utilization are the lowest in all of the data set. Therefore, we select "22/March/2011" data set as the I/O intensive task. The comparison algorithms are also NPA (None Power Aware) [42], DVFS [15], THR-MMT-1.0 [33], THR-MMT-0.8 [33], MAD-MMT-2.5 [33], IQRMMT-1.5 [33], KAM-MMS-2.0 [35], and KAI-MMS-1.0 [35]. As with Section 4.6.2, we put the combination of ATF (see Section 4.1), KMI (see Section 4.2), MPCU (see Section 4.3), VPME (see Section 4.4), and parameter d denoted by KMI-MPCU-d (KMI-MPCU-d is for I/O intensive task). In the experiment, the value of parameter d was varied from 0.5 to 3.0 increased by 0.5. Figs. 15–20 show the experimental results as a function of different value of d.

Figs. 15–20 display the energy efficiency (Eq. (11) in Section 4.5), energy consumption, SLA violations (Eq. (10) in Section 4.4), SVTAH (Eq. (8) in Section 4.4), PDCVM (Eq. (9) in Section 4.4) and the number of VM migrations under different value of d , respectively. As shown in Fig. 16 KM1-MPCU-2.0 obtains the minimum energy consumption value when parameter $d = 1.5$. However, in Fig. 17, SLA violations can get a minimum value when parameter $d = 3.0$. Considering both energy consumption and SLA violations (that is energy efficiency), the optimum value for energy efficiency can be got when parameter $d = 2.0$ (see Fig. 15). Therefore, we use KMI-MPCU-2.0 to make a comparison with other energy-aware algorithms.

Table 7 illustrates the experimental results compared with the other baseline algorithms. Table 7 shows the energy efficiency, energy consumption, SLA violations, SVTAH, PDCVM, and number of VM migrations for NPA (None Power Aware) [42], DVFS [15],

Table 7
Comparison with other energy-aware algorithms.

Algorithms	Energy efficiency	Energy consumption (KWh)	SLA violations (10^{-7})	SVTAH (%)	PDCVM (%)	Number of VM migrations
NPA	–	2410.80	–	–	–	–
DVFS	–	1014.21	–	–	–	–
THR-MMT-1.0	32	101.62	3115	27.72	0.11	25 560
THR-MMT-0.8	136	120.91	609	5.49	0.11	33 417
MAD-MMT-2.5	148	117.88	574	5.35	0.11	32 795
IQR-MMT-1.5	134	121.11	615	5.46	0.11	33 061
KAM-MMS-2.0	4922	84.65	24	2.03	0.01	8 736
KAI-MMS-1.0	4612	103.26	21	1.80	0.01	8 190
KMI-MPCU-2.0	5694	67.55	26	2.90	0.01	12 607

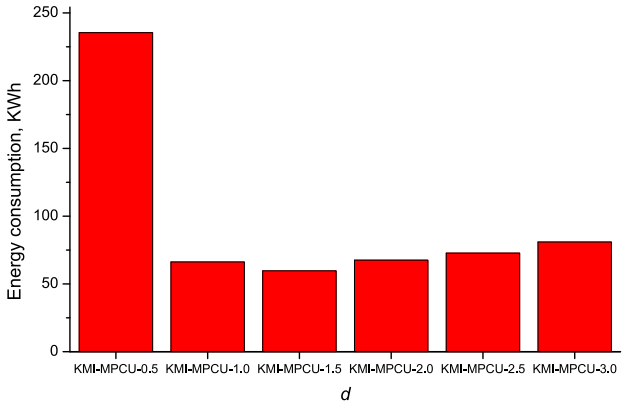


Fig. 16. Energy consumption under different value of *d*.

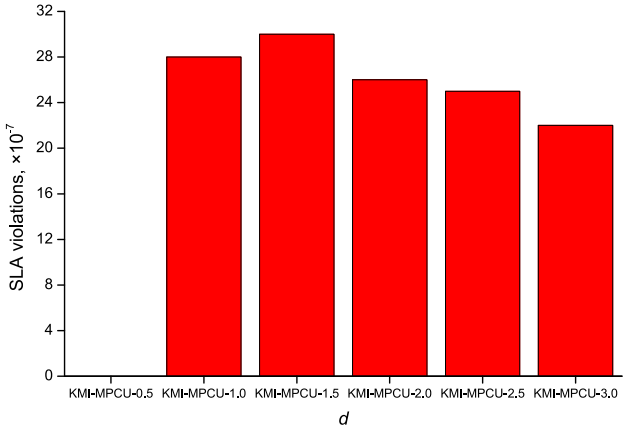


Fig. 17. SLA violations under different value of *d*.

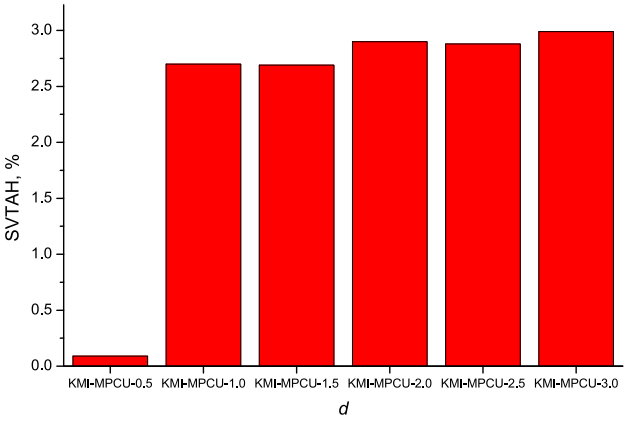


Fig. 18. SVTAH under different value of *d*.

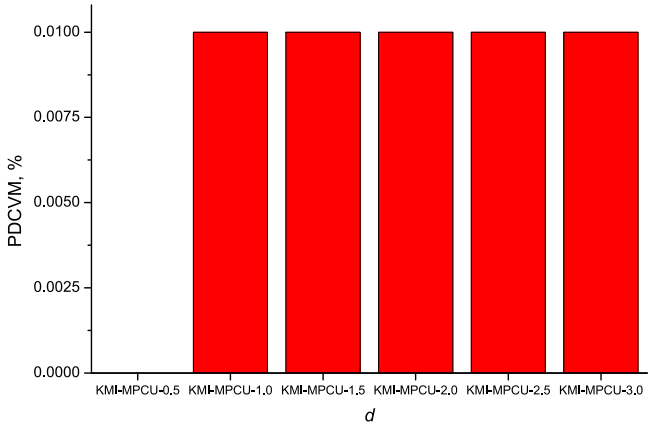


Fig. 19. PDCVM under different value of *d*.

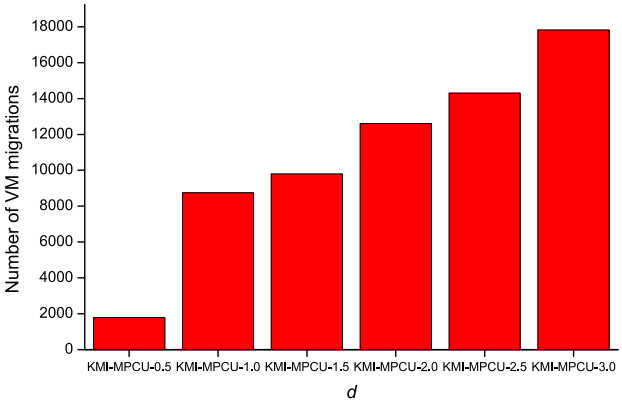


Fig. 20. Number of VM migration under different value of *d*.

THR-MMT-1.0 [33], THR-MMT-0.8 [33], MAD-MMT-2.5 [33], IQR-MMT-1.5 [33], KAM-MMS-2.0 [35], KAI-MMS-1.0 [35], and KMI-MPCU-2.0. As for energy efficiency, the higher value of energy efficiency, the better. In terms of energy consumption, SLA violations, SVTAH, and PDCVM, the less value, the better. For the number of VM migrations, excessive or too little VM migration is bad for energy efficiency. NPA do not take any measure to save power during processing tasks, it consumes 2410.8 kWh. DVFS takes dynamic voltage and frequency scaling technique to reduce energy consumption, it consumes 1014.21 kWh. In contrast to NPA, DVFS is effective. However, both NPA and DVFS do not involve the migration of VM, we use notation "—" to express energy efficiency, SLA violations, SVTAH, PDCVM, and number of VM migrations.

In contrast to THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5, algorithm KMI-MPCU-2.0 improves energy efficiency by more than 10 times, reduces energy consumption by

more than 40%, SLA violations by more than 10 times, SVTAH by more than 40%, PDCVM by more than 10 times, number of VM migration by more than 2 times. The reason can be explained as follows: (1) the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) are based on two-threshold framework, while algorithm KMI-MPCU-2.0 is based on adaptive three-threshold framework. In our previous study such as [31], we have identified that the adaptive three-threshold algorithm is more effective than two-threshold algorithm; (2) the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) consider minimizing the energy consumption during the VM placement, while algorithm KMI-MRCU-1.0 considers maximum energy efficiency. The experimental results in Section 4.6.1 have been proved that the latter has better performance than the former; (3) for the overloaded host detection algorithm, the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) leverage the statistical analysis of historical data, while algorithm KMI-MRCU-1.0 utilizes K-Means cluster algorithm before using the historical data; (4) during selection VMs, the four algorithms (THR-MMT-1.0, THR-MMT-0.8, MAD-MMT-2.5, and IQR-MMT-1.5) consider single factor such as CPU or memory utilization, while algorithm KMI-MRCU-1.0 considers both CPU and memory utilization.

Compared with KAI-MMS-1.0 and KAM-MMS-2.0, algorithm KMI-MPCU-2.0 improves energy efficiency by more than 20% and reduces energy consumption by more than 20% under keeping almost equal SLA violations. The reason can be concluded as follows: (1) the two algorithms (KAI-MMS-1.0 and KAM-MMS-2.0) consider minimizing the energy consumption during the VM placement, while algorithm KMI-MPCU-2.0 considers maximum energy efficiency. The experimental results in Section 4.6.1 have been proved that the latter has better performance than the former; (2) for processing I/O intensive task, the host overloaded algorithm and VM selection policy of KMI-MPCU-2.0 is more effective than KAI-MMS-1.0 and KAM-MMS-2.0, the reason is KMI-MPCU-2.0 considers both CPU and memory factors. Based on Table 7, a conclusion can be drawn that KMI-MPCU-2.0 is more effective than other energy-aware algorithms.

5. Conclusion

This paper puts forward two energy-aware algorithms (KMI-MRCU-1.0 and KMI-MPCU-2.0) based on the adaptive three-threshold framework (ATF), KMI algorithm, VM selection policies (MRCU, MPCU), and maximum energy efficiency placement of VM (VPME) according to the difference of workload. The experimental results show that: (1) regarding the energy efficiency, the algorithm with maximizing the energy efficiency (VPME) has a better performance than the algorithm with minimizing the energy consumption during the placement of VM; (2) during the selection of a VM, considering both CPU and memory factor is more effective than a single factor such as CPU. Moreover, the algorithms proposed in this paper are more effective than the other energy-aware algorithms regardless the workload types. We are planning to implement the proposed algorithms on real-world cloud platforms and study how much energy efficiency improvements and reduction in the operation cost is obtained.

Acknowledgments

This work was done while the first author had a visiting position at the School of Information Technology, Deakin University, Australia. This work was supported by the National Natural Science Foundation of China (nos. 61572525, 61373042, and 61404213), and China Scholarship Council. This work was also supported partially by Deakin University and the Deanship of Scientific Research at King Saud University, Riyadh, Saudi Arabia through the research group project No RGP-VPP-318. The help of Maliha Omar is also sincerely appreciated.

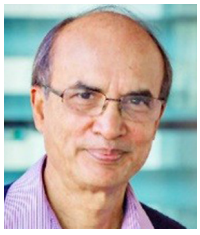
References

[1] R. Wang, P. Shang, J. Zhang, Q. Wang, T. Liu, J. Wang, MAR: A novel power management for cmp systems in data-intensive environment, IEEE Trans. Comput. 65 (6) (2016) 1816–1830.
[2] Y. Chen, J.M. Chang, EMaaS: Cloud-based energy management service for distributed renewable energy integration, IEEE Trans. Smart Grid 6 (6) (2015) 2816–2824.
[3] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.
[4] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. (FGCS) 28 (5) (2012) 755–768.
[5] J.M. Kaplan, W. Forrest, N. Kindler, Revolutionizing Data Center Energy Efficiency; technical Report, McKinsey& Company: New York, NY, USA, 2008.
[6] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Trans. Parallel Distrib. Syst. (TPDS) 24 (7) (2013) 1366–1379.
[7] Jemal H. Abawajy, Mohd Farhan Md Fudzee, Mohammad Mehedi Hassan, Majed A. Alrubaian, Service level agreement management framework for utility-oriented computing platforms, J. Supercomput. 71 (11) (2015) 4287–4303.
[8] L.A. Barroso, U. Holzle, The case for energy-proportional computing, Computer 40 (12) (2007) 268–280.
[9] M.R.V. Kumar, S. Raghunathan, Heterogeneity and thermal aware adaptive heuristics for energy efficient consolidation of virtual machines in infrastructure clouds, J. Comput. System Sci. 82 (2) (2016) 191–212.
[10] G. Han, W. Que, G. Jia, L. Shu, An efficient virtual machine consolidation scheme for multimedia cloud computing, Sensors 16 (2) (2016) 1–17.
[11] S.B. Shaw, A.K. Singh, Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center, Comput. Electr. Eng. 47 (1) (2015) 241–254.
[12] A. Verma, P. Ahuja, A. Neogi, pMapper: Power and migration cost aware application placement in virtualized systems, in: Proc. Ninth ACM/IFIP/USENIX Int'l Conf. Middleware, 2008, pp. 243–264.
[13] G. Jung, M.A. Hiltunen, K.R. Joshi, R.D. Schlichting, C. Pu, Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures, in: Proc. 30th Int'l Conf. Distributed Computing Systems, ICDCS, 2010, pp. 62–73.
[14] G. Buttazzo, Scalable applications for energy-aware processors, in: Embedded Software, in: ser. Lecture Notes in Computer Science, vol. 2491, Springer Berlin Heidelberg, 2002, pp. 153–165.
[15] H. Hanson, S.W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, 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, Aug, 2007, pp. 219–224.
[16] C.M. Wu, R.S. Chang, H.Y. Chan, A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters, Future Gener. Comput. Syst. 37 (1) (2014) 141–147.
[17] A. Wierman, L.L. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems, in: Proceedings of the 28th Conference on Computer Communications, INFOCOM, 2009, pp. 2007–2015.
[18] L.L. Andrew, M. Lin, A. Wierman, Optimality, fairness, and robustness in speed scaling designs, in: Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS, 2010, pp. 37–48.
[19] K. Flautner, S. Reinhardt, T. Mudge, Automatic performance setting for dynamic voltage scaling, Wirel. Netw. 8 (5) (2002) 507–520.
[20] A. Weissel, F. Bellosa, Process cruise control: event-driven clock scaling for dynamic power management, in: Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, 2002, pp. 238–246.
[21] S. Lee, T. Sakurai, Run-time voltage hopping for low-power real-time systems, in: Proceedings of the 37thAnnual ACM/IEEE Design Automation Conference, DAC, 2000, pp. 806–809.
[22] J.R. Lorch, A.J. Smith, Improving dynamic voltage scaling algorithms with PACE, ACM SIGMETRICS Perform. Eval. Rev. 29 (1) (2001) 50–61.
[23] R. Buyya, R. Ranjan, 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, 2009, pp. 1–11.
[24] A. Beloglazov, R. Buyya, Energy efficient allocation of virtual machines in cloud data centers, in: IEEE TCSC Doctoral Symposium, Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2010, Melbourne, Australia, May, 2010, pp. 577–578.

[25] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: IEEE TCSC Doctoral Symposium, Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2010, Melbourne, Australia, May, 2010, pp. 17–20.
[26] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, C. He, Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing, Computing 98 (3) (2016) 303–317.
[27] X. Fu, C. Zhou, Virtual machine selection and placement for dynamic consolidation in Cloud computing environment, Front. Comput. Sci. 9 (2) (2015) 322–330.
[28] X. Zhu, et al., 1000 Islands: Integrated capacity and workload management for the next generation data center, in: Proc. Fifth Int'l Conf. Autonomic Computing, ICAC, 2008, pp. 172–181.
[29] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi, A. Kemper, An integrated approach to resource pool management: Policies, efficiency and quality metrics, in: Proc. IEEE 38th Int'l Conf. Dependable Systems and Networks, DSN, 2008, pp. 326–335.
[30] VMware Distributed Power Management Concepts and Use, Information Guide, VMware Inc. 2010.
[31] Z. Zhou, Z.G. Hu, T. Song, J.Y. Yu, A novel virtual machine deployment algorithm with energy efficiency in cloud computing, J. Cent. South Univ. 22 (3) (2015) 974–983.
[32] A. Beloglazov, R. Buyya, Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers, in: Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and E-Science (MGC 2010), ACM, Bangalore, India, 2010.
[33] A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, Concurr. Comput.: Pract. Exper. (CCPE) 24 (13) (2012) 1397–1420.
[34] B. Guenter, N. Jain, C. Williams, Managing cost performance and reliability tradeoffs for energy-aware server provisioning, in: Proc. IEEE INFOCOM, 2011, pp. 1332–1340.
[35] Z. Zhou, Z. Hu, K. Li, Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers, Sci. Program. 2016 (1) (2016) 1–11.
[36] A. Higai, A. Takefusa, H. Nakada, M. Oguchi, A study of effective replica reconstruction schemes at node deletion for HDFS, in: Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2014, pp. 512–521.
[37] B. Mao, H. Jiang, S. Wu, Y. Fu, L. Tian, Read-performance optimization for deduplication-based storage systems in the cloud, ACM Trans. Storage 10 (2) (2014) 1–44.
[38] T. Gunarathne, J. Qiu, D. Gannon, Towards a collective layer in the big data stack, in: Proceedings of IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2014, pp. 236–245.
[39] Xiang Li, Peter Garraghan, Xiaohong Jiang, Zhaohui Wu, Jie Xu, Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy, IEEE Trans. Parallel Distrib. Syst. (2017). http://dx.doi.org/10.1109/TPDS.2017.2688445. 28 March.
[40] Nguyen, Trung Hieu, Mario Di Francesco, Antti Yla-Jaaski, Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers, IEEE Trans. Serv. Comput. (2017). http://dx.doi.org/10.1109/TSC.2017.2648791. 05 January.
[41] W. Voorsluys, J. Broberg, S. Venugopal, 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, in: LectureNotes in Computer Science, vol. 5931, Springer, Berlin, Germany, 2009, pp. 254–265.
[42] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms, Softw. - Pract. Exp. 41 (1) (2011) 23–50.
[43] F. Durao, J.F.S. Carvalho, A. Fonseka, V.C. Garcia, A systematic review on cloud computing, J. Supercomput. 68 (3) (2014) 1321–1346.
[44] K.S. Park, V.S. Pai, CoMon: a mostly-scalable monitoring system for planetLab, Oper. Syst. Rev. 40 (1) (2006) 65–74.
[45] X. Fan, W.D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th Annual International Symposium on Computer Architecture, (ISCA'07), ACM, 2007, pp. 13–23.
[46] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, G. Jiang, Power and performance management of virtualized computing environments via lookahead control, Cluster Comput. 12 (1) (2009) 1–15.

Zhou Zhou received the Ph.D. degree from Central South University, Changsha, China, in 2017, majoring in computer science. He is currently a lecturer at Changsha University. Also, he has accepted a post-doctoral position in Hunan University for two years (from 2017–2019). His research interests include Cloud computing, energy consumption model, energy-efficient resource management, and virtual machine deployment.

Jemal Abawajy (SM'11) is a full professor at Faculty of Science, Engineering and Built Environment, Deakin University, Australia. Prof. Abawajy has delivered more than 50 keynote and seminars worldwide and has been involved in the organization of more than 300 international conferences in various capacity including chair and general co-chair. He has also served on the editorial-board of numerous international journals including IEEE Transaction on Cloud Computing. Prof. Abawajy is the author/co-author of more than 250 refereed articles and supervised numerous Ph.D. students to completion.



Morshed Chowdhury received his Ph.D. degree in Computing from Monash University, Australia in 1999. He is a Senior Lecturer in the School of Information Technology, Deakin University, Burwood, Australia. He has more than 12 years of industry experiences and has published more than one hundred research papers. He has organized a number of international conferences and served as a member of program committees of several international conferences. He has also acted as reviewer of many IEEE and Elsevier journal papers.

Zhigang Hu received his M.S. degree and Ph.D. degree in Central South University in 1988, 2002 respectively. Now, He is a professor at the School of Software, Central South University. His research interests include high performance computing, Cloud computing, energy-efficient resource management, and virtual machine deployment.

Keqin Li is a SUNY distinguished professor of computer science. He is also an Intellectual Ventures endowed visiting chair professor at Tsinghua University, China. His research interests mainly include design and analysis of algorithms, parallel and distributed computing, and computer networking. He has more than 290 refereed research publications. He is currently or has served on the editorial board of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, Journal of Parallel and Distributed Computing, International Journal of Parallel, Emergent and Distributed Systems, International Journal of High Performance Computing and Networking, International Journal of Big Data Intelligence, and Optimization Letters. He is a senior member of the IEEE.

Hongbing Cheng received his Ph.D. degree in Network and Information Security from Nanjing University of Posts & Telecommunications in 2008. He worked as research fellow in Nanjing University, China; University of Stavanger, Norway and Manchester University, England from 2010 to 2013. He is a Professor of College of Computer and Software, Zhejiang University of Technology, Hangzhou, China. He has authored more than 50 refereed journal and conference papers. His research interests include cloud computing, information and network security, big data security, and wireless sensor networks. Dr. Cheng is an

Associate Editor of Journal of Network and Information Security. He has served as Symposium Chair and Session Chair for several international conferences.

Abdulhameed Alelaiwi is a faculity member of Software Engg. Department, at the College of Computer and Information Sciences, King Saud University. Riyadh, Saudi Arabia. He received his Ph.D. degree in Software Engineering from the College of Engineering, Florida Institute of Technology-Melbourne, USA. He has authored and co-authored many publications including refereed IEEE/ACM/Springer journals, conference papers, books, and book chapters. His research interest includes software testing analysis and design, cloud computing, and multimedia. He is a member of IEEE.

Fangmin Li received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2001, majoring in computer science. He is currently a Professor at Changsha University. He has authored several books on embedded systems and over 30 academic papers in wireless networks, and also holds ten Chinese patents. His current research interests include cloud computing, wireless communications and networks, and energy-efficient resource management.