International Journal of Parallel Programming (2019) 47:481–501
https://doi.org/10.1007/s10766-018-00622-x

Migration Cost and Energy-Aware Virtual Machine Consolidation Under Cloud Environments Considering Remaining Runtime

Heyang Xu[1] · Yang Liu[1] · Wei Wei[1] · Ying Xue[1]

Received: 14 September 2018 / Accepted: 18 December 2018 / Published online: 3 January 2019

Abstract
By live migration technology, multiple virtual machines (VMs) can be consolidated into a fewer physical servers and the idle ones can be shut down or switched to low-power mode, thus reducing the energy consumption of cloud data centers. However, live migration can result in performance degradation of migrated VMs, or even interrupting their services. At the same time, live migration can also aggravate the overheads of data transmissions and produce additional energy consumption in cloud data centers. All these negative influences belong to migration cost (MC) caused by VM migration, which becomes an important cost factor that can't be ignored. Otherwise, another important concern, remaining runtime of the migrated VM, also has influence on the efficiency of VM consolidation, which is not well addressed as well. This paper investigates MC-aware VM consolidation problem and formulates the problem as a multi-constraint optimization model by considering migration cost and remaining runtime of VMs. Based on the proposed model, a heuristic algorithm, called MC-aware VM consolidation (MVC) algorithm, is developed. Finally, based on a real-world cloud trace, we conduct extensive experimental studies to verify the validity of the proposed model and algorithm. Experimental results show that, compared with some popular algorithms, MVC algorithm can effectively decrease the migration cost and, at the same time guarantee the energy consumption within a certain low level.

Keywords Live migration · Migration cost · Remaining runtime · Virtual machine consolidation

✉ Yang Liu
liu_yang@haut.edu.cn

Extended author information available on the last page of the article

1 Introduction

Cloud computing is a large-scale distributed computing paradigm supported by state-of-the-art data centers, in which a pool of computing resources is available to users via the Internet [1]. In recent years, increasing demand for computational resources has led to a significant growth in the number of physical servers, along with almost a double of the energy consumed by these servers and cooling infrastructures that support them [2]. Their share of power consumption is approximately between 1.1% and 1.5% of the total electricity used worldwide and is projected to rise even more [3]. High energy consumption gives rise to a large amount of operational cost which can accumulate more than the construction cost of servers and infrastructures in a short period [4]. Data center providers are going to great lengths to minimize their cooling costs, with some providers installing their data halls in cold weather climates [5, 6].This indicates the need for cloud service providers to adopt energy efficient resource management approach to ensure that their profit margin is not dramatically reduced due to high energy costs [7].

Virtualization technology allows cloud providers to create multiple VMs on a single physical server (PS) and provides performance isolation between applications running on the different VMs, which is widely adopted in modern cloud data centers (CDCs) [8]. By using live migration [9, 10], VMs distributed on multiple low-utilization servers can be dynamically consolidated on a fewer PSs and the idle ones can be switched to low-power modes to reduce energy consumption. This approach is known as VM consolidation, which is seen as an efficient solution to cut down electric cost of CDCs [11, 12]. However, VM migration may depress the performance of the migrated VM, or even temporarily interrupting its service [13]. At the same time, this process also aggravates the overheads of data transmissions and results in additional energy consumption of a CDC. All these negative influences belong to the migration cost (MC) caused by VM migration.

It is note that different types of VMs occupy different amount of physical resources (such as compute, memory, storage and bandwidth, etc.) and execute different applications, thus the migration costs caused by heterogeneous VMs are distinguishing heavily [14]. Although the MC of a single migrated VM is relatively low, the comprehensive MC of a modern CDC is quite considerable due to the fact that frequent VM migrations may be needed in the daily management of a CDC [15]. Sometimes, the MC is even higher than the benefit of energy saving by VM consolidation. Therefore, the migration cost caused by VM consolidation has become an important cost factor in CDCs and can't be ignored any more.

Otherwise, another concerned factor, remaining runtime of VMs to be migrated, also has impact on the efficiency of VM consolidation. For example, if a VM with short remaining runtime is selected to be migrated, the MC of migrating the VM cloud be larger than the cost saving of consolidating the VM.

From above discussions, it can be seen that, in VM consolidation, migration costs and remaining runtimes of the migrated VMs are two important aspects, which can significantly influence the efficiency of VM consolidation. The two aspects shall be considered and properly addressed so that a more comprehensive VM consolidation algorithm for CDCs can be developed, which could help to decrease the energy con-

sumption and electric cost of cloud providers, leading to a sustainable and green cloud system [15].

Many efforts have been focused on VM consolidation in CDCs [2, 3, 10–13] (details are given in Sect. 2). In the literature, most common methods to select VMs to be migrated are according to the factors of the number of migrations [2, 16], occupied resources of VMs [4, 17], or load balancing [17, 18] and etc. Migration cost and remaining runtime, as two important restraining aspects in VM consolidation, are still not well addressed. To the best of our knowledge, none of the existing VM consolidation algorithms has taken into account the factor of remaining runtime so far and only few research works on VM consolidation consider the factor of MC, which deserves further research. Therefore, in this paper, we make an endeavor to investigate the problem of MC-aware VM consolidation in CDCs by considering the influence of the two important impacted aspects. The studied problem is thus formulated as a multi-constraint optimization model. Then, a multi-step heuristic algorithm, called MC-aware VM consolidation (MVC) algorithm, is developed. Finally, extensive experimental studies have been conducted based on a real-world cloud trace. Experimental results show that, compared with some popular algorithms (i.e., algorithms presented in [17, 19]), the proposed MVC algorithm can effectively decrease the migration cost and, at the same time, guarantee the energy consumption within a certain low level at the cost of slightly relaxing 0.12 percentage point of SLA violation.

The rest of this paper is organized as follows. Section 2 introduces a review of the related references. Section 3 presents the details of MC-aware VM consolidation problem and its formulation. The descriptions of the developed MVC algorithm are presented in Sect. 4. Simulations, results and analyses are given in Sect. 5. Finally, we conclude this paper and present the future work in Sect. 6.

2 Related Work

Extensive attentions have been paid on energy-aware resource management in cloud computing environments. VM consolidation is seen as an efficient solution to improve resource utilization and reduce electric cost of CDCs. Recently, many studies have explored VM consolidation in CDCs and some VM consolidation methods have been proposed. For example, Beloglazov and Buyya [7] introduced a modified best fit decreasing (MBFD) algorithm by first sorting the VMs in the decreasing order and PSs in the increasing order on the basis of CPU processing capacity and then dynamically migrating VMs on PSs with suitable resources by using first fit decreasing algorithm. Sharma et al. [20] proposed a hybrid approach, HGAPSO, to minimize the energy consumption by saving the wastage of resource in a CDC. Perumal and Subbiah [21] proposed a worst fit heuristic VM placement algorithm to place the VMs over the PSs and a VM consolidation algorithm to improve power conservation while Mastroianni et al. [22] developed a probability-based VM consolidation algorithm, ecoCloud, in self-organization clouds. Marotta et al. [23] developed a simulated annealing based algorithm to solve VM consolidation problem by evaluating the attractiveness of the possible VM migrations. Cui et al. [24] studied policy-based

VM consolidation problem under a multi-tier tree topology DC network environment and proposed a synergistic scheme to jointly consolidate network policies and VMs.

Guo et al. [13] and Zhao et al. [25] further took into account the association between VMs. In [13], the authors proposed a VM migration algorithm based on a group selection method, which is according to the degree of connectivity of the partitioned VM groups. Zhao et al. [25] developed a power-aware and performance-guaranteed VM placement algorithm to balance between saving energy cost and avoiding VM performance degradation in CDCs. While Fioccola et al. [26] introduced an energy-aware resource orchestration framework and a green migration based VM consolidation algorithm to minimize the overall power consumption and, at the same time, satisfy users' QoS requirements.

Moreover, Gutierrez-Garcia et al. [17] and Xu et al. [18] further took load balance into account. In [17], a load balancing and energy-aware consolidation protocol algorithm was proposed to consolidate heterogeneous VMs in a distributed manner. While in [18], the authors developed a greedy-based load balance consolidation algorithm to minimize the number of active PSs and at the same time, balance the loads among these PSs. Although these research works have shown the effectiveness in saving energy consumption of CDCs, the algorithms presented above mostly ignored the impact of VM migrations.

Some studies aimed to reduce energy consumption while keeping the number of migrations as small as possible. For example, Beloglazov et al. [7] investigated the problem of energy-aware resource allocation of cloud data centers and proposed several heuristic algorithms to reduce the energy consumption via dynamic allocation of VMs. Ye et al. [27] tried to eliminate the performance degradation due to VM co-location and migration. They proposed a profiling-based VM consolidation framework and a polynomial time algorithm to minimize both the numbers of used PSs and VM migrations. Wolke and Pfeiffer [28] adopted several well-known vector bin-packing heuristics (such as first fit and best fit) to address this issue. Tao et al. [29] further took communications between VMs into account, formulated the problem of dynamic VM migration as a triple-objective optimization model and designed a binary graph matching-based bucket-code learning algorithm. Mann [30] studied VM consolidation in multi-core cloud environments by also taking into account three aspects: the number of used PSs, the number of migrations and the number of overload CPUs. The author adopted a constraint programming method to solve the proposed weighted cost function of the three aspects. However, these methods estimate the migration cost only in terms of the number of VM migrations, without considering the inherent heterogeneity of VM migration cost [9, 14].

Only few woks consider migration cost in VM consolidation. For example, Wu et al. [19] investigate how to minimize energy cost at low migration cost by designing a consolidation score function, in which the two conflicting objectives are normalized and summed up with different preference weights. They proposed an improved grouping genetic algorithm (IGGA) to obtain the maximum value of consolidation score by dynamic VM consolidation.

From above-mentioned review of the related works, it can be seen that many efforts have been focused on tackling the VM consolidation problem and proposed some effective approaches based on random selection migration (RSM), minimiza-

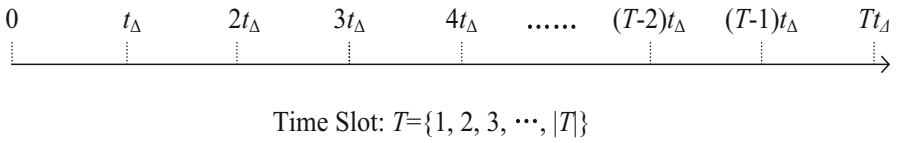

Fig. 1 The division of the whole normal operation period of a cloud data center, T

tion number of migrations (MUoM), minimum migration time (MMT), resource-based migration (RbM), or balance-based migration (BbM) and evolution-based algorithms [31, 32]. However, the impact of migration cost, as an important concern in CDCs, has not been well solved and none of the existing approaches considers the impact of remaining runtimes of migrated VMs [33]. To overcome these disadvantages, this paper studies MC-aware VM consolidation problem and formulates the problem as a multi-constraint optimization model by considering the impact of migration cost and remaining runtimes of migrated VMs.

3 MC-Aware VM Consolidation Problem

3.1 Problem Description

In a normal operation period (T) of a CDC, VM consolidation is to dynamically adjust the mapping of VMs to suitable PSs, according to the dynamic nature of the CDC, such as arrivals of new VM requests, completed execution of VMs, changing of occupied resources of running VMs, and so on. Before presenting the details of the studied MC-aware VM consolidation problem, some definitions should be explicitly understood.

Definition 1 Time Slot (TS). Without loss of generality, assume that the whole normal operation period of a cloud data center is T. As shown in Fig. 1, the whole period of time T can be divided into $|T|$ equal time units. The length of each time unit is t_Δ. One time unit is called a time slot, which can be seen as the essential process unit of time, and thus T can be represented by $T = \{1, 2, \ldots, |T|\}$.

Definition 2 Cloud Data Center (CDC). Without loss of generality, suppose that a CDC consists of M heterogeneous PSs, denoted by $\mathbf{PS} = \{PS_1, PS_2, \ldots, PS_j, \ldots, PS_M\}$, in which PS_j ($1 \leq j \leq M$) represent j-th PS. Denote by Ω_j and Γ_j the number of CPU cores and memory size provided by physical server PS_j, respectively. In a certain time slot $t (1 \leq t \leq |T|)$, if there is at least one VM running on a PS, we call the server as an active server. All the active servers in time slot t constitute an active server set, represented by $\mathbf{AS}(t)$. Denote by $m(t)$ the number of active servers in $\mathbf{AS}(t)$. Thus, $\mathbf{AS}(t)$ and $m(t)$ satisfy the following relations: $\mathbf{AS}(t) \subseteq \mathbf{PS}$ and $m(t) \leq M$.

Definition 3 Virtual Machine (VM). All the VM requests arrival to the CDC during the whole period of time T constitute a VM set $\mathbf{V}$. Denote by $N = |\mathbf{V}|$ the number of VM requests in $\mathbf{V}$. All the running VMs in time slot t constitute a running VM set $\mathbf{V}(t)$. Denote by $n(t)$ the number of VMs in $\mathbf{V}(t)$. Thus, $\mathbf{V}(t)$ and $n(t)$ satisfy the

following relations: $\mathbf{V}(t) \subseteq \mathbf{V}$ and $n(t) \leq N$. Let V_i represent the i-th VM in $\mathbf{V}(t)$, thus $V_i \in \mathbf{V}(t)$ and $1 \leq i \leq n(t)$. Denote by ω_{it} and γ_{it} the amounts of CPU cores and memory required by VM request V_i during time t respectively. According to some real-life cloud environments [34], the occupied resources of VMs have a strong dynamic nature. Therefore, in order to improve the applicability, this paper deal with the scenario that the occupied resource capacities of VM requests are varied during different time slots, which means that the values of ω_{it} and γ_{it} are changing with time. Denote by $at(V_i)$ and $rt(V_i)$ the arrival time and runtime of VM request V_i respectively, and then the running VM set $\mathbf{V}(t)$ in time slot t can be formally described as:

$$\mathbf{V}(t) = \{V_i |\ at(V_i) < \text{t and } at(V_i) + rt(V_i) > t\}. \tag{1}$$

where expression $at(V_i) < t$ means that VM request V_i is submitted to the CDC by a certain cloud user before time slot t and expression $at(V_i) + rt(V_i) > t$ means that the execution of VM request V_i is not completed before time slot t.

Definition 4 Hot Servers (HSs). For a certain PS, if the sum of the occupied resources of all VMs running on it exceeds its resource capacity for any kind of resources (such cores of CPU, Memory, etc.), then we call the PS as a hot server (HS). Once a physical server becomes a HS, it will inevitably violate the service level agreement (SLA), in which the cloud provider guarantees to provide the negotiated quality of service (QoS) to cloud users. The situation will result in loss of profit for the cloud provider. In order to prevent the time of violating SLA from increasing, one or more VMs running on the HS should be migrated to other PSs with adequate amounts of resources.

Definition 5 VM Mapping Policy (VMP). For any time slot t ($1 \leq t \leq |T|$), a mapping of VMs in $\mathbf{V}(t)$ to PSs in $\mathbf{PS}(t)$ is a VMP in time slot t, which can be denoted by a matrix $\mathbf{X}(t) = (x_{ij})_{n(t) \times m(t)}$. If VM request V_i is running on physical server PS_j, then $x_{ij} = 1$; otherwise, $x_{ij} = 0$. $n(t)$ and $m(t)$ are the numbers of running VMs and active PSs in the CDC during time slot t, respectively.

Definition 6 States of PSs. Suppose that all the PSs in the CDC can only be in normal-power state or low-power state. For any physical server $PS_j \in \mathbf{PS}$, if there is at least one VM running on the server, then PS_j is an active server and in normal-power state; otherwise, PS_j is in low-power state. Denote by $s_j(t)$ ($1 \leq j \leq M$) the state of physical server PS_j during time slot t. The states of all the PSs in CDC during time slot t constitute a m-dimensional state vector $\mathbf{S}(t) = (s_1(t), s_2(t), \ldots, s_M(t))$, where $s_j(t) = 1$ indicates that physical server PS_j is in normal-power state and $s_j(t) = 0$ indicates that physical server PS_j is in low-power state. $\mathbf{S}(t)$ can be calculated by the obtained VM mapping policy $\mathbf{X}(t)$, so we have

$$s_j(t) = \begin{cases} 1, & \sum_{i=1}^{n(t)} x_{ij} \geq 1; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

From the above-mentioned definitions, the MC-aware VM consolidation (MVC) problem can be formally described as follows: for any time slot t ($1 \leq t \leq |T|$), all the running VMs in the CDC constitute a set $\mathbf{V}(t)$ and $n(t) = |\mathbf{V}(t)|$; all the active PSs in the CDC constitute a set $\mathbf{PS}(t)$ and $m(t) = |\mathbf{PS}(t)|$; the current VMP is $\mathbf{X}(t) = (x_{ij})_{n(t)\times m(t)}$; then the MVC problem is how to dynamically adjust the mappings of VMs in $\mathbf{V}(t)$ to minimization number of PSs in normal-working state to decrease the energy consumption of the CDC under and some given constraints, according to the dynamic characters of CDC. The mentioned dynamic characters include arrivals of new VM requests, changing of required resources of VMs and the completions of VM requests.

3.2 Optimization Model

VM consolidation is generally used to optimize resource usage and reduce the energy consumption of CDCs. Reducing the number of active PSs in cloud data center to serve the same amount of VM requests with similar performance is of great attractions for cloud providers. Similar to other research on VM consolidation [2, 11, 16–19], this paper uses the number of active PSs as the main metric to measure the degree of energy consumption of a cloud data center. Different with existing studies, we investigate the MVC problem in a long run of the whole normal operation period of a CDC, by conducting VM consolidation in each time slot according to the dynamic natures of the data center. Therefore, the studied problem is formulated as follows:

$$\text{Min} \sum_{j=1}^{M} s_j(t) \quad \forall t \in \{0, 1, \ldots, |T|\}. \tag{3}$$

The optimization objective is to minimize the number of active physical servers in each time slot. Besides energy optimization, we also consider the other two important influencing factors (i.e., migration cost and remaining runtimes) in MVC and the proposed optimization model is subject to following constraints:

C1). Range of variables: the values of decision variables x_{ij} and s_j can only be 0 or 1. Formally:

$$\forall V_i \in \mathbf{V}(t), \quad \forall PS_i \in \mathbf{PS}(t) \; : \; x_{ij} \in \{0, 1\} \text{ and } s_j(t) \in \{0, 1\}. \tag{4}$$

C2). VM mapping constraint: each VM request can only be allocated to exactly one PS to execute. Formally:

$$\forall V_i \in \mathbf{V}(t) : \quad \sum_{j=1}^{m(t)} x_{ij} = 1. \tag{5}$$

C3). Resource capacity constraint: the total CPU/memory requirements of VMs allocated on a physical server should not exceed its CPU/memory capacity. Formally:

$$\forall PS_j \in \mathbf{PS}(t): \quad \sum_{i=1}^{n(t)} x_{ij} \cdot \omega_i(t) \leq \Omega_j \text{ and } \sum_{i=1}^{n(t)} x_{ij} \cdot \gamma_i(t) \leq \Gamma_j. \tag{6}$$

C4). SLA violation percentage (SVP) constraint: The maximum duration of a PS to be allowed to violate SLA must not be longer than one time slot. As the occupied resources vary with time, it could result in a situation that the total resource requirements of VMs allocated on a physical server exceed its resource capacity and violate the SLA. To decrease the time of SLA violation, one or more VMs running on the hot server need to migrate to other physical servers with adequate amounts of resources. Denote by $SLA_j(t)$ whether physical server PS_j violates SLA during time slot t and it can be given by:

$$SLA_j(t) = \begin{cases} 1, \ \sum_{i=1}^{n(t)} x_{ij} \cdot \omega_i(t) > \Omega_j \text{ or } \sum_{i=1}^{n(t)} x_{ij} \cdot \gamma_i(t) > \Gamma_j; \\ 0, \ \text{otherwise.} \end{cases} \tag{7}$$

For the purpose of improving the QoS, the duration of SLA violation must be as short as possible during the whole operation period T. Therefore, in the proposed optimization model, we restrict that the sustained time of SLA violation for a PS must not be longer than a time slot.

C5). Remaining runtime constraint: as the analysis presented in Sect. 1, the MC of a VM with short remaining runtime cloud be larger than the cost saving of consolidating the VM. In order to avoid unnecessary VM migrations, the proposed optimization model restricts that the VMs whose remaining runtimes are less than a time slot will not be migrated.

Overall, the studied MC-aware VM consolidation is formulated as an optimization model with multiple constraints, i.e., minimizing the number of active PSs in each time slot (Eq. (3)) and constraints C1)–C5). From the first two constraints, it can be seen that this problem is actually a variant of the combinatorial optimization problem. Due to the NP-hardness of the studied problem, approximation algorithms that suffice to find a near optimal solution are more promising. Therefore, we propose a heuristic algorithm which is presented in next section.

4 MC-Aware VM Consolidation (MVC) Algorithm

In this section, we present the details of the developed MVC algorithm, which consists of four steps, i.e., pre-processing step, hot server removing step, VM placement step and VM consolidation step.

4.1 Pre-Processing (PP) Step

In this step, MVC algorithm mainly addresses two issues: firstly, for each VM running in the previous time slot, judging whether it has been completed or not, if yes, then MVC algorithm should release its occupied physical resource free; secondly, adjusting the occupied resources of the running VMs in current time slot t, according to their real-time resource requirements. The two aspects are realized by a pre-processing (PP) function whose pseudo-code is shown in Algorithm 1. The inputs of the PP function are the current time slot t and the set of running VMs in time slot $(t-1)$, $\mathbf{V}(t-1)$. The output is the set of running VMs in time slot t, $\mathbf{V}(t)$. The detailed processes of the PP function are as follows:

Algorithm 1: Pre-Processing (PP) Function

Input: current time slot t, $\mathbf{V}(t\text{-}1)$
Output: $\mathbf{V}(t)$

```
1  initialization: V(t)= V(t-1);
2  foreach VM V_i in V(t) do
3      if at(V_i)+rt(V_i)≥(t-1) && at(V_i)+rt(V_i)<t then
4          free the occupied resource of VM V_i;
5          V(t)= V(t)-{V_i};
6      end if
7  end foreach
8  if t %5==0 then //changing occupied resources of running VMs every 5 time slots
9      foreach VM V_i in V(t) do
10         change the occupied resources of V_i according to its current requirements;
11     end foreach
12 end if
13 return V(t);
```

Firstly, before the starting of each time slot t, PP function should free the occupied resources of VMs completed in time slot $(t-1)$ and shutdown the completed VMs (lines 2–7, Algorithm 1). The judging condition in line 3, $at(V_i) + rt(V_i) \geq (t-1)$ and $at(V_i) + rt(V_i) < t$, means that VM V_i is still running in the start of time slot $(t-1)$ and completed before the start of time slot t. It indicates that VM request V_i is completed during time slot $(t-1)$.

Secondly, for every 5 time slots, PP function changes the occupied resources of each running VM according to its current resource requirements (lines 8–12, Algorithm 1). The purpose of this step is to accommodate to the dynamic natures of the real-life cloud scenarios. In google cluster-usage traces (version 2) which is one of the popular real-life cloud traces reported by Google Inc in 2014, the occupied resources of each VM change every 5 min periodically. Therefore, in the developed algorithm, we choose to dynamically change the required resources of running VMs for every five time slots as well.

4.2 Hot Server Removing (HS-Removing) Step

After the PP step, the MVC algorithm gets into HS-Removing step whose main assignment is to cut down the resource utilization of hot servers by migrating one or more VMs form them. The assignment is realized by HS-Removing function (as shown in Algorithm 2). The inputs of the HS-Removing function are: t-the current time slot t, $\mathbf{V}(t)$-the set of running VMs, $\mathbf{PS}(t)$-the set of active physical servers in time slot t, and the current VM mapping policy $\mathbf{X}(t)$. The output is a new VM mapping policy $\mathbf{X}'(t)$ after removing hot servers in time slot t.

In HS-Removing function, two important factors, MC and remaining runtime, are taken into account when chooses the VMs to migrate. The MC of VM V_i includes four parts, i.e., migration time $T_{mig}(V_i)$, downtime $T_{down}(V_i)$, migration energy $E_{mig}(V_i)$ and performance degradation caused by migrating VM V_i [2, 9, 13, 14]. The fourth part is equal to increase the runtime of the migrated VM by 10% of its migration time because, in the migration process, the average performance degradation can be estimated as approximately 10% of the CPU utilization [2]. The other three parts are different in both unit and weight of their values. Thus, in order to find the suitable VM with minimal MC in the hot server, the HS-Removing function defines a parameter of cost factor, shown in Definition 7.

Definition 7 Cost Factor (CF). The cost factor of migrating a VM is a weighted normalization value of migration time, downtime and migration energy. If physical server PS_j violates SLA in time slot t, denote by $\mathbf{V}_j(t)$ the VMs running on PS_j. For each VM V_i in $\mathbf{V}_j(t)$, the cost factor of migrating VM V_i, denoted as $CF_{mig}(V_i)$, can be given by:

$$CF_{mig}(V_i) = \alpha \cdot \frac{T_{mig}(V_i)}{\max\limits_{V_i \in \mathbf{V}_j(t)} \{T_{mig}(V_i)\}} + \beta \frac{T_{down}(V_i)}{\max\limits_{V_i \in \mathbf{V}_j(t)} \{T_{down}(V_i)\}} + \gamma \frac{E_{mig}(V_i)}{\max\limits_{V_i \in \mathbf{V}_j(t)} \{E_{mig}(V_i)\}}. \tag{8}$$

In Eq. (8), the terms $\max\limits_{V_i \in \mathbf{V}_j(t)} \{T_{mig}(V_i)\}$, $\max\limits_{V_i \in \mathbf{V}_j(t)} \{T_{down}(v_i)\}$ and $\max\limits_{V_i \in \mathbf{V}_j(t)} \{E_{mig}(v_i)\}$ represent the maximum of migration time, downtime and migration energy of migrating all the VMs in $\mathbf{V}_j(t)$, respectively. α, β and γ are weighting coefficients of three aspects and $\alpha + \beta + \gamma = 1$.

Algorithm 2: Hot Server Removing (HS-Removing) Function

Input: current time slot t, $\mathbf{V}(t)$, $\mathbf{PS}(t)$ and $\mathbf{X}(t)$
Output: new VM mapping policy $\mathbf{X}'(t)$

```
1  initialization: X'(t)= X(t);
2  foreach physical server PS_j in PS(t) do
3      if PS_j is a hot server according to Eq.() then
4          foreach VM V_i in V_j(t) do
5              if V_i can be completed during in time slot t then
6                  add V_i to its completed VM set CV_j(t);
7              end if
8          end foreach
9          if PS_j is not a hot server after freeing the occupied resources by CV_j(t) then
10             continue;
11         foreach VM V_i in V_j(t) do
12             if PS_j is not a hot server after migrating V_i then
13                 calculate CF_mig(V_i);
14                 mark the VM with minimum CF as V_k;
15             end if
16         end foreach
17         mark VM V_k as the VM to migrate;
18         PS_d ← PS_Selection(V_k);
19         migrate V_k from PS_j to PS_d;
20         change the value of VMP X'(t): x'_kj=0、x'_kd=1;
21     end if
22 end foreach
```

The detailed processes of the HS-Removing function are as follows:

Initially, the values of elements in $\mathbf{X}'(t)$ are set as the values of corresponding elements in $\mathbf{X}(t)$.

Then, for each physical server PS_j in $\mathbf{PS}(t)$, the HS-Removing function judges whether PS_j is a hot server or not, according to Eq. (9). If yes, it will get into the following procedures (lines 3–21, Algorithm 2): First, for all VMs running on PS_j, HS-Removing function finds out the VMs that can be completed during time slot t and then judges whether PS_j can remove SLA violation when these VMs free their occupied resource; if yes, the HS-Removing function can guarantee the constraint of SLP without any VM migration and continue to judge next PSs (lines 4–10, Algorithm 2); otherwise, the HS-Removing function selects the VM, which has the minimum MC and can satisfy the SLA violation constraint after migration, and migrates the VM to other PSs with adequate resource capacity (lines 11-20, Algorithm 2).

$$\sum_{i=1}^{n(t)} x_{ij} \cdot \omega_i(t) > \Omega_j \quad \text{or} \quad \sum_{i=1}^{n(t)} x_{ij} \cdot \gamma_i(t) > \Gamma_j. \tag{9}$$

In line 18 of Algorithm 2, PS_Selection(V_k) function is used to select a destination PS for VM V_k, which needs to migrate. The pseudo-code of the function is shown

in Algorithm 3 whose inputs are VM V_k and $\mathbf{PS}(t)$, and output is the selected destination server PS_j. The detailed processes of the PS selection function are as follows: The function firstly selects the destination server for VM V_k from active server set $\mathbf{PS}(t)$ (lines 1–6, Algorithm 3). However, if there is no suitable PS exists in $\mathbf{PS}(t)$ to accommodate V_k, the function will start a new PS (lines 7–11, Algorithm 3).

Algorithm 3: PS_Selection function

Input: VM V_k and $\mathbf{PS}(t)$
Output: destination physical server
1 sort the physical servers in $\mathbf{PS}(t)$ in ascending order by the sum of residual CPU and memory capacity, such as $P'_1, P'_2, \ldots, P'_{m(t)}$;
2 **for** j=1 to $m(t)$ **do**
3 **if** there is adequate resource capacity on P'_j to allocate to VM V_k **then**
4 **return** P'_j;
5 **end if**
6 **end for**
7 **if** $j>m(t)$ **then** //no suitable physical server exists in $\mathbf{PS}(t)$ to accommodate V_k
8 start a new physical server in low-power state with maximum sum of CPU and memory capacity, such as PS_j;
9 add physical server PS_j to active physical server set $\mathbf{PS}(t)$;
10 **end if**
11 **rerurn** PS_j;

4.3 VM Placement Step

Denote by $\mathbf{V}_{in}(t)$ the set of VM requests submitted to CDC by users during time slot t. For each VM request V_i in $\mathbf{V}_{in}(t)$, its arrival time satisfies the following expression: $t \leq at(V_i) < (t+1)$. VM placement step is invoked at the start of time slot t and its main task is to deploy all the new VMs requests submitted during time slot $(t-1)$ to suitable PSs. The algorithm of this step is called VM placement (VMP) function whose pseudo-code is shown in Algorithm 4. The inputs are the current time slot t, $\mathbf{V}_{in}(t-1)$, $\mathbf{PS}(t)$ and $\mathbf{X}(t)$. The output is a new VM mapping policy $\mathbf{X}'(t)$ after deploying the VM requests in $\mathbf{V}_{in}(t-1)$ to PSs in $\mathbf{PS}(t)$. The detailed processes of the VMP function are as follows:

Initially, the values of elements in $\mathbf{X}'(t)$ are set as the values of corresponding elements in $\mathbf{X}(t)$ (line 1, Algorithm 4).

Then, VMP function sorts the active PSs in ascending order on the basis of the sum of residual CPU and memory capacity, such as $PS'_1, PS'_2, \ldots, PS'_{m(t)}$ (line 2, Algorithm 4).

Third, for each new VM request V_k, VMP function first checks whether there is a suitable PS to deploy V_k from active server set $\mathbf{PS}(t)$ (line 4, Algorithm 4). If yes, VMP function deploys VM request V_k to the selected physical server PS'_j and add V_k to the running VM set $\mathbf{V}(t)$ (lines 5–10, Algorithm 4). Otherwise, the VMP function will start a new physical server and deploys VM request V_k to the new started server (lines 12–18, Algorithm 4).

Algorithm 4: VM Placement (VMP) Function

Input: current time slot t, $\mathbf{V}_{in}(t\text{-}1)$, $\mathbf{V}(t)$, $\mathbf{PS}(t)$ and $\mathbf{X}(t)$
Output: new VM mapping policy $\mathbf{X}'(t)$

```
1  initialization: X'(t)= X(t);
2  sort the physical servers in PS(t) in ascending order by the sum of residual CPU
     and memory capacity, such as PS'_1, PS'_2, ..., PS'_m(t);
3  foreach VM V_i in V_in(t-1) do
4      for j=1 to m(t) do
5          if PS'_j can satisfy V_i's resource requirements then
6              deploy V_i to physical server PS'_j;
7              V(t)= V(t) ∪ {V_i};
8              change the value of VMP X'(t);
9              break;
10         end if
11     end foreach
12     if there is no suitable physical server in PS(t) to deploy VM V_i then
13         start a new physical server in low-power state with maximum sum of CPU
             and memory capacity, such as PS_j;
14         deploy V_i to physical server PS_j;
15         V(t)= V(t) ∪ {V_i};
16         PS(t) = PS(t) ∪ { PS_j };
17         change the value of VMP X'(t);
18     end if
19 end foreach
```

4.4 VM Consolidation (VMC) Step

In VMC step, the proposed MVC algorithm adopts threshold-based approach, which has been proven to be an effective approach [7, 20, 35]. The main task of this step is to consolidate VMs running on under-utilized physical servers. Denote by $SUMT_{mig}(PS_j)$ the overall migration time of physical servers PS_j, which is equal to the sum of migration times of all the VMs running on PS_j. Thus, we have:

$$SUMT_{mig}(PS_j) = \sum_{V_i \in \mathbf{V}_j(t)} T_{mig}(V_i). \tag{10}$$

The inputs of the VMC function include the defined lower utilization threshold U_{Lower}, $\mathbf{PS}(t)$ and $\mathbf{X}(t)$. The output is a new VM mapping policy $\mathbf{X}'(t)$ after VM consolidation in time slot t. The pseudo-code of VMC function is shown in Algorithm 5. When there exists a physical server PS_j whose sum of CPU and memory utilization is lower than U_{Lower} in $\mathbf{PS}(t)$, VMC function will execute the following processes(lines 2–16, Algorithm 5): First, calculates the overall migration time $SUMT_{mig}(PS_j)$ according to Eq. (10). Second, for each VM (such as V_i) running on PS_j, VMC function will judge whether the remaining runtime of V_i is shorter than $SUMT_{mig}(PS_j)$ or not. If yes, it means that VM V_i can be completed before $SUMT_{mig}(PS_j)$ and there is no need

to migrate VM V_i. Thus, the function leaves VM V_i continually running on physical server PS_j until V_i is completed (lines 5–7, Algorithm 5). Otherwise, VMC function will migrate VM V_i to a suitable physical server (lines 8–11, Algorithm 5). Finally, after processing all the VMs running on PS_j, VMC function will switch physical server PS_j to low-power state when there is no VM running on it (lines 14 and 15, Algorithm 5).

Continue the above-mentioned processes until the resource utilizations of the physical servers in $\mathbf{PS}(t)$ are all higher than U_{Lower}.

Algorithm 5: VM Consolidation (VMC) Function

Input: U_{Lower}, $\mathbf{PS}(t)$ and $\mathbf{X}(t)$
Output: new VM mapping policy $\mathbf{X}'(t)$
1 initialization: $\mathbf{X}'(t)= \mathbf{X}(t)$;
2 **while** there exists a physical server PS_j whose sum of CPU and memory utilization is lower than U_{Lower} in $\mathbf{PS}(t)$ **do**
3 calculate $SUMT_{mig}(PS_j)$ according to Eq.();
4 **foreach** VM V_i running on PS_j **do**
5 **if** the remaining runtime of VM V_i is shorter than $SUMT_{mig}(PS_j)$ **then**
6 VM V_i doesn't need to be migrated;
7 $SUMT_{mig}(PS_j)=SUMT_{mig}(PS_j)-T_{mig}(V_i)$;
8 **else**
9 $PS_d \leftarrow$ PS_Selection(V_i);
10 migrate V_i from PS_j to PS_d;
11 change the value of VMP $\mathbf{X}'(t)$: $x'_{ij}=0$ and $x'_{id}=1$;
12 **end if**
13 **end foreach**
14 switch physical server PS_j to low-power state;
15 $\mathbf{PS}(t) = \mathbf{PS}(t) - \{PS_j\}$;
16 **end while**

Hereinbefore, we introduce the four steps of the developed MVC algorithm in details. The overall flow of MVC algorithm is as follows: at the start of each time slot t ($1 \leq t \leq |T|$), MVC algorithm firstly executes pre-processing function to free the occupied resources of VMs which are completed in time slot $(t-1)$, and dynamically changes the occupied resources of running VMs according to their current requirements at each time slot t, where $t\%5 = 0$. After changing the occupied resources, some physical servers may become hot servers and violate SLA. Once a physical server violating SLA, MVC algorithm will call HS-Removing function to migrate one or more VMs from the hot server to avoid going against the SVP constraint of the proposed optimization model. Then MVC algorithm invokes VMP function to deploy the new VM requests, submitted by users during time slot $(t-1)$, to suitable physical servers. Finally, MVC algorithm execute VMC function to consolidate VMs running on low-utilization servers and switch the ideal ones to low-power state to cut down the energy consumption of the cloud data center.

5 Performance Evaluation

A series of experiment studies has been conducted to verify the feasibility of the proposed optimization model and the performance of the proposed MVC algorithm. The experimental configurations and result analyses are presented in the following sections.

5.1 Experimental Configurations

Our experiments are conducted based on CloudSim toolkit [36], which is a modern and extensible cloud simulation framework and widely adopted by most related research. The setups of cloud infrastructure and VM requests used in the experiments are extracted from a real-life cloud scenario, Google cluster-usage traces [34]. The reported traces contain the records from a cluster of about 12.5 k machines over about a month-long period. From which, we generally choose 29,944 task records from task event tables generated in first 12 h and 1046 physical machine records from a machine event table. Each task record, which can be seen as a VM request, contains attributes of arrival time, runtime, dynamic CPU requests and memory requests that change every 5 min. The last two attributes are normalized values from 0 to 1, in which 1 represents the maximum value of the CPU/memory request in the task event table. Each machine record can be seen as a physical server, which contains attributes of machine ID, states, CPU and memory capacity, which are also normalized from 0 to 1. These normalized values are directly used in the experiment because the CPU/memory capacities of physical servers are about 4–10 times as large as the CPU/memory requests. The total length of simulation time continues 12 h and the time slot defined in Definition 1 is set as 1 min. Thus the whole normal operation period (12 h) of the simulated cloud data center is divided into 720 time slots. The three weighting coefficients in Eq. (8), i.e., α, β and γ, are set as 0.3, 0.3 and 0.4 respectively. The lower utilization thresholds of CPU and memory are both set to 0.2.

5.2 Analyses of Results

To evaluate the performance of the developed MVC algorithm, we compare it with other two recent and representative algorithms on VM consolidation, i.e., energy-aware server consolidation protocol (ESCP) [17] and improved grouping genetic algorithm (IGGA) [19], on three aspects of metrics: energy consumption, VM migration and SLA violation percentage (SVP). In order to decrease causal factors' influences, each experiment has been run 10 times and the results presented in the paper are the mean value of the results obtained by the 10 experiments.

5.2.1 Energy Consumption

In this aspect, we compare mainly two energy consumption related metrics, i.e., number of active physical servers in each time slot and the total number of active physical servers in the whole experimental period, i.e., 12 h.

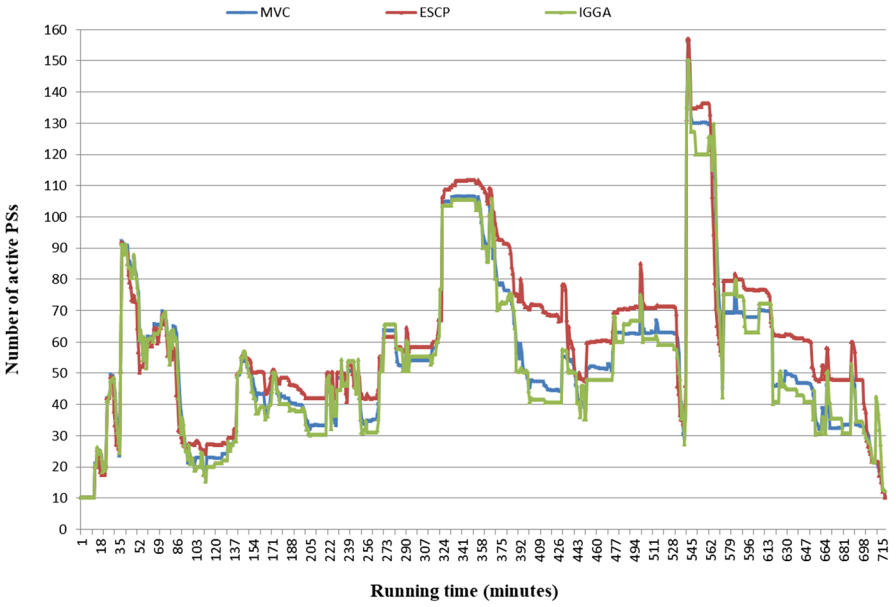

Fig. 2 Number of used physical servers in each time slot

Table 1 Total no. of used physical servers during the whole experimental period (12 h)

Algorithms	MVC	ESCP	IGGA
Total no. of used PSs	39,596	42,823	38,552

Active physical servers are the physical servers used in each time slot. The smaller value of the number of active physical servers, the lower energy consumption of the cloud data center. As shown in Fig. 2, to execute the same number of VM requests submitted by users, the number of used physical servers obtained by the proposed MVC algorithm is smaller than that of ESCP algorithm, and slightly more than that of IGGA algorithm. More specially, Table 1 shows the results of the total number of used physical servers during the whole experimental period. The total used physical servers of MVC algorithm is 7.5% smaller than that of ESCP algorithm and 2.7% slightly more than that of IGGA algorithm. It means that IGGA algorithm and the proposed MVC algorithm can be more effective on energy saving than ESCP algorithm in the long run. The reason lies in: for a VM to be allocated, MVC algorithm always chooses the physical server with the minimal residual resource capacity that can satisfy resource requirements of the VM, while ESCP algorithm prefers to select the physical server with lowest resource usage, which will have bad impact on the step of VM consolidation. Both of the two algorithms don't tolerate SLA violation and they will invoke hot server removing function when a physical server becomes a hot server. However, IGGA allows all physical servers execute workloads out of their proceeding capacities and thus IGGA needs the smallest physical servers to process the same number of VM requests.

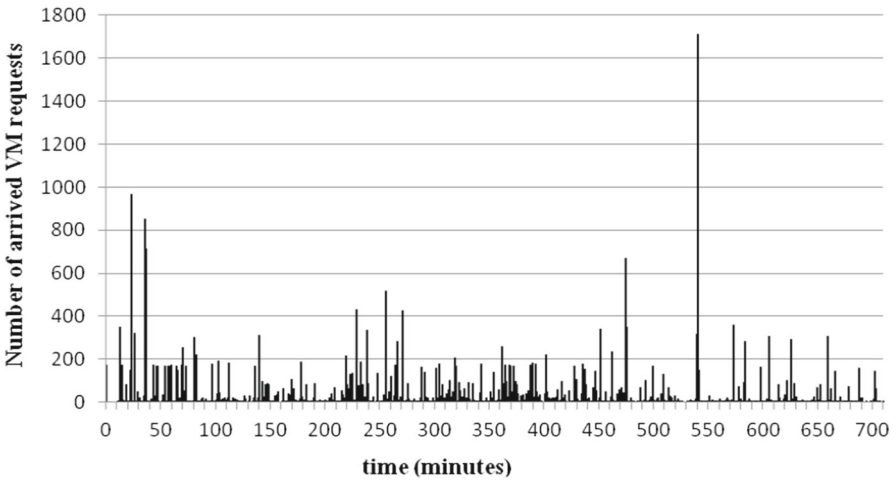

Fig. 3 Number of new arrival VM requests in each time slot

Table 2 Results obtained by different algorithms during the whole period (12 h)

Algorithms	MVC	ESCP	IGGA
Total no. of migrated VMs	1951	2258	**1879**
Total MC	**113,469**	194,712	**110,483**
Average MC	**58.01**	86.2	58.8

Bold values reflect the optimal value of each line

Otherwise, another phenomenon shown in Fig. 2 is that the results obtained by the three compared algorithms fluctuate heavily in different time slots. This is caused by the trend of new arrival VM requests to the data center. The number of VM requests submitted to the data center in each time slot is presented in Fig. 3. It can be seen easily that the number of active physical servers in each time slot has almost the same trend with the new arrival VM requests to the data center.

5.2.2 VM Migration

Generally, the VM migration related metrics are the total number of migrated VMs (TMV) in the whole experimental period, the total migration cost and the average migration cost. The total number of migrated VMs in the whole period equals the sum of migrated VMs in each time slot and so does the total migration cost. The average migration cost is the total migration cost divided by the total number of migrated VMs. These three metrics is mainly used to analysis the negative effective of VM migrations of different algorithms.

Table 2 shows the results of the three studied metrics obtained by the three compared algorithms during the whole experimental period. It can be seen that, the values of the three metrics obtained by MVC algorithm and IGGA algorithm are near the same and both of the two algorithms can both perform well on the three metrics. Specially, the

Table 3 SLA violation percentage of different algorithms during the whole period (12 h)

Algorithm	MVC	ESCP	IGGA
SVP	0.41%	**0.29%**	5.62%

Bold value reflects the optimal value of each line

results of TMV, total MC and average MC obtained by MVC algorithm are 13.6%, 41.7% and 32.7% respectively smaller than these of ESCP algorithm, which means that the proposed MVC algorithm can guarantee a relatively low impact on the performance of the cloud data center during the process of VM consolidations. The reasons are as follows: firstly, when choosing VMs to be migrated in HS-Removing step, the proposed MVC algorithm takes the factor of remaining runtime into account and if the remaining runtime of a selected VM is less than one time slot, then MVC algorithm doesn't migrate it, and thus can reduce some unnecessary migrations; secondly, when there is a need to migrate VM, the proposed MVC algorithm choose the VM with lowest migration cost from eligible VMs; however, ESCP algorithm selects the migrated VM with maximum occupied CPU/memory resource to avoid frequent SLA violation, and thus the average MC obtained by MVC algorithm is 44.6% lower than that obtain by ESCP algorithm. Therefore, considering the impacts of remaining runtime and migration cost in choosing VMs to be migrated can effectively avoid unnecessary migration and reduce additional negative influences caused by VM migrations during the process of VM consolidation.

5.2.3 SLA Violation Percentage (SVP)

SVP is the percentage of SLA violation time in the whole period and can be calculated by the total SLA violation time of all hot physical servers divided to the total running time of all active physical servers in the whole period T, which can be given by Eq. (11). The smaller the value is, the lower time that a physical server violates SLA.

$$SVP = \frac{\sum_{t=1}^{|T|} \sum_{j=1}^{m(t)} SLA_j(t)}{\sum_{t=1}^{|T|} m(t)} \tag{11}$$

In the conducted experiments, the whole period T is divided to $|T|$ time slots with equal length. Thus, in Eq. (11), the length of SLA violation for each physical servers equals to the times of the server's violating SLA.

Table 3 shows the results of SVP obtained by the compared algorithms during the whole experimental period. It can be seen that IGGA algorithm violates SLA more frequently than the other two algorithms and the SLA violation percentage is up to 5.62%. This is because IGGA algorithm doesn't consider the factor of service level agreement and allows a hot server executes workloads out of its proceeding capacities, and thus will cause higher probability to violate SLA. While MVC algorithm and ESCP algorithm will invoke VM migration immediately once a physical server becomes a hot server and violates SLA. Moreover, in MVC algorithm, physical servers are permitted

to violate SLA within a constrained range. The constraint is that the sustainable time of violating SLA for each physical server must not be longer than one time slot (1 min in experiments). With this constraint, once a physical server violate SLA, MVC algorithm firstly judges whether there is one or more VMs whose remaining runtimes are short than a time slot. If yes, this VM (or these VMs) can be completed within a time slot and the hot server will not violate SLA any longer. In this situation, the hot server can satisfy the SVP restrict, constraint C4 of the proposed optimization model (in Sect. 3.2), and there is no need to migrate any VM. That is why the SVP of the proposed MVC algorithm is slightly higher than that of ESCP algorithm.

6 Conclusions

Energy consumption has become one of the major concerns for the owners of CDCs. VM consolidation is seen as an effective approach to improve energy efficiency of CDCs, but it may depress the performance of migrated VMs, or even temporarily interrupting their services, during the migration period. As a result, these negative influences caused by VM migrations should not be ignored. Some studies focus on minimizing energy consumption while keeping the number of migrations as small as possible. However, these methods estimate the migration cost only in terms of the number of migrations and fail to take into account the inherent heterogeneity of costs caused by migrating different VMs. In order to decreasing the negative influences of VM migrations, this paper develops a MC-aware VM consolidation (MVC) algorithm by considering remaining runtime and migration cost when choosing VMs to be migrated, Simulation results show that, compared with some popular algorithms, MVC algorithm can decrease the migration cost and, at the same time guarantee the energy consumption within a certain low level.

Acknowledgements This work is partially supported by the National Natural and Science Foundation of China (Nos. 61472460, 61702162 and U1504607), Natural Science Project of the Education Department of Henan Province (Nos. 19A520021 and 17A520004), Program for Innovative Research Team (in Science and Technology) in University of Henan Province (No. 17IRTSTHN011), Science and Technology Project of Science and Technology Department of Henan Province (No. 172102110013), Plan for Nature Science Fundamental Research of Henan University of Technology (No. 2018QNJH26), Plan For Scientific Innovation Talent of Henan University of Technology (No. 2018RCJH07) and the Research Foundation for Advanced Talents of Henan University of Technology (2017025).

References

1. Armbrust, M., Fox, A., Griffith, R.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. Pract. Exp. **24**(13), 1397–1420 (2012)
3. Mastelic, T., Oleksiak, A., Claussen, H., et al.: Cloud computing: survey on energy efficiency. ACM Comput. Surv. **47**(2), 1–36 (2015)
4. Cao, J., Wu, Y., Li, M.: Energy efficient allocation of virtual machine in cloud computing environments based on demand forecast. In: Proceedings of the 7th International Conference on Grid and Pervasive Computing, pp. 137–151 (2012)

5. McCulloch, G.: The true cost of a data center becoming HPC compliant (2017). http://www.datacenterdynamics.com/content-tracks/servers-storage/the-true-cost-of-a-data-center-becoming-hpc-compliant/98890.article. Accessed 16 June 2018
6. Abada, A., St-Hilaire, M.: Renewable energy curtailment via incentivized inter-datacenter workload migration. In: Proceedings of 11th International Conference on Cloud Computing, pp. 143–157 (2018)
7. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**(5), 755–768 (2012)
8. Di, S., Kondo, D., Wang, C.: Optimization of composite cloud service processing with virtual machines. IEEE Trans. Comput. **64**(6), 1755–1768 (2015)
9. Dargie, W.: Estimation of the cost of VM migration. In: Proceedings of the 23rd IEEE International Conference on Computer Communication and Networks, pp. 1–8 (2014)
10. Ahmad, R.W., Gani, A., Hamid, S.H.A., et al.: A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J. Netw. Comput. Appl. **52**, 11–25 (2015)
11. Ferreto, T., Netto, M., Calheiros, R., De Rose, C.: Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. **27**(8), 1027–1034 (2011)
12. Varasteh, A., Goudarzi, M.: Server consolidation techniques in virtualized data centers: a survey. IEEE Syst. J. **11**(2), 772–783 (2017)
13. Guo, Z., Yao, W., Wang, D.: A virtual machine migration algorithm based on group selection in cloud data center. In: Proceedings of 15th IFIP International Conference on Network and Parallel Computing, pp. 24–36 (2017)
14. Liu, H., Jin, H., Xu, C.Z., Liao, X.: Performance and energy modeling for live migration of virtual machines. Cluster Comput. **16**, 249–264 (2013)
15. Xu, H., Liu, Y., Wei, W., Zhang, W.: Incentive-aware virtual machine scheduling in cloud computing. J. Supercomput. **74**(7), 3016–3038 (2018)
16. Beloglazov, A., Buyya, R.: Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. IEEE Trans. Parallel Distrib. Syst. **24**(7), 1366–1379 (2013)
17. Gutierrez-Garcia, J.O., Ramirez-Nafarrate, A.: Collaborative agents for distributed load management in cloud data centers using live migration of virtual machines. IEEE Trans. Serv. Comput. **8**(6), 916–929 (2015)
18. Xu, H., Yang, B.: Energy-aware resource management in cloud computing considering load balance. J. Inf. Sci. Eng. **33**(1), 1–16 (2017)
19. Wu, Q., Ishikawa, F., Zhu, Q., Xia, Y.: Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. IEEE Trans. Serv. Comput. (2016). https://doi.org/10.1109/tsc.2016.2616868
20. Sharma, N., Guddeti, R.M.: Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans. Serv. Comput. (2016). https://doi.org/10.1109/tsc.2016.2596289
21. Perumal, V., Subbiah, S.: Power-conservative server consolidation based resource management in cloud. Int. J. Netw. Manag. **24**(6), 415–432 (2014)
22. Mastroianni, C., Meo, M., Papuzzo, G.: Probabilistic consolidation of virtual machines in self-organizing cloud data centers. IEEE Trans. Cloud Comput. **1**(2), 215–228 (2013)
23. Marotta, A., Avallone, S.: A simulated annealing based approach for power efficient virtual machines consolidation. In: Proceedings of IEEE 8th International Conference on Cloud Computing, pp. 445–452 (2015)
24. Cui, L., Cziva, R., Tso, F.P., et al.: Synergistic policy and virtual machine consolidation in cloud data centers. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)
25. Zhao, H., Wang, J., Liu, F., et al.: Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Trans. Parallel Distrib. **29**(6), 1385–1400 (2018)
26. Fioccola, G.B., Donadio, P., Canonico, R., et al.: Dynamic routing and virtual machine consolidation in green clouds. In: Proceedings of IEEE International Conference on Cloud Computing Technology and Science, pp. 590–595 (2016)
27. Ye, K., Wu, Z., Wang, C., et al.: Profiling-based workload consolidation and migration in virtualized data centers. IEEE Trans. Parallel Distrib **26**(3), 878–890 (2015)
28. Wolke, A., Pfeiffer, C.: Improving enterprise VM consolidation with high-dimensional load profiles. In: Proceedings of the 2014 IEEE International Conference on Cloud Engineering, pp. 283–288 (2014)

29. Tao, F., Li, C., Liao, T., Laili, Y.: BGM-BLA: a new algorithm for dynamic migration of virtual machines in cloud computing. IEEE Trans. Serv. Comput. **9**(6), 910–925 (2016)
30. Mann, Z.Á.: Multicore-aware virtual machine placement in cloud data centers. IEEE Trans. Comput. **65**(11), 3357–3369 (2016)
31. Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R.: Virtual machine consolidation in cloud data centers using ACO meta-heuristic. In: Proceedings of European Conference on Parallel Processing, pp. 306–317 (2014)
32. Farahnakian, F., Ashraf, A., Pahikkala, T., et al.: Using ant colony system to consolidate VMs for green cloud computing. IEEE Trans. Serv. Comput. **8**(2), 187–198 (2015)
33. Jammal, M., Hawilo, H., Kanso, A., et al.: Mitigating the risk of cloud services downtime using live migration and high availability-aware placement. In: Proceedings of IEEE International Conference on Cloud Computing Technology and Science, pp. 578–583 (2016)
34. Google cluster-usage traces (version 2) (2014). http://code.google.com/p/googleclusterdata/. Accessed 16 Oct 2017
35. Murthy, M.K.M., Sanjay, H.A., Anand, J.: Threshold based auto scaling of virtual machines in cloud environment. In: Proceedings of 11th IFIP International Conference on Network and Parallel Computing, pp. 247–256 (2014)
36. Calheiros, R., Ranjan, R., Beloglazov, A., et al.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**, 23–50 (2011)

Affiliations

Heyang Xu[1] · Yang Liu[1] · Wei Wei[1] · Ying Xue[1]

Heyang Xu
xuheyang124@126.com

Wei Wei
nsyncw@126.com

Ying Xue
xueying_0422@126.com

[1] College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, Henan, China