IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 13, 2017, accepted December 21, 2017, date of publication December 28, 2017,
date of current version February 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2787735

A Game Based Consolidation Method of Virtual Machines in Cloud Data Centers With Energy and Load Constraints

LIANGMIN GUO, GUIYIN HU, YAN DONG, YONGLONG LUO, AND YING ZHU
Department of Computer Science and Technology, Anhui Normal University, Wuhu 241003, China
Anhui Provincial Key Laboratory of Network and Information Security, Wuhu 241003, China

Corresponding author: Liangmin Guo (lmguo@ustc.edu.cn).

This work was supported in part by the National Natural Science Foundation of China under Grant 61572036, Grant 61672039, Grant 61602009, and Grant 61772034, in part by the Natural Science Foundation of Anhui Province under Grant 1508085QF133, and in part by the Research Program of Anhui Province Education Department under Grant KJ2014A088.

ABSTRACT To reduce energy consumption and balance the resource load of physical machines (PMs) in cloud data centers, we present a game-based consolidation method of virtual machines (VMs) with energy and load (applied load) constraints. First, we test every measured value of the resource load using a t-test to filter outliers. Based on these values, the future resource load is forecast using gray theory. Second, all online PMs are grouped by the number of VMs on them and their future load values. Based on the groupings, a pre-processing algorithm for selecting destination PMs is proposed to determine a set of destination PMs for a VM awaiting migration. Finally, we select the final destination PM for the VM using game-based methods aimed at optimizing overall energy consumption. The experimental results show that our method can reduce energy consumption as well as balance loads without unnecessarily increasing the number of VM migrations.

INDEX TERMS Cloud data center, VM migration, load, energy, game.

I. INTRODUCTION

With the rapid development of information technology, information applications in enterprises are increasing, and their service requirements are constantly growing. An information platform that has been built based on traditional technology does not work for their growing business needs. The rise of cloud computing provides a better way to solve this problem. Virtualization technology [1], [2] is a core infrastructure component of cloud computing. Therefore, a virtual resource pool using virtualization technology, which is freely adjustable and can be configured on-demand [3], has become an urgent need of many enterprises.

Driven by user requirements, the number of VMs has increased so that new challenges to the resource scheduling technology of virtual machines have emerged. In a large-scale VM cluster, the number and the load of VMs changes constantly with user requirements. As more or all VMs on a PM (hereinafter referred to as a PM) are performing computational tasks, there is a strong possibility that resource contention will increase the execution time of tasks and reduce the quality of services. Meanwhile, some PMs may be underload, be idle, or be single resource-intensive; that is, all kinds of resources or some sort of resource in them is not being utilized effectively. In addition, when the VMs on PMs are not performing computing tasks, these computing resources are still taken up, and other tasks block access to them. The use of static resource management tends to cause resource waste or inadequacy, which results in more energy consumption or load imbalance. Furthermore, the energy consumption of a data center determines its power costs. Worldwide, the cost of power has been estimated to be more than $30 billion every year [4]. A Google data center consumes the energy equivalent to that used by a city such as San Francisco [5]. Therefore, the reduction in energy consumption and load balancing are primary areas of research. However, most research efforts focus only on balancing load [6]–[10], [12]–[15], [41], or only on saving energy [16]–[23], [25], [27]–[29], [39], [40]. Because these two goals contradict each other, less research has considered both load balancing and energy saving.

In this paper, inspired by the leaving and keeping processes in [34] and [35], we propose a game based consolidation

method of VMs in cloud data centers with energy and load constraints. We forecast the resource load using gray theory to reduce the delay of load throttling (i.e., timely adjust the load by migration in case of the higher degree of load balance). Then, we group PMs based on the number of VMs and the feature of the future load (i.e., higher load, imbalance load, and normal load) to help to determine the destination PM set of each VM awaiting migration with the goal of balancing loads. If a VM awaiting migration is on a PM with imbalance load, then we try to select another eligible PM with imbalance load as the destination PM. Lastly, we choose the final destination PM by estimating the energy variation of the VM before and after migration, and using game based methods aimed at optimizing overall energy consumption.

The main contributions of this paper are as follows: 1) a new energy consumption model for estimating the energy variability of the VM before and after migration, 2) a new method for selecting the destination PM of a VM awaiting migration, 3) a method for predicting future resource loads, and 4) a game based consolidation method of VMs in cloud data centers with energy and load constraints.

The rest of the paper is organized as follows. In Section II, we introduce related work. We note which models and algorithms for balancing load and saving energy exist. Afterwards, we state the problem and put forth an energy consumption model and a method of predicting resource load. Our game based consolidation strategy is proposed in Section IV. Section V analyzes the time complexity of our method. In Section VI, we evaluate our method. Finally, our conclusions and recommendations for future work are summarized in Section VII.

II. RELATED WORK

In this section, we review relevant literature on load balancing and energy saving.

A. LOAD BALANCING

In recent years, some scholars have proposed methods based on task scheduling [6]–[10]. In [10], a weighted round robin load balancing method was proposed, which considers both PM processing power and load and can balance the load performance when users log on simultaneously. These methods adjust the number of tasks performed by different PMs to change their loads. Most researchers have used methods based on VM live migration [11]. Wu *et al.* [12] proposed a method of prediction-based elastic load balancing resource management in cloud computing, which can dynamically add or delete VMs based on the applied load. Gutierrez-Garcia and Ramirez-Nafarrate [13] proposed an agent-based load balancing method. In this method, the agents can determine which VMs should be migrated, their destination PMs, and when they should be migrated. In [14], a method is introduced for the deployment and scheduling of VMs, which is based on a multi-attribute analysis to solve the problem of uneven loads among PMs. A resource scheduling strategy based on ant colony optimization is given in [15]. Li *et al.* [41] proposed a multiple-objective optimization method to balance the load of multiple resources across host machines and in each PMs.

B. ENERGY SAVING

To save energy, researchers have put forward two kinds of methods: one changes the processor frequency or voltage [16], [17], and the other reduces the number of online PMs [18]–[26], [40]. In [18] and [19], resource provisioning and allocation algorithms for energy-efficient management were proposed taking QoS expectations into account. A new CPU re-allocation algorithm was proposed that combines the DVFS concept with live migration techniques to improve the efficiency of energy management and adaptation with real-time service [20]. Van Do and Rotter [21] presented an allocation method of virtual machines based on the priority. Abda *et al.* [22] designed an energy-aware allocation mechanism that employs DNA-based scheduling strategies to minimize overall energy consumption. Mazumdar and Pranzo [23] presented a snapshot-based solution for the server consolidation problem to save energy; it considers issues such as reducing the number of VM migrations and consolidating the loads of running PMs. In [24], a three-dimensional virtual resource scheduling method for energy saving was proposed, which includes three stages: virtual resource allocation, virtual resource scheduling and virtual resource optimization. Ghribi *et al.* [25] presented an optimal allocation algorithm that uses a linear and integer formulation to resolve the replacement of VMs. Jin *et al.* [26] presented a resource-use-status-driven resource reconfiguration scheme to balance loads and save energy, which employs a greedy method to select the destination PM that owns enough idle resources and where the number of idle resources is most similar to the request of the VM awaiting migration. These methods all use VM migration to reduce the number of online servers. In [40], a custom branch-and-bound algorithm is proposed to save energy by reducing the sum of the number of active PMs and migrations.

In addition to these methods, a new energy-saving scheduling method was put forward in [27], which ranks PMs within the cloud based on their application-specific energy efficiency and assigns the whole load to the most energy-efficient machine while maintaining performance. Duan *et al.* [28] proposed an improved ant colony algorithm for saving energy, which includes a prediction model based on fractal mathematics and a scheduler based on an improved ant colony algorithm. [29] used virtualization technology to increase the utilization of the PMs and hence reduce the total number of active PMs. Ahvar *et al.* [39] proposed a VM placement method based on cost and carbon emission efficient in distributed clouds by a prediction-based A$*$ algorithm with Fuzzy Sets technique, which considers geographically varying energy prices and carbon emission rates as well as optimizing both network and server resources.

In the above methods, either only load balancing is considered or only energy saving is performed; little research

TABLE 1. Comparisons of the existing methods and our method.

Reference	Method	Problems addressed	Improvement/achievement	Weakness/limitations
[12]	Load balancing resource management algorithm	To provide with elastic management of server pool	Reduce the delay of applying for virtual machines	• Cause more waste of resources • Only take into account the average load of server pool
[13]	Agent-based load balancing heuristics	To balance load across PMs and minimize the number of VM migrations	Take into account both PM heterogeneity and VM resource usage heterogeneity	• Do not avoid the delay of load throttling • Do not take into account the imbalance of different resources load in a single PM • If there is no eligible destination PM, then the VM awaiting migration will not be migrated
[14]	AHP based load balancing approach	VM scheduling	Reduce the overall loss caused by dynamic load balancing	• Do not take into account the imbalance of different resources load in a single PM • If there is no eligible destination PM, then the VM awaiting migration will not be migrated
[15]	Interaction artificial bee colony based load balancing method	VM placement	Balance load and less execution time	• Do not avoid the delay of load throttling • Do not take into account the imbalance of different resources load in a single PM • Compare with only artificial bee colony algorithm
[41]	Multiple-objective optimization approach	Load balancing	Balance load of resources across PMs and in a single PM	• Do not avoid the delay of load throttling
[18-19]	Heuristics for dynamic reallocation of VMs	Energy efficient allocation of VMs	Reduce energy consumption	• Focus only on energy consumption
[20]	CPU reallocation algorithm	Efficiency of energy management	Reduce energy consumption and execution time	• Focus only on energy consumption
[21]	Analytic performance model	Energy aware allocation	Reduce energy consumption	• Focus only on energy consumption
[22]	DNA-based scheduling strategies	Task scheduling	Reduce energy consumption	• Focus only on energy consumption
[23]	Snapshot-based solution	Server consolidation	Reduce energy consumption	• Focus only on energy consumption
[24]	Virtual resource scheduling method	VM scheduling	Reduce energy consumption and balance load	• Do not take into account the imbalance of different resources load in a single PM • If any of VMs on a PM with the lowest CPU utilization cannot be migrated, then the migrations of VMs from the PM are canceled
[25]	Exact VM allocation algorithm	VM scheduling	Consolidation and reduce energy consumption	• Compare with only basic algorithm • Focus only on energy consumption
[26]	Resource-use-status-driven resource reconfiguration scheme	Energy and load aware consolidation of VMs	Reduce energy consumption and balance load	• Do not take into account the imbalance of different resources load in a single PM • Only reconfigure processor resource
[40]	Custom branch-and-bound algorithm	VM placement	Reduce energy consumption	• Focus only on energy consumption
Our method	Game based VM consolidation method	Energy and load aware consolidation of VMs	Balance load and reduce energy consumption	• The prediction accuracy for SCE 2 need to be further improved

on both methods together has been conducted. Therefore, we present a game based consolidation strategy of VMs in cloud data centers with energy and load constraints. Table 1 shows the comparisons of the existing methods, which are load balancing methods by migration and energy saving methods by reducing the number of online PMs, and our method.

III. PROBLEM DESCRIPTION AND MODELS

A. PROBLEM DESCRIPTION

Suppose that there are N PMs in a cloud data center: $p_1, p_2, \ldots, p_i, \ldots, p_N$. At some time, there are K VMs: $v_1, v_2, \ldots, v_j, \ldots, v_K$. We define a distribution matrix of VMs, denoted by $D = (d_{ij})_{N\times K}$. If VM v_j is placed in PM p_i, then let $d_{ij} = 1$; otherwise, $d_{ij} = 0$. Every column in the matrix D only has a 1-digit number, and the number of "1s" in line i is the number of VMs on PM p_i. To reduce energy consumption and balance the load, the problem of consolidating VMs is abstracted as the following problem.

$$Max \ (U_{all}(\cdot)) \tag{1}$$

$$s.t. \sum_{i=1}^{N} d_{ij} = 1, \quad j = 1, 2, \ldots, K \tag{2}$$

$$\sum_{j=1}^{K} d_{ij} > \lambda_1, \quad i = 1, 2, \ldots, N \tag{3}$$

$$u_{cpu_{p_i}} \leq \Omega_{cpu} \;\&\; u_{mem_{p_i}} \leq \Omega_{mem} \;\&\; u_{net_{p_i}} \leq \Omega_{net}, \quad i = 1, 2, \ldots, N \tag{4}$$

$$d_{ij} = \{0, 1\}, \quad i = 1, 2, \ldots, N, \; j = 1, 2, \ldots, K \tag{5}$$

In (1), $U_{all}(\cdot)$ is the total income after consolidation, which is the sum of the changes in the energy consumption before and after migrations during time T (the change is defined by (9)). We need to achieve the optimal global energy consumption. Constraint (2) ensures each VM belongs to only one PM. Constraint (3) ensures the number of VMs on a PM is greater than λ_1. If the number is not greater than λ_1, we must reduce the number of online PMs to save energy consolidation. Finally, Constraint (4) bounds the load of each PM. Ω_{cpu}, Ω_{mem}, and Ω_{net} are the upper bounds of resources (CPU, memory, and network bandwidth), respectively. $u_{cpu_{p_i}}$, $u_{mem_{p_i}}$, and $u_{net_{p_i}}$ are the future loads of the CPU, memory and network. If the future load of a PM cannot meet (4), then we must reduce its load by migration.

B. ENERGY CONSUMPTION MODEL

In addition to cooling and lighting, the energy consumed by a cloud data center is consumed mainly by the CPUs, memory, and other physical equipment on the PMs. The energy consumption of each piece of equipment is different. The CPU in a PM accounts for at least one-third of the total energy consumption [30]. The number depends on the CPU load: the higher the load is, the more energy is consumed. However, the energy consumption of the other equipment aside from the CPU is relatively stable and is only determined by whether the PM is turned on [36], [37].

At time t, the power (energy) consumption of a single PM is denoted by (6):

$$P(t) = P_{cpu}(t) + P_{other}(t). \tag{6}$$

where $P_{other}(t)$ is the sum of the power consumed by other equipment. It tends to stabilize after the PM starts. We assume this value is the same for any of the running PMs. $P_{cpu}(t)$ is the energy consumption of the CPU for a single PM, which can be calculated according to (7).

$$P_{cpu}(t) = P_{no-virtual}(t) + \frac{\beta_1}{T}\int_{t_0}^{t_0+T}\sum_{i=1}^{a} ur_{cpu}(v_i, t)dt + \frac{\beta_2}{T}\int_{t_0}^{t_0+T}\sum_{j=1}^{b} ur_{\overline{cpu}}(v_j, t)dt \tag{7}$$

where $P_{no-virtual}(t)$ is the energy consumption when there is no VM in the PM and has a constant value. $\frac{\beta_1}{T}\int_{t_0}^{t_0+T}\sum_{i=1}^{a} ur_{cpu}(v_i, t)dt$ is the average power consumption of a CPU-intensive VM during the time period $[t_0, t_0 + T]$, and $\frac{\beta_2}{T}\int_{t_0}^{t_0+T}\sum_{j=1}^{b} ur_{\overline{cpu}}(v_j, t)dt$ is the average power consumed by a low-CPU-intensive VM during the time period $[t_0, t_0 + T]$. $ur_{cpu}(v_i, t)$ and $ur_{\overline{cpu}}(v_j, t)$ represent the 100-times-higher CPU utilization of v_i and v_j, respectively. a and b represent the number of CPU-intensive VMs and low-CPU-intensive VMs. β_1 and β_2 are regulatory factors, and $\beta_1 + \beta_2 = 1$.

During time period T, a PM's energy consumption is accurately estimated by (8).

$$E = \int_{t_0}^{t_0+T} P(t)dt \tag{8}$$

Suppose that ΔE_v is the change in the energy consumption before and after migration, which can be computed by (9).

$$\Delta E_v = \int_{t_1-T}^{t_1} P_{s(v)}(t)dt + \int_{t_1-T}^{t_1} P_{d(\overline{v})}(t)dt - \int_{t_2}^{t_2+T} P_{s(\overline{v})}(t)dt - \int_{t_2}^{t_2+T} P_{d(v)}(t)dt - E_{s \xrightarrow{v} d} \tag{9}$$

where t_1 and t_2 are the start and end times of the migration. $\int_{t_1-T}^{t_1} P_{s(v)}(t)dt$ and $\int_{t_2}^{t_2+T} P_{s(\overline{v})}(t)dt$ represent the energy consumption of the source PM before and after VM v is migrated from source PM. If there is no VM on the PM and close it, then $\int_{t_2}^{t_2+T} P_{s(\overline{v})}(t)dt$ is zero. $\int_{t_1-T}^{t_1} P_{d(\overline{v})}(t)dt$ and $\int_{t_2}^{t_2+T} P_{d(v)}(t)dt$ represent the energy consumption of the destination PM before and after VM v is migrated to it. $E_{s \xrightarrow{v} d}$ represents the energy consumption during the process of VM v migration. It is calculated according to (10).

$$E_{s \xrightarrow{v} d} = \int_{t_1}^{t_2} \Delta P_s(t)dt + \int_{t_1}^{t_2} \Delta P_d(t)dt + E_{dest-on} \tag{10}$$

where $\int_{t_1}^{t_2} \Delta P_s(t)dt$ and $\int_{t_1}^{t_2} \Delta P_d(t)dt$ are the increased energy consumption of the source and destination PM, respectively. $E_{dest-on}$ is the increased energy consumption as a result of turning on a PM, which is a constant value. If we do not need to turn on a new PM as the destination PM when VM v is migrated, then $E_{dest-on} = 0$. $\Delta P_d(t)$ is usually constant too.

C. LOAD PREDICTION MODEL

The historical load of PMs will have an effect on their future load [31], so it is reasonable to predict their future load based on their historical load. In this paper, we use an unbiased GM(1, 1) model [32] to predict future load. This model is an exponential smoothing model without inherent bias. It uses a moving weighted average method. The corresponding weight is decreasing according to the index law. It gives higher weight to the load near the predicted point. Therefore, it can predict accurately. The procedure is as follows:

Step 1: Take the sequence $(u_1^{(0)}, u_2^{(0)}, u_3^{(0)}, \ldots, u_n^{(0)})$, where $u_i^{(0)}$ is the average load measured in the i-th time period. To improve the accuracy of the measured value, the load

in each interval will be measured k times, denoted by $\{u_{i_1}^{(0)}, u_{i_2}^{(0)}, \ldots, u_{i_k}^{(0)}\}$. Normal distribution is a common distribution. The limits of distributions such as Poisson distribution, the binomial distribution and other distributions are normal distribution. These distributions can be approximated by normal distribution. In addition, the trace of load has often multimodal distribution [31]. Therefore, we suppose that $\{u_{i_1}^{(0)}, u_{i_2}^{(0)}, \ldots, u_{i_k}^{(0)}\}$ is an approximate normal distribution. Then, we can use the t-test to eliminate errors in the data. If $u_{i_j}^{(0)}$ is questionable and $|u_{i_j}^{(0)} - \frac{1}{k-1}\sum_{x\neq j} u_{i_x}^{(0)}| / \sqrt{\frac{1}{k-1}\sum_{y\neq j}(u_{i_y}^{(0)} - \frac{1}{k-1}\sum_{x\neq j} u_{i_x}^{(0)})^2} > t(\alpha)$, then it is an outlier and will be eliminated. $t(\alpha)$ is a t-test threshold where α is a significance level; normally $\alpha = 0.05$. Using the above procedure, the average of the remaining data is the load during the time period.

Step 2: Obtain the sequence $(u_1^{(1)}, u_2^{(1)}, u_3^{(1)}, \ldots, u_n^{(1)})$ after one accumulation, where $u_m^{(1)} = \sum_{i=1}^{m} u_i^{(0)}$, $m = 1, 2, \ldots, n$.

Step 3: Compute the unbiased GM(1,1) model:

$$u_m^{(1)} = \alpha_1^{m-1} u_1^{(1)} + \frac{1-\alpha_1^{m-1}}{1-\alpha_1} \times \alpha_2, \quad m = 1, 2, \ldots, n. \tag{11}$$

Step 4: Compute the estimate of α_1 and α_2:

$$(\hat{\alpha}_1, \hat{\alpha}_2)^T = (B^T B)^{-1} B^T Y_n, \tag{12}$$

where

$$B = \begin{pmatrix} u_1^{(1)} & 1 \\ u_2^{(1)} & 1 \\ \cdots & \cdots \\ u_{n-1}^{(1)} & 1 \end{pmatrix}, \quad \text{and } Y_n = \begin{pmatrix} u_2^{(1)} \\ u_3^{(1)} \\ \cdots \\ u_n^{(1)} \end{pmatrix}.$$

Step 5: Based on step 4, we get the approximation of $u_m^{(1)}$:

$$\hat{u}_m^{(1)} = \hat{\alpha}_1^{m-1} u_1^{(1)} + \frac{1-\hat{\alpha}_1^{m-1}}{1-\hat{\alpha}_1} \times \hat{\alpha}_2. \tag{13}$$

Step 6: Suppose $\hat{u}_1^{(0)} = u_1^{(0)}$, then we get $\hat{u}_m^{(0)}$ by using (13):

$$\hat{u}_m^{(0)} = (\hat{\alpha}_1 - 1)\hat{\alpha}_1^{m-2} u_1^{(1)} + \hat{\alpha}_1^{m-2}\hat{\alpha}_2, \quad m = 2, 3, \ldots, n. \tag{14}$$

According to (14), we can calculate the load $\hat{u}_{n+1}^{(0)}$ in the $(n+1)^{\text{th}}$ time period, so we can determine the future load of the CPU, memory and network: u_{cpu}, u_{mem}, and u_{net}.

IV. GAME BASED CONSOLIDATION OF VMS WITH ENERGY AND LOAD CONSTRAINTS

Consolidation of VMs is only possible with the help of VM migration. Therefore, we need to consider the following three aspects: 1) the time of migration, which can be chosen based on numerous tests; 2) which VMs should be migrated. In this paper, migrated VMs usually occupy less memory, have maximum CPU utilization, or have minimum CPU utilization; 3) the selection of the destination PM, which is the basic problem this paper seeks to resolve, as discussed in the following section.

A. PREPROCESSING OF DESTINATION PM SELECTION

When selecting a destination PM, the remaining resources on the PM must be more than what the VM awaiting migration requires. To make the best use of the resources on the PM and reduce negative effects on its performance, which are generated when a VM is migrated to it, all current PMs are organized in ascending order of the number of VMs on them. Suppose the number of VMs on PM p_i is x_{p_i}. If $x_{p_i} \leq \lambda_1$, then the PM p_i is placed into set R_1, and if $\lambda_1 < x_{p_i} < \lambda_2$, then p_i is placed into set R_2. Otherwise, p_i is placed into set R_3. According to this method, all online PMs are divided three groups (R_1, R_2, and R_3). In general, the load of the PMs in R_1 is lower, and the VMs on them can be migrated to other PMs to reduce the number of online PMs. Some PMs in R_3 may have higher loads, and one or more VMs on them can be migrated to reduce the load.

The PMs in R_3, aside from the ones where migration was performed, are further divided into three groups according to the size relationship between the future load of resources (CPU, memory, and network bandwidth) and the upper bounds Ω_{cpu}, Ω_{mem}, and Ω_{net} of their load: $Group_{high}$, $Group_{imbalance}$, and $Group_{normal}$. $Group_{high}$ is a group with a higher load, $Group_{imbalance}$ is a group with a load imbalance, and $Group_{normal}$ is a normal group where the load of each PM is relatively balanced. The grouping algorithm is as follows.

Algorithm 1 Grouping of PMs

```
1: procedure Grouping_PM(R3)
2: for every PM p_i in R3 do
3:     if p_i is not performing migration then
4:         if u_{cpu_{p_i}} > Ω_cpu & u_{mem_{p_i}} > Ω_mem & u_{net_{p_i}} > Ω_net then
5:             p_i ∈ Group_high
6:         else
7:             if u_{cpu_{p_i}} > Ω_cpu | u_{mem_{p_i}} > Ω_mem | u_{net_{p_i}} > Ω_net then
8:                 p_i ∈ Group_imbalance
9:             else
10:                p_i ∈ Group_normal
11:            end if
12:        end if
13:    end if
14: end for
15: end procedure
```

Based on the set to which a source PM belongs, the preprocessing of a destination PM selection (algorithm 2) is performed to choose the preliminarily qualified PMs (line 3 in algorithm 3). Thus, we can determine a corresponding candidate set of destination PMs for each VM waiting for migration. If $V = \{v_1, v_2, \ldots, v_i, \ldots, v_z\}$ is a set of VMs

Algorithm 2 Pre-Processing of Destination PMs

procedure Pre-processing_PM_selection(v_i, p_s)
if $p_s \in R_3$ **then** //to reduce the load
  if $p_s \in Group_{imbalance}$ and $|Group_{imbalance}| > 1$ **then**
    s_i = **Filtering_PM**($Group_{imbalance}$ except for p_s)
    if $|s_i| = 0$ **then** //select PMs from R_2
      s_i = **Filtering_PM**(R_2)
      if $|s_i| = 0$ **then** //select PMs from R_1
        s_i = **Filtering_PM**(R_1)
      end if
    end if
  else
    if $p_s \in Group_{high}$ **then**
      s_i = **Filtering_PM**(R_2)
      if $|s_i| = 0$ **then**
        s_i = **Filtering_PM**(R_1)
      end if
    end if
  end if
end if
if $p_s \in R_1$ **then** //to save energy consumption
  s_i = **Filtering_PM**(R_2)
  if $|s_i| = 0$ **then**
    s_i = **Filtering_PM**(R_1 except for p_s)
  end if
end if
end procedure

Algorithm 3 Filtering of PMs in the Set R

procedure Filtering_PM(R) //to select qualified PMs
for every PM p in R **do**
  if $u_{cpu_p} + u_{cpu_{v_i}} < \Omega_{cpu}$ & $u_{mem_p} + u_{mem_{v_i}} < \Omega_{mem}$ & $u_{net_{up}} + u_{net_{v_i}} < \Omega_{net}$ **then**
    put p in destination PM candidate set s
  end if
end for
return s
end procedure

waiting for migration at time T, then their candidate sets are $s_1, s_2, \ldots, s_i, \ldots, s_z$. Suppose that p_s, which normally belongs to R_1 or R_3, is the source PM of v_i. Then, the pre-processing algorithm (algorithm 2) is as follows.

If $p_s \in Group_{imbalance}$ and $|Group_{imbalance}| > 1$, then select the eligible PMs from $Group_{imbalance}$ except for p_s, and put them into s_i. If there are no eligible PMs ($|s_i| = 0$), then select the eligible PMs from R_2, and perform the above processing. If $|s_i| = 0$, then select the eligible PMs from R_1. If $p_s \in Group_{high}$ or $p_s \in R_1$, then select the eligible PMs from R_2, and if there are no eligible PMs, then from R_1.

B. GAME MODEL

To achieve the best energy consumption as a whole, which occurs when every VM waiting for migration competes for destination PMs, we suppose that the problem belongs to a cooperative game.

For all $v_i \in V$, suppose the PM p_i that can maximize ΔE_v is selected from v_i's corresponding candidate set. If any $p_j (j = 1, 2, \ldots, i-1, i+1, \ldots, z)$ and p_i are not the same PM, then p_i is the destination PM of v_i. Otherwise, the corresponding destination PM should be chosen by a game with the aim of achieving the optimal global-energy-consumption. The game process is as follows:

Step 1: Every VM waiting for migration, that is $v_1, v_2, \ldots, v_j, \ldots, v_{k_1}$ $(k_1 \leq z)$, is a participant. The policy set of v_j is ST_j = {cooperation, competition}, where "cooperation" indicates v_j aims to achieve the optimal global-energy-consumption, and "competition" indicates that v_j aims to achieve the optimal energy consumption itself. Suppose the income $U_{v_j}(\cdot) = \Delta E_{v_j}$, which is generated by v_j's migration.

Step 2: During time T, the total income $U_{all}(\cdot) = \sum \Delta E_v$ after consolidation. With the aim of maximizing $U_{all}(\cdot)$, every VM waiting for migration can be matched with a PM that is the VM's destination PM.

Step 3: If a VM v_j fails to match with the PM that can maximize ΔE_{v_j}, then the other PM, which can make ΔE_v the second largest, will be selected from its candidate set s_j. If there is competition at this point, then perform step 1 and step 2. If it fails again, then the other PM will be selected, which can make ΔE_v the third largest change in energy consumption, and so on, until the destination PM is found. If such a PM is not found from s_j, then the migration fails, and a new candidate set needs to be established. If the candidate set is empty, then open a new PM as the destination PM. Otherwise, the p_j in the new candidate set may be the destination PM of v_j.

TABLE 2. Income matrix.

	v_j -cooperation	v_j -competition
v_i - cooperation	$(\Delta E_{v_i}, \Delta E'_{v_j})$ or $(\Delta E'_{v_i}, \Delta E_{v_j})$	$(q_{i_1}\Delta E_{v_i} + (1-q_{i_1})\Delta E'_{v_i}, q_{i_1}\Delta E'_{v_j} + (1-q_{i_1})\Delta E_{v_j})$
v_i -competition	$(q_{i_2}\Delta E_{v_i} + (1-q_{i_2})\Delta E'_{v_i}, q_{i_2}\Delta E'_{v_j} + (1-q_{i_2})\Delta E_{v_j})$	$(q_{i_3}\Delta E_{v_i} + (1-q_{i_3})\Delta E'_{v_i}, q_{i_3}\Delta E'_{v_j} + (1-q_{i_3})\Delta E_{v_j})$

For example, v_i and v_j contend for a PM p. Table 2 shows their income matrix. ΔE_{v_i} and ΔE_{v_j} are the incomes generated by v_i and v_j migrating to p independently. $\Delta E'_{v_i}$ and $\Delta E'_{v_j}$ are the incomes generated by migrating to suboptimal PM independently. q_{i_1}, q_{i_2}, and q_{i_3} are respectively probability when p is as v_i's destination PM in the following three kinds of conditions: 1) v_i cooperates, and v_j competes; 2) v_i competes, and v_j cooperates; 3) v_i and v_j both compete.

If $\Delta E_{v_i} + \Delta E'_{v_j} > \Delta E'_{v_i} + \Delta E_{v_j}$, then the total income of v_i and v_j is $\Delta E_{v_i} + \Delta E'_{v_j}$ when they cooperate with each other. Therefore, we can obtain (15), where the equality hold up if $q_{i_1} = 1, q_{i_2} = 1$, and $q_{i_3} = 1$. In addition, only when they cooperate is the total income largest. If $\Delta E_{v_i} + \Delta E'_{v_j} < \Delta E'_{v_i} + \Delta E_{v_j}$, the same procedure may be easily adapted to

obtain.

$$\begin{cases} \Delta E_{v_i} + \Delta E'_{v_j} \geq q_{i_1}\Delta E_{v_i} + (1-q_{i_1})\Delta E'_{v_i} + q_{i_1}\Delta E'_{v_j} \\ \qquad\qquad + (1-q_{i_1})\Delta E_{v_j} \\ \Delta E_{v_i} + \Delta E'_{v_j} \geq q_{i_2}\Delta E_{v_i} + (1-q_{i_2})\Delta E'_{v_i} + q_{i_2}\Delta E'_{v_j} \\ \qquad\qquad + (1-q_{i_2})\Delta E_{v_j} \\ \Delta E_{v_i} + \Delta E'_{v_j} \geq q_{i_3}\Delta E_{v_i} + (1-q_{i_3})\Delta E'_{v_i} + q_{i_3}\Delta E'_{v_j} \\ \qquad\qquad + (1-q_{i_3})\Delta E_{v_j} \end{cases} \tag{15}$$

C. GAME BASED CONSOLIDATION PROCESS OF VMS

Assume that there are z VMs waiting for migration in a certain time period T: $v_1, v_2, \ldots, v_i, \ldots, v_z$. The process of consolidation is as follows:

Step 1: All current PMs are sorted in ascending order of the number of VMs on them.

Step 2: According to the relationship between the number of VMs and thresholds (λ_1 and λ_2), these PMs are divided into three sets: R_1, R_2, and R_3.

Step 3: Calculate the future load of the CPU, memory and network of the PMs in R_3: u_{cpu}, u_{mem}, and u_{net}.

Step 4: The PMs in R_3 are grouped using algorithm 1.

Step 5: Based on the set which the source PM belongs to, we apply algorithm 2 to gain a candidate set of destination PMs for every VM waiting for migration: $s_1, s_2, \ldots, s_i, \ldots, s_z$.

Step 6: Compute the change in their energy consumption ΔE_v using energy consumption model. Suppose when the PM p_i is the destination PM of VM v_i, ΔE_{v_i} is largest. Then, do the following:

Step 6.1: If any p_j ($j = 1, 2, \ldots, i-1, i+1, \ldots, z$) and p_i are not the same, then v_i will be migrated to p_i.

Step 6.2: Otherwise, carry out the above game (in B of section IV) to determine the optimal global-energy-consumption.

V. COMPLEXITY ANALYSIS

The time cost of sorting the PMs according to the number of VMs depends on the number of online PMs, denoted by N, so the time cost is $O(N \log N)$. The cost of the step 2 in C of section IV is $O(N)$. The computing of the future load of CPU, memory and network is $O(n)$, where n is the length of sequence. The cost of algorithm 1 is $O(|R_3|)$. If we suppose $p_s \in Group_{imbalance}$, $p_s \in Group_{high}$, and $p_s \in R_1$ have equal probability of appearing, then the largest cost of gaining a candidate set for a VM is $O(\frac{1}{3}|Group_{imbalance}|+|R_2|+|R_1|) = O(|Group_{imbalance}|+|R_2|+|R_1|)$. If any $p_j (j = 1, 2, \ldots, i-1, i+1, \ldots, z)$ and p_i are not the same, then the time cost of determining the destination PM of v_i is $O(|s_1| + |s_2| + \ldots + |s_z| + z)$. Otherwise, if we can find the destination PM of v_i in s_i, the time cost of determining its destination PM is $O(|s_1| + |s_2| + \ldots + |s_z| + z + k_1 + k_2 + \ldots + k_y)$, where $k_1, k_2, \ldots, k_y$ are the number of VMs competing with v_i, and $y \leq |s_i|$. We assume the probability of the former case is x. Therefore, the total cost for z VMs is $O(NlogN + z[x(|s_1| + \ldots + |s_z| + z) + (1-x)(|s_1| + \ldots + |s_z| + z + k_1 + k_2 + \ldots + k_y)]) = O(N \log N + z^2 N)$.

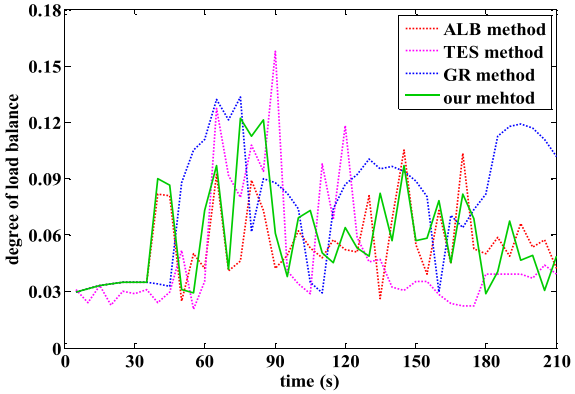


FIGURE 1. Degree of load balance (SCE 1).

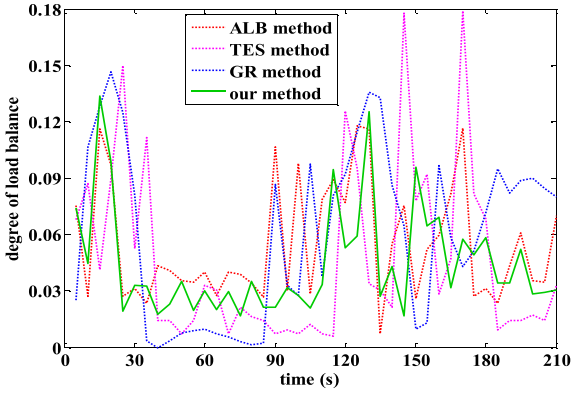


FIGURE 2. Degree of load balance (SCE 2).

VI. EXPERIMENTS

Experiments were performed to evaluate our proposed method, the agent-based load balancing method (abbreviated as ALB method) [13], the three-dimensional scheduling method for energy saving (abbreviated as TES method) [24] and the greedy method (abbreviated as GR method) [26]. To save experimental costs, we used CloudSim toolkit [33] to implement the experiments. CloudSim is one of the most commonly used cloud simulators. Researchers and developers can extend its functionality through programming. It can be used to help quantify and compare the performance of various resource scheduling and allocation strategies.

In the experimental evaluation, we mainly focused on analyzing the effectiveness of our proposed method from the following perspective: the degree of load balance, the number of migrations, the number of online PMs, and accuracy of prediction, etc. We considered two scenarios, a stable load (SCE 1) and an unstable load (SCE 2). In SCE 1, the load varies within a small range over a short time period. In SCE 2, the load varies within a wide range over a short time period.

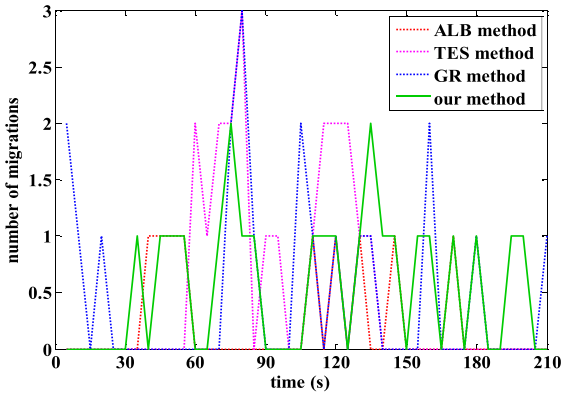


FIGURE 3. Number of migrations (SCE 1).

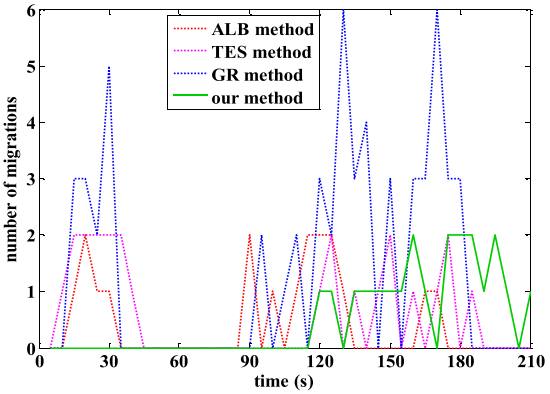
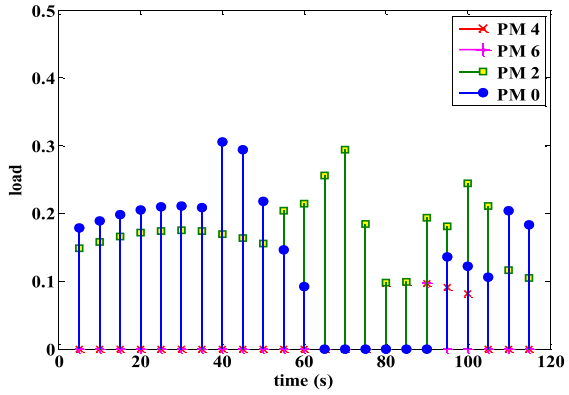
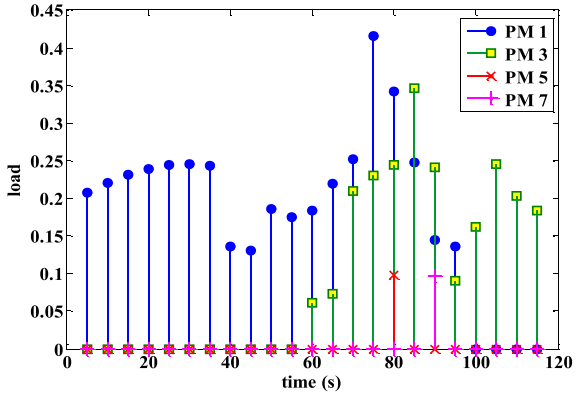


FIGURE 4. Number of migrations (SCE 2).



(a)



(b)

FIGURE 5. Migrations to mitigate resource competition.

TABLE 3. Distribution of VMs.

ID of PMs	ID of VMs
0	0, 3, 5
1	1, 4, 7
2	2, 6

TABLE 4. Distribution of tasks.

ID of VMs	ID of tasks
0	0, 8, 16
1	1, 9, 17
2	2, 10, 18
3	3, 11, 19
4	4, 12
5	5, 13
6	6, 14
7	7, 15

A. PARAMETER SETTINGS

Some experimental parameter settings are as follows.

1) There are eight PMs in the experiments, three of which are activated in the beginning. In addition, there are eight virtual machines and twenty cloud tasks. Based on the CloudSim, the configuration for CPU, memory and network bandwidth of each PM is 1000 MIPS, 2 GB, and 10000 Mbps. The configuration for CPU, memory, and network bandwidth of each VM is 300 MIPS, 512 MB, and 1000 Mbps. In the beginning, the distribution of the virtual machines and the tasks is as shown in Table 3 and Table 4. For example, the virtual machine 0, 3, and 5 are on PM 0. Tasks 0, 8, and 16 are run on virtual machine 0.
2) Based on the capacity of a PM and the resource allocated to VMs on the PM, we roughly set λ_1 and λ_2 to 1 and 3 respectively.
3) In the experiments the CPU and network load we generate fluctuates between 0 and 0.5. The memory load fluctuates between 0 and 0.2. Therefore, we suppose Ω_{cpu}, Ω_{mem}, and Ω_{net} are 0.3, 0.1, and 0.3 respectively. That is, when $u_{cpu_{p_i}} > 0.3$ & $u_{mem_{p_i}} > 0.1$ & $u_{net_{p_i}} > 0.3$, the PM p_i belongs to $Group_{high}$.
4) The load of PM p_i in R_1 satisfies the following conditions: $u_{cpu_{p_i}} < 0.15$ & $u_{mem_{p_i}} < 0.03$ & $u_{net_{p_i}} < 0.15$.
5) T is 5 seconds.
6) The length of time of every migration is 3 seconds.
7) To reduce the consolidation error, β_1 and β_2 are 0.5 [38].
8) $P_{no-virtual}(t)$ is 20 watts, and $P_{other}(t)$ is 30 watts.

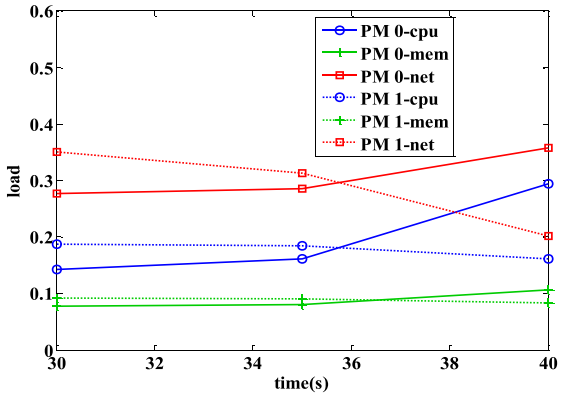


FIGURE 6. Processing for the group with load imbalance.

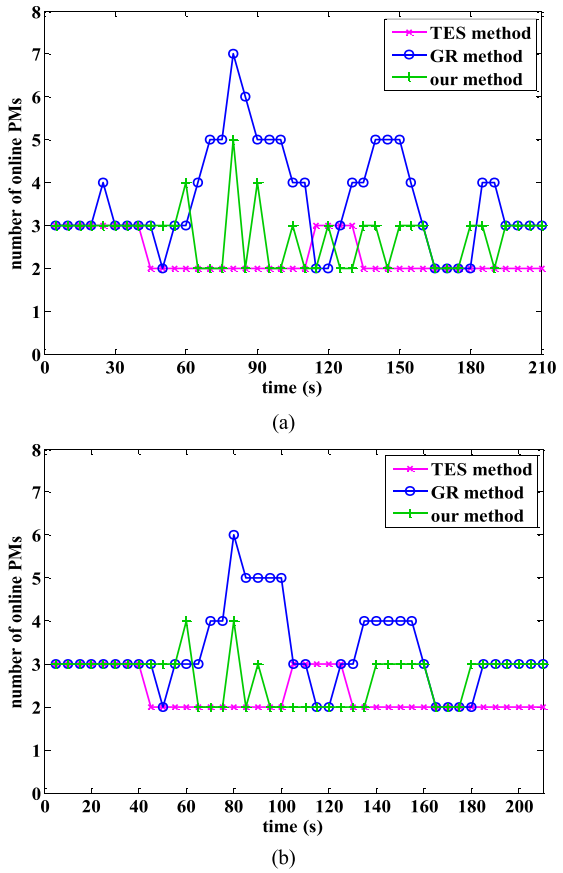


FIGURE 7. Number of online PMs (SCE 1).

9) If the historical load data are sufficient when predicting the future load, set $n = 5$; if the historical data are limited (less than 5), set $n = 2$; otherwise, we don't predict, and deal with the actual load instead.

B. ANALYSIS AND COMPARISON

1) DEGREE OF LOAD BALANCE

We use the standard deviation of the load to express the degree of load balance (b_degree_{load}), as shown in (16). $Load_i$ is the load of PM i, and N is the number of online PMs. The smaller the degree of load balance, the more balanced the load. Fig. 1 and Fig. 2 show the load balance of our method, the ALB method, the TES method and the GR method. From Fig. 1 and Fig. 2, we can see that our method balances the loads better. This is because in our method the PMs are grouped according to their loads. Then, the loads of the corresponding PMs in different groups are adjusted by migration, with the selection of the destination PMs based on the groupings. Table 5 shows the mean value and standard deviation of the degree of load balance. From Table 5 we can see that our method is slightly better than other three methods, and the degree of load balance is relatively stable. Only in SCE 1, our method is slightly inferior to the ALB method. In addition, although the mean of the TES method is lower than our method in SCE 1, its standard deviation is larger than our method; that is to say, the degree of load balance for our

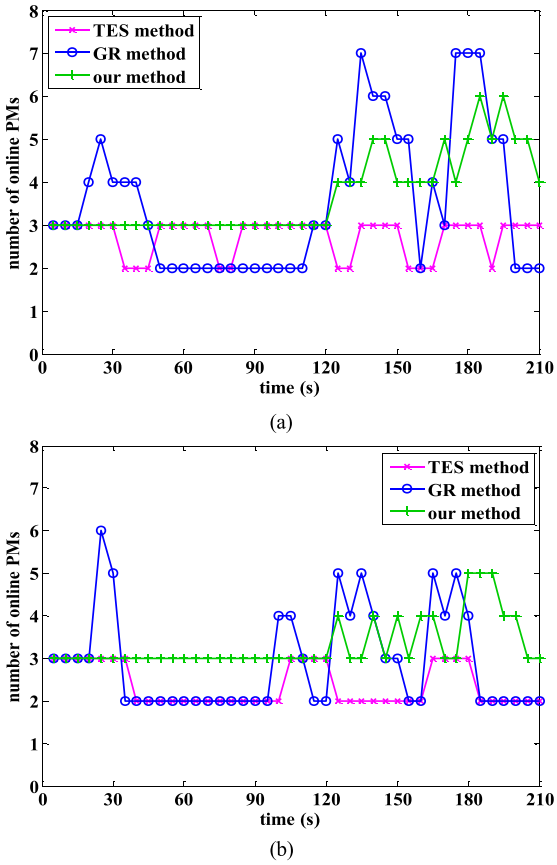


FIGURE 8. Number of online PMs (SCE 2).

TABLE 5. Mean value and standard deviation of degree of load balance

		mean value	standard deviation
SCE 1	ALB method	0.056	0.019
	TES method	0.049	0.032
	GR method	0.078	0.033
	our method	0.059	0.025
SCE 2	ALB method	0.056	0.031
	TES method	0.046	0.045
	GR method	0.060	0.044
	our method	0.045	0.028

TABLE 6. Some migrations.

time	migrations	trigger condition
35-38 s	VM 1 on PM 1 is migrated to PM 0.	PM 1 belongs to the group with load imbalance.
45-48 s	VM 0 on PM 0 is migrated to PM 1.	The load of PM 0 is high, and the load of PM 1 is low.
50-53 s	VM 3 on PM 0 is migrated to PM 2.	The load of PM 0 is high, and the load of PM 2 is low.
55-58 s	PM 3 is activated, and VM 5 on PM 0 is migrated to PM 3.	PM 0 belongs to the group with load imbalance.
63-66 s	VM 1 on PM 0 is migrated to PM 3, and then PM 0 is shut down.	The load on PM 0 and 3 is consolidated.
70-73 s	VM 2 on PM 2 is migrated to PM 1.	PM 1 and 2 belong to the group with load imbalance.
75-78 s	PM 4 and 5 are activated. VM 4 on PM 1 is migrated to PM 4, and VM 6 on PM 2 is migrated to PM 5.	PM 1 belongs to the group with load imbalance. The load of PM 2 is high.
80-83 s	VM 7 on PM 1 is migrated to PM 3.	The load of PM 1 is high, and the load of PM 3 is low.
83-86 s	VM 6 on PM 5 is migrated to PM 2, and then PM 5 is shut down.	The load on PM 2 and 5 is consolidated.
85-88 s	PM 6 and 7 are activated. VM 0 on PM 1 is migrated to PM 6. In addition, VM 5 on PM 3 is migrated to PM 7.	PM 1 and 3 belong to the group with load imbalance.
90-93 s	PM 0 is activated. VM 1 on PM 3 is migrated to PM 0.	The load on PM 3 is high.
93-96 s	VM 5 on PM 7 is migrated to PM 2, and VM 0 on PM 6 is migrated to PM 3. Then, PM 6 and 7 are shut down.	The load on PM 2 and 7 is consolidated, and the load on PM 3 and PM 6 is consolidated.
98-101 s	VM 2 on PM 1 is migrated to PM 3. Then, PM 1 is shut down.	The load on PM 1 and 3 is consolidated.
103-106 s	VM 4 on PM 4 is migrated to PM 0. Then, PM 4 is shut down.	The load on PM 0 and 4 is consolidated.
105-108 s	VM 3 on PM 2 is migrated to PM 0.	The load of PM 2 is high, and the load of PM 0 is low.
115-118 s	PM 5 is activated. VM 7 on PM 3 is migrated to PM 5.	PM 3 belongs to the group with load imbalance.

method is relatively stable.

$$b\_degree_{load} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(Load_i - \frac{1}{N}\sum_{i=1}^{N} Load_i)^2} \quad (16)$$

2) MIGRATION OF VMS

Fig. 3 and Fig. 4 show the number of migrations of the above four methods during the period 0 to 210 seconds. In SCE 1, the total number of migrations is 9 by the ALB method, the total number of migrations is 21 by the TES method, the total number of migrations is 20 by the GR method, and the total number of migration is 23 by our method. In SCE 2, the total number of migrations is 18 by the ALB method, the total number of migrations is 24 by the TES method, the total number of migrations is 57 by the GR method, and the total number of migration is 21 by our method. That is, for both scenarios the number of migrations for our method is similar, but the number of migrations for the other three methods is different. The results indicate the number of VM migrations for our method is stable in both scenarios. This is because our method can select the destination PMs for VMs more accurately. We can also see that the total number of migrations for the ALB method is less than our method in both scenarios. This is because an overload PM keeps a VM awaiting migration for the ALB method when the other online PMs will not accept the VM. Furthermore, we can also see that our method can timely adjust the load by migration in case of the higher degree of load balance.

Fig. 5 shows the actual load variation of some PMs for some migrations. To observe the variation clearly, they are shown in Fig. 5 (a) and Fig. 5 (b). When the predicted load of a PM exceeds the threshold, the load is transferred by migrating it at the appropriate time in order to mitigate resource competition. Some migrations and the trigger condition for the migration during this period are as shown in Table 6. During the period of 45 to 53 seconds, the load of PM 0 is adjusted twice. This is because our method adjusts PM loads depending on the predicted value of load, and there are errors between the predicted values and the actual values. Fig. 6 shows the load adjustment of some PMs with a load imbalance. At 35 seconds, a migration on PM 1 is occurring. Table 7 shows the load of PM 0 and PM 1. We know PM 1 belongs to the group with a load imbalance, and PM 0 belongs to the set R_2. According to the above game based consolidation strategy, some VM on PM 1 needs to migrate to PM 0.

TABLE 7. Load of PM 0 and PM 1.

	at 30 s	at 35 s	at 40 s
PM 0-cpu	0.142916	0.160555	0.294927
PM 0-mem	0.077076	0.080464	0.106767
PM 0-net	0.277359	0.296000	0.357456
PM 1-cpu	0.187499	0.184444	0.161834
PM 1-mem	0.092133	0.090156	0.082723
PM 1-net	0.350014	0.313333	0.222051

3) NUMBER OF ONLINE PMS

Fig. 7 and Fig. 8 show the number of online PMs during the period 0 to 210 seconds in both scenarios respectively. In Fig. 7(a) and Fig. 8(a), our method prescribes the new PMs (i.e., they were turned on recently) can be consolidated after 25 seconds. The reason for the delay (i.e., 25 seconds) is so adequate historical load data can be obtained for

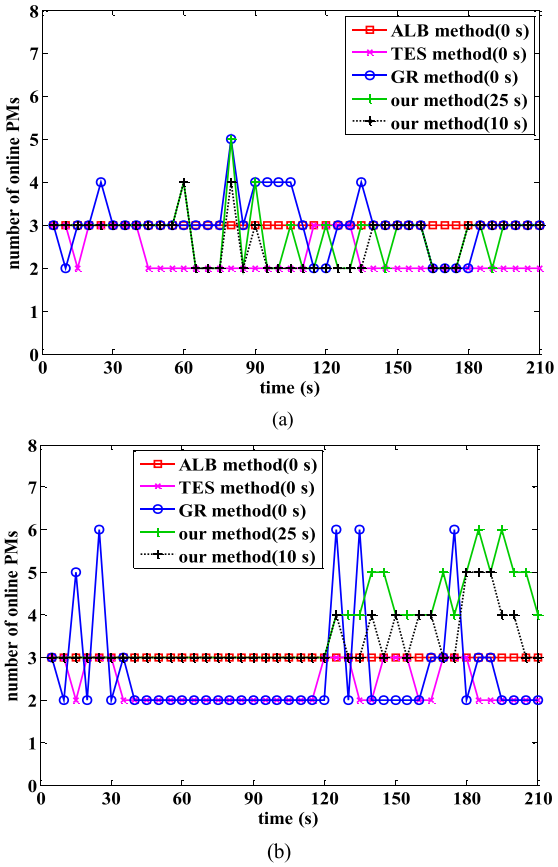


FIGURE 9. Number of online PMs. (a) SCE 1. (b) SCE 2.

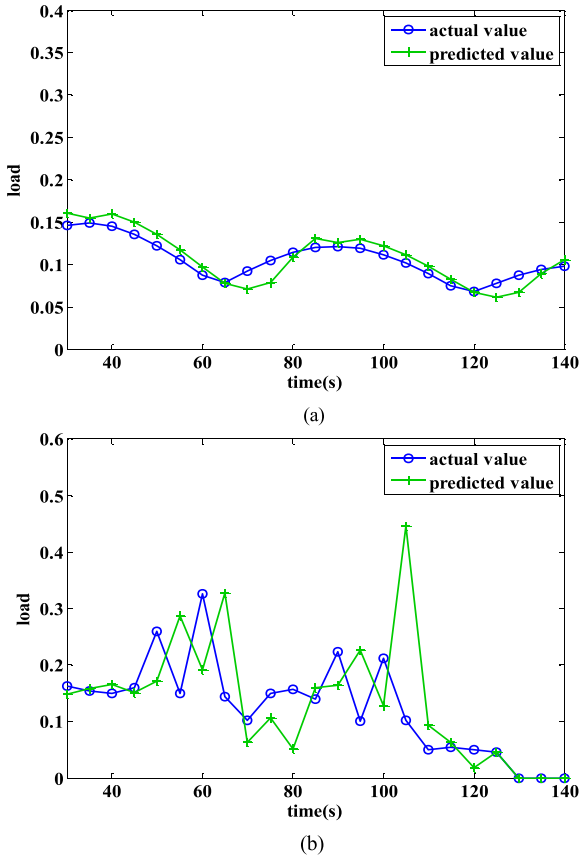


FIGURE 10. Predicted values and actual values of the load. (a) SCE 1. (b) SCE 2.

TABLE 8. Mean value and standard deviation of online PMs.

		delay	mean value	standard deviation
SCE 1	ALB method	0 s	3	0
	TES method	25 s	2.27	0.45
		10 s	2.30	0.46
		0 s	2.25	0.43
	GR method	25 s	3.60	1.18
		10 s	3.26	0.93
		0 s	3.02	0.64
	our method	25 s	2.74	0.66
		10 s	2.67	0.56
SCE 2	ALB method	0 s	3	0
	TES method	25 s	2.74	0.44
		10 s	2.36	0.48
		0 s	2.33	0.47
	GR method	25 s	4.00	1.66
		10 s	3.00	1.17
		0 s	3.00	1.21
	our method	25 s	4.00	0.91
		10 s	3.00	0.60

TABLE 9. Accuracy of predicted value.

	SCE 1	SCE 2
MAE	0.011	0.065
RMSE	0.012	0.102

prediction. In Fig. 7(b) and Fig. 8(b), we adjusted the delay to 10 seconds to adapt to the following situation: the load of a new PM is lower. If its load is lower, it can be transferred by migration. When a PM is idle, it can be turned off to save energy. In addition, in 10 seconds, we can get two load data updates for prediction with our method. From Fig. 7 and Fig. 8, we can see that the number of online PMs using our method is relatively stable. This is because the selection of the destination PMs depends on the result of grouping and the game, and our method considers the PMs with load imbalance. However, our method is slightly inferior to the TES method. This is because our method considers the balance of load. For the ALB method, because a new PM is not turned on when no online PM would accept the VM awaiting migration, the delay is not applicable.

For the GR method and the TES method, the delay is not necessary, as it uses the actual load to consolidate. Therefore, we compared our method (the delay is 25 s and 10 s respectively) to the GR method and the TES method (the delay is 0 s). The results are as shown in Fig. 9. From Fig. 9, we can see that our method works better than the GR method when the delay is 10 s. For the ALB method, the number of online

PMs remains constant. The reason has been introduced in the previous paragraph. For the TES method, the number of online PMs is lower than our method. This is because our method considers the balance of load. Table 8 shows the mean value and standard deviation of online PMs. It can also be seen our method performs better than the GR method, but is slightly inferior to the TES method.

4) PREDICTION OF LOAD

Fig. 10 shows the predicted and actual values of the load. From Fig. 10, we can see the difference between them is not large. Table 9 shows the accuracy of prediction. The prediction accuracy was computed using (17) and (18), where $actl_i$ and $pred_i$ represent the actual value and the predicted value at the i-th test point respectively, and X is the total number of test points. *MAE* is the mean absolute error of the predicted value. *RMSE* is the root mean squared error of the predicted value. From Table 9, we see the prediction accuracy is high.

$$MAE = \frac{1}{X}\sum_{i=1}^{X}|actl_i - pred_i| \tag{17}$$

$$RMSE = \sqrt{\frac{1}{X}\sum_{i=1}^{X}(actl_i - pred_i)^2} \tag{18}$$

VII. CONCLUSIONS AND FUTURE WORKS

This paper proposed a game based consolidation method of virtual machines in cloud data centers with energy and load constraints in a cloud data center. We first predict the future load values of resources using gray theory to timely adjust the load by migration in case of the higher degree of load balance. PMs are grouped according to the number of VMs and their future load to help to select destination PMs. Then, we use a pre-processing algorithm for destination PM selection to determine a destination PM set of a VM waiting for migration. Finally, we select the final destination PM through the game based method aimed at the optimal overall energy consumption. Experiments show that our method can reduce energy consumption as well as balance loads, without unduly increasing the number of VM migrations.

Our method does not consider the relevance of the VMs. For example, the VMs that need to communicate with each other should be placed in the same PM or adjacent PMs. We will further optimize the migration cost and improve the prediction accuracy for SCE 2 in future work. The cost is connected with the migration time that is spent transferring the memory data. In the future, we elaborate the experimental environments and use continuous real-time load to further improve the accuracy of the experimental data.

REFERENCES

[1] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun.*, Dalian, China, Sep. 2008, pp. 5–13.

[2] P. Barham *et al.*, "Xen and the art of virtualization," in *Proc. 19th ACM Symp. Operat. Syst. Principles*, Bolton Landing, New York, USA, 2003, pp. 164–177.

[3] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *Computer*, vol. 37, no. 11, pp. 68–74, 2004.

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

[6] D. Tang, H. Jin, and Y. Zhang, "Performance evaluation of dynamic load balancing system for cluster," *Chin. J. Comput.*, vol. 27, no. 6, pp. 803–811, 2004.

[7] J. Deng, Y. Zhao, and H. Yuan, "A service revenue-oriented task scheduling model of cloud computing," *J. Inf. Comput. Sci.*, vol. 10, no. 10, pp. 315–316, 2013.

[8] S. T. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," *Perform. Eval.*, vol. 81, pp. 20–39, Nov. 2014.

[9] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. 6th Chinagrid. Conf.*, 2011, pp. 3–9.

[10] S. Chen, Y. Chen, and S. Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services," *Comput. Electr. Eng.*, vol. 58, pp. 154–160, Feb. 2017.

[11] X. Song, J. Shi, R. Liu, J. Yang, and H. Chen, "Parallelizing live migration of virtual machines," in *Proc. 9th ACM Int. Conf. Virtual Execution Environ.*, Houston, TX, USA, 2013, pp. 85–95.

[12] H. Wu, C. Wang, and J. Xie, "TeraScaler ELB an algorithm of prediction-based elastic load balancing resource management in cloud computing," in *Proc. 27th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Barcelona, Spain, 2013, pp. 649–654.

[13] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Agent-based load balancing in cloud data centers," *Cluster Comput.*, vol. 18, no. 3, pp. 1041–1062, 2015.

[14] Z. Wei, G. Xiaolin, L. Jiancai, W. Gang, and D. Min, "Deployment and scheduling of virtual machines in cloud computing: An ahpapproach," *J. Xi'an Jiaotong Univ.*, vol. 47, no. 2, pp. 28–33, 2013.

[15] J. S. Pan, H. Wang, H. Zhao, and L. Tang, "Interaction artificial bee colony based load balance method in cloud computing," in *Proc. 8th Int. Conf. Genet. Evol. Comput.*, Nanchang, China, 2014, pp. 49–57.

[16] Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," *Future Generat. Comput. Syst.*, vol. 50, pp. 62–74, Sep. 2015.

[17] C. M. Wu, R. S. Chang, and H. Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generat. Comput. Syst.*, vol. 37, pp. 141–147, Jul. 2014.

[18] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generat. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[19] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, Melbourne, VIC, Australia, May 2010, pp. 577–578.

[20] W. Chawarut and L. Woraphon, "Energy-aware and real-time service management in cloud computing," in *Proc. 10th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol.*, Krabi, Thailand, 2013, pp. 1–5.

[21] T. Van Do and C. Rotter, "Comparison of scheduling schemes for on-demand IaaS requests," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1400–1408, 2012.

[22] S. K. Abda, S. A. R. Al-Haddadb, F. Hashimb, A. B. H. J. Abdullahc, and S. Yussof, "An effective approach for managing power consumption in cloud computing infrastructure," *J. Comput. Sci.*, vol. 21, pp. 349–360, Jul. 2017.

[23] S. Mazumdar and M. Pranzo, "Power efficient server consolidation for cloud data center," *Future Generat. Comput. Syst.*, vol. 70, pp. 4–16, May 2017.

[24] W. Zhu, Y. Zhuang, and L. Zhang, "A three-dimensional virtual resource scheduling method for energy saving in cloud computing," *Future Generat. Comput. Syst.*, vol. 69, pp. 66–74, Apr. 2017.

[25] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, Delft, The Netherland, 2013, pp. 671–678.
[26] H. Jin, L. Deng, S. Wu, X. Shi, and L. Zhou, "Automatic poweraware reconfiguration of processor resource in virtualized clusters," *J. Comput. Res. Develop.*, vol. 48, no. 7, pp. 1123–1133, 2011.
[27] Q. Zhang, G. Metri, S. Raghavan, and W. Shi, "RESCUE: An energy-aware scheduler for cloud environments," *Sustain. Comput.-Inf.*, vol. 4, no. 4, pp. 215–224, 2014.
[28] H. Duan, C. Chen, G. Min, and Y. Wu, "Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems," *Future Generat. Comput. Syst.*, vol. 74, pp. 142–150, Sep. 2017.
[29] V. Ebrahimirad, M. Goudarzi, and A. Rajabi, "Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers," *J. Grid Comput.*, vol. 13, no. 2, pp. 233–253, 2015.
[30] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. 1st ACM Symp. Cloud Comput.*, New York, NY, USA, 2010, pp. 39–50.
[31] P. A. Dinda, "The statistical properties of host load,"İ in *Proc. Int. Workshop Lang., Compilers, Run-Time Syst. Scalable Comput.*, 1998, pp. 319–334.
[32] T. Y. Pai *et al.*, "Comparisons of grey and neural network prediction of industrial park wastewater effluent using influent quality and online monitoring parameters," *Environ. Monitor. Assessment*, vol. 146, no. 1, pp. 51–66, 2008.
[33] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services," Grid Comput. Distrib. Syst. Lab., Univ. Melbourne, Melbourne, VIC, Australia, Tech. Rep. GRIDS-TR-2009-1, Mar. 2009.
[34] L. Zhou, "QoE-driven delay announcement for cloud mobile media," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 84–94, Jan. 2017.
[35] L. Zhou, "On data-driven delay estimation for media cloud," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 905–915, May 2016.
[36] Y. Y. Chen, A. Das, W. B. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proc. Int. Conf. Meas. Modeling Comput. Syst.*, Banff, AB, Canada, 2005, pp. 303–314.
[37] J. Song, T. T. Li, Z. X. Yan, J. Na, and Z. L. Zhu, "Energy-efficiency model and measuring approach for cloud computing," *J. Softw.*, vol. 23, no. 2, pp. 200–214, 2012.
[38] P. Xiao, D. Liu, and X. Qu, "An virtual machine scheduling algorithm based on energy consumption ratio model in cloud computing," *Acta Electronica Sinica*, vol. 43, no. 2, pp. 305–311, 2015.
[39] E. Ahvar, S. Ahvar, Z. A. Mann, N. Crespi, J. Garcia-Alfaro, and R. Glitho, "CACEV: A cost and carbon emission-efficient virtual machine placement method for green distributed clouds," in *Proc. 13th IEEE Int. Conf. Services Comput.*, San Francisco, CA, USA, Jun. 2016, pp. 275–282.
[40] D. Bartók and Z. Á. Mann, "A branch-and-bound approach to virtual machine placement," in *Proc. 3rd Int. Conf. HPI Cloud Symp. 'Operat. Cloud'*, 2015, pp. 49–63.
[41] R. Li, Q. Zheng, X. Li, and J. Wu, "A novel multi-objective optimization scheme for rebalancing virtual machine placement," in *Proc. 9th IEEE Int. Conf. Cloud Comput.*, San Francisco, CA, USA, Jun./Jul. 2016, pp. 710–717.

LIANGMIN GUO received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China, in 2011. Since 2013, she has been an Associate Professor with the School of Mathematics and Computer Science, Anhui Normal University. She is currently a master's student Supervisor of Anhui Normal University. Her research interests include cloud computing, information security, and recommender system.

GUIYIN HU received the M.S. degree from the School of Computer Science, Nanjing University of Posts and Telecommunications, China, in 2010. He is currently a Lecturer with the Department of Software Engineering, Anhui Normal University. His research interests include information security and virtual tours.

YAN DONG received the M.S. degree in economics from the University of International Business and Economics, Beijing, China, in 2006. She is a Research Associate with the Experiment Center, School of Mathematics and Computer Science, Anhui Normal University. Her research interests are E-commerce and data mining.

YONGLONG LUO received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China in 2005. Since 2007, he has been a Professor with the School of Mathematics and Computer Science, Anhui Normal University. He is currently the Ph.D. Supervisor of Anhui Normal University. He is the Director of the Engineering Technology Research Center of Network and Information Security. His research interests include information security and spatial data processing.

YING ZHU is currently pursuing the master's degree with the School of Mathematics and Computer Science, Anhui Normal University, China. Her main research interests include space query and service evaluation.