Received March 17, 2018, accepted May 7, 2018, date of publication May 16, 2018, date of current version June 26, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2835670

Enhancing Energy-Efficient and QoS Dynamic Virtual Machine Consolidation Method in Cloud Environment

YAQIU LIU[1], XINYUE SUN[1], WEI WEI[2], (Senior Member, IEEE), AND WEIPENG JING[1], (Member, IEEE)
[1]College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, China
[2]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

Corresponding author: Wei Wei (taneo@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 31370768, in part by the Natural Science Foundation of the Hei Longjiang Province of China under Grant F2017001, and in part by the Harbin Science and Technology Bureau Science and Technology Innovation Talents under Grant 2014RFQXJ132.

ABSTRACT Virtual machine (VM) consolidation techniques are a means to improve energy efficiency and the utilization of cloud data center resources. However, aggressive VM consolidation approaches lead to physical host over-utilization and generate massive undesired VM migrations, which cause degradation of the performance of both the hosts and the VM. Additionally, it has been a significant challenge to improve energy efficiency and resource utilization in the data center while delivering services with guaranteed quality of service (QoS). To address the problem, we propose an enhancing energy-efficient and QoS dynamic virtual machine consolidation (EQVC) method, which consists of four algorithms that correspond to different stages in VM consolidation. In this approach, we select redundant VMs from the hosts before they overload and migrate the VMs to other hosts to save energy and guarantee QoS requirements. We also introduce a host-model with adaptive reserved resources to prevent re-overload of hosts. To prove the effectiveness of our proposed method and algorithms, we experiment under different workload traces from a real system. The experimental results demonstrate that EQVC approach can significantly outperform other traditional methods regarding energy consumption, QoS guarantees, and the number of VM migrations.

INDEX TERMS Cloud computing, energy efficiency, quality of service, virtual machine migration.

I. INTRODUCTION

With cloud computing gaining significant momentum, a large number of data centers have been established around the world [1]. However, the issue of high energy consumption by data centers has become more visible and accounted for 1.5% of the global electricity in 2010 [2]. The significant power consumption depends on physical hosts, which run on the data center. Data collected for the review of green computing have shown that the energy consumption by the physical hosts has contributed approximately 60% to the overall cost of a data center [3]. However, based on the research [4], it has been estimated that the average utilization of host resources is between 15% and 20% of data centers. Therefore, it is essential to focus on improving the utilization of data center resources.

One method to enhance resource usage and save energy is dynamic VM consolidation, which has been widely employed in resource management in data centers. This approach periodically reallocates VMs to hosts by using live VM migration technology according to current resources requirements of the VMs. Additionally, the VMs can be dynamically consolidated to decrease the number of active hosts, while reducing the energy cost of the data center [5]. However, the difficulty of dynamic VM consolidation has increased due to the unpredictably fluctuating cloud workloads, and aggressive VM consolidation leading to service level agreement (SLA) violations and degradation of QoS because request resources of clients' applications are not fulfilled [6]–[9]. As a result, the goal of the minimization of energy consumption and the maximization of QoS is the leading challenge of dynamic VM consolidation.

On the other hand, the process of VM consolidation is always accompanied by large-scale VM migration. Voorsluys *et al.* [5] studied living VM migration, and they indicated that VM migration not only increases the cost of computing resources but also degrades the performance of the overall system and causes SLA violations. Additionally, massive VM live migrations increase power consumption and lead to QoS degradation, which opposes the target of VM consolidation. Recent studies [10]–[19] show that most VM consolidation approaches concentrate on improving energy efficiency and guaranteeing QoS but generate a large number of VM migrations. Additionally, a small number of VM migrations prevents the VM consolidation from improving resource utilization and energy efficiency. Therefore, the goal of our VM consolidation is to enhance energy efficiency and QoS with fewer VM migrations.

In this paper, we propose a VM consolidation (EQVC) method with low energy cost for cloud providers and providing guaranteed QoS for users, while reducing the number of VM migrations. The main contributions of the paper are as follows:

- We propose a host overload detection algorithm that identifies the overloaded host using the auto regressive integrated moving average (ARIMA) model to predict the future workload of the host. This algorithm detects the possibility of host overload to prevent potential SLA violations and host overload.
- We implement a host underload detection algorithm that identifies the underloaded host by analyzing the host's state and energy efficiency. All VMs on the host are migrated and the host is shut down to reduce power consumption.
- From the perspective of the CPU capacity loss, we find that there are many inefficient VM migrations, which generate additional energy consumption and reduce the performance of the VMs. We develop a VM selection algorithm based on the loss of CPU capacity in VM migration to improve the performance of the VMs and reduce invalid VM migrations.
- To prevent re-overloading physical hosts, we design an adaptive reserved resource for the hosts and apply it in our proposed VM placement algorithm. This reduces power consumption while improving QoS satisfaction.

The rest of this paper is as follows: we discuss the related work in Section 2. In Section 3, we introduce the system model, the power model, and the VM migration cost model. Section 4 presents the VM consolidation techniques. In Section 5, we introduce the simulation experiments and analyze the experimental result. Finally, we conclude and propose future research directions.

II. RELATED WORK

Nathuji and Schwan [6] first applied a dynamic VM consolidation method to minimize the energy cost of data centers. They developed an energy-saving approach that consolidated data center VMs using VM live migrations. Based on the foundation laid in their work [6], Beloglazov and Buyya [11] categorized the dynamic VM consolidation into four parts: host overload detection, host underload detection, VM selection, and VM placement.

Beloglazov *et al.* [10] studied the process of dynamic VM consolidation and proposed an upper static CPU utilization threshold to determine when a host has overloaded. However, the static threshold did not satisfy the rapidly changing cloud environments related to the dynamic and unpredictable cloud workloads. Beloglazov and Buyya [11] implemented an adaptive dynamic threshold to solve the problem of the variable cloud workloads. They developed the median absolute deviation (MAD), interquartile range (IQR), and local regression (LR) based on the statistical analysis of historical data to detect the overloaded host. Farahnakian *et al.* [12] investigated a CPU usage prediction method based on the linear regression technique to identify overloaded hosts. The approach migrated some VMs from the hosts which may be overloaded in the future, to reduce the possibility of host overloading. Gupta and Pateriya [13] proposed a host workload forecasting method using an AR model and applied it to the real cloud infrastructure. In contrast to the linear regression method [12], the proposed model improved the accuracy of predicting host overload and resulted in reducing the probability of host overload. Li *et al.* [14] developed an adaptive overload threshold based on the Bayesian network (BMEN) and detected the overloaded hosts based on the CPU utilization and the overload probability in current hosts. In summary, host overload detection based on the prediction method performs well and has been widely applied to VM consolidation. Therefore, in this paper, an ARIMA model is used for predicting the CPU utilization in host overload detection because it forecasts accurately.

Beloglazov *et al.* [10] also proposed a low static CPU utilization threshold to identify the underloaded host in which all VMs must be migrated to other hosts. Additionally, Beloglazov and Buyya [11] implemented a host underload detection algorithm for VM consolidation. This approach migrates all the VMs from an underloaded host that has the lowest CPU utilization to other active hosts and repeats this process until the VM cannot be assigned to the remaining hosts. However, the authors did not address the problem of different hosts having varied energy efficiency. Based on the research [11], Han *et al.* [15] investigated a power-aware algorithm to identify underloaded hosts to shut down for energy-saving. The algorithm sets a lower threshold of the CPU usage and proposes a PE value to evaluate the energy efficiency of the hosts. The selected host, which has the minimum PE and the lower threshold, migrates all VMs to other hosts. However, they also use the static threshold as a measure to determine the underloaded hosts. Thus, we develop a dynamic candidate list of underloaded hosts to select the underloaded hosts.

Beloglazov *et al.* [10] proposed two VM selection algorithms: minimization of migration (MM) and highest potential growth (HPG). The MM is applied to minimize the number of VM migrations, and the HPG selects the VM that has the lowest CPU utilization. Beloglazov and Buyya [11] implemented three VM selection algorithms for VM consolidation: maximum correlation (MC), minimum migration time (MMT), and random choice (RC). The MC chooses the VMs that have the higher correlation of CPU utilization with other VMs, the MMT selects the VMs with the least migration time, and the RC selects the VMs randomly. Li *et al.* [14] introduced a VM selection algorithm based on migration and capacity awareness. The VMs were selected by weighing the VM migration and host overload probability based on BNEM. Jiang *et al.* [16] investigated an ABC-based VM selection algorithm to select the VMs that led to the maximum decline in energy consumption. Khoshkholghi *et al.* [17] made a tradeoff between SLA and energy to select adequate VMs. Wen *et al.* [18] proposed a VM selection algorithm to minimize the number of VM migrations by calculating the Euclidean distance between the VMs' workloads and the hosts' workloads. However, they did not reduce undesired VM migrations. Thus, most of the above VM selection algorithms did not address the impact of VM migration on the QoS of cloud applications. Therefore, we investigate the relationship between VM migration and QoS and propose a VM selection algorithm to guarantee QoS and decrease the number of inefficient VM migrations.

Zhu *et al.* [7] modeled the process of VM placement as a bin-packing problem and proposed a modification of the FFD algorithm which establishes a new mapping between the VMs and hosts. Their work simplified the problem of VM placement but resulted in many VM migrations, which had a negative influence on energy consumption and QoS. Therefore, Beloglazov and Buyya [11] proposed a power-aware best fit decreasing (PABFD) for VM placement algorithm. The approach allocated each VM to a host that provides the least increase of power consumption to reduce energy consumption. However, they did not ensure users' QoS related to the rapidly changing cloud workloads. Han *et al.* [15] investigated a remaining utilization-aware (RUA) algorithm for VM placement and applied a heuristic of setting static remains CPU utilization of 0.17 to prevent host overload. However, this approach led to the waste of computing resources when the cloud workloads were stable. Our approach uses adaptive reserve CPU utilization to make a tradeoff between preventing SLA violations and improving the resource utilization. Li *et al.* [19] developed an energy-efficient and QoS-aware model and proposed a VM placement algorithm based on the particle swarm optimization method, which sets the power consumption per QoS value as the fitness objective. However, the approach caused problems with being trapped in a locally optimal solution. Gupta and Pateriya [13] proposed a VM placement algorithm that considered the VMs' resources requirement and hosts' resources utilization. Li *et al.* [14] proposed a migration and power-aware best fit decreasing for VM placement based on BNEM. Khoshkholghi *et al.* [17] proposed the best RAM and bandwidth placement algorithm for VM placement to find the hosts where the selected VMs use minimum energy and are least likely to commit SLA violations. Monil and Malony [20] evaluated the challenge of achieving a balance between QoS and energy consumption and devised a combined strategy to achieve the placement of the VM using the best fit decreasing bin packing method. Qiu *et al.* [21] provided a method that considers load balancing, power consumption, and migration costs. They regarded VM placement as a multi-objective optimization problem and then solved the problem using the modified genetic algorithm. Fard *et al.* [22] proposed a VM placement to allocate VMs to hosts within high-level category hosts to preserve high category host activity and low category hosts inactivity. In summary, the above VM placement algorithms apply only to minimizing energy consumption, and ignored host re-overload. However, our VM placement algorithm reduces energy consumption and prevents host re-overload.

Compared with the above approaches, our method minimizes energy cost, prevents host re-overload and decreases the number of undesired VM migrations without compromising QoS conditions.

III. EVALUATION MODELS AND METRICS

In this paper, the target cloud system model is the Infrastructure as a Service (IaaS) environment that consists of heterogeneous physical hosts. The performance of each host can be described by the CPU capacity that is defined in millions of instructions per second (MIPS), amount of RAM, and network bandwidth. The system storage is network attached storage (NAS), which enables live VM migration. The cloud data center detects the performance of physical hosts at regular intervals.

A. SYSTEM MODEL

In a data center, the types of resources are expressed as follows: $H = \{h_1, h_2, \ldots, h_i, \ldots, h_N\}$ is the set of N cloud data center hosts, and $VM_i = \{v_1, v_2, \ldots, v_j, \ldots, v_m\}$ is the set that m VMs deployed in h_i.

The characteristics of v_j are described as follows: v_j^{mc} is the max requested CPU capacity of v_j, v_j^{rc} represents the current requested CPU capacity of v_j, and the current CPU utilization of v_j is defined as v_j^{u}. From the definitions of v_j^{mc}, v_j^{rc} and v_j^{cpu}, the relationships can be defined as follows:

$$v_j^{rc} = v_j^{mc} \times v_j^{u} \tag{1}$$

h_i^{rc} represents the requested CPU capacity of h_i and is calculated as follows:

$$h_i^{rc} = \sum_{v_j \in VM_i} v_j^{rc} \tag{2}$$

The max CPU capacity of h_i is denoted as h_i^{mc} and h_i^{u} is the current CPU utilization of h_i. Equation (3) expresses the relationships between h_i^{rc}, h_i^{mc} and h_i^{u}.

$$h_i^{u} = \begin{cases} \dfrac{h_i^{rc}}{h_i^{mc}} & if\ h_i^{rc} < h_i^{mc} \\ 1 & if\ h_i^{rc} \geq h_i^{mc} \end{cases} \tag{3}$$

B. POWER MODEL

The power consumption of the hosts is the determined by the CPU, RAM, disk storage, and network interfaces [23]. Fan et al. [24] explained that the CPU is the primary cause of the host's energy consumption, and the power of the host is linearly related to its CPU utilization. According to the research [9], the power model by CPU utilization is defined in (4):

$$P(h) = K \times h^{\max} + (1 - K) \times h^{\max} \times h^{u} \tag{4}$$

where $h^{\max}$ represents the host's max power when the host's CPU utilization is 100%, and K is the percentage of power consumption for an idle host and set to 0.7 based on the research [25].

The CPU utilization may change over time, which means the host's CPU utilization is a function of time. Therefore, the total energy consumption of a host is defined as follows:

$$EC = \int_{t_0}^{t_1} P(h^{u}(t))dt \tag{5}$$

C. LIVE MIGRATION COST

VM can transfer between hosts without suspension by using VM live migration techniques [6], [26]. The average performance degradation is equivalent to 10% of the CPU utilization of the VM during the migration [5]. Thus, we define the model of live migration cost based on the research [11] as follows:

$$T_j^m = \frac{v_j^{ram}}{h_i^{bw}} \tag{6}$$

$$v_j^{du} = 0.1 \cdot \int_{t_0}^{t_0+T_j^m} v_j^u(t)dt \tag{7}$$

where T_j^m represents the migration time of v_j, v_j^{ram} represents the amount of RAM of v_j, h_i^{bw} represents the available network bandwidth of h_i, v_j^{du} represents the total decreased CPU utilization during the VM migration, and $v_j^u(t)$ represents the CPU utilization of the VM at time t.

According to this model, VM migration leads to performance degradation, which causes SLA violations and decreased QoS. Thus, it is essential to improve the QoS by reducing the VM migrations.

D. QOS EVALUATION

QoS is presented in the form of an SLA, which is decided in terms of characteristics such as the CPU capacity of the host or maximum response time [27]. In the cloud environment, users submit the request for creating VMs to the data center and sign the SLA with the data center. According to the research [11], the SLA is defined as the CPU capacity that the VM request must be fulfilled by the physical host and therefore proposed the SLATAH and PDM metrics to measure SLA violations in an IaaS environment.

SLATAH represents the percentage of time that the active host experienced CPU utilization of 100% and is defined in (8).

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{8}$$

where N is the total number of hosts, T_{s_i} is the time of the SLA violations when the h_i experienced utilization of 100%, and T_{a_i} is the active time of h_i. PDM represents the overall performance degradation by VM migrations and is defined in (9).

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{9}$$

where M is the total number of VMs, C_{d_j} is the total loss of CPU capacity by migrating v_j, and C_{r_j} is the total requested CPU capacity of v_j. Based on SLATAH and PDM, a combined metric SLAV measures the QoS of the data center and is calculated as (10). Decreasing the SLATAH, PDM, and SLAV can result in an improved QoS.

$$SLAV = SLATAH \times PDM \tag{10}$$

IV. EQVC METHOD

In this section, we propose the EQVC method that reduces power consumption and improves QoS with a small number of VM migrations. According to the research [11], the problem of dynamic VM consolidation can be divided into four parts: (1) determining when a host is considered overloaded and then migrating one or more VMs from the host; (2) determining when a host is considered underloaded, all VMs from the host need to be migrated and the host is shut down; (3) selecting the VM that should be migrated from the overloaded host; and (4) finding new hosts for the VMs from overloaded and underloaded hosts. Corresponding to the above four stages of VM consolidation, the EQVC approach is comprised of the following four algorithms:

(1) Host overload detection algorithm: By predicting the future workload of the host, the excess VMs are migrated before the host is overloaded. This algorithm prevents host overloading and improves QoS.

(2) Host underload detection algorithm: According to the use of the host, create a list of candidate underloaded hosts, and select the underloaded host from the list by comparing the energy efficiency of the host. All VMs from the selected host are migrated and the host is shut down to improve resource utilization and save energy in the data center.

(3) VM selection algorithm: The proposed VM selection algorithm determines whether to migrate VMs from the overloaded host by comparing the loss of CPU capacity of the VM migration and the host overload. When a VM must be migrated, select the VM with the lowest CPU capacity loss. This algorithm reduces unnecessary VM migration while reducing the loss of CPU capacity and improving QoS satisfaction.

(4) VM placement algorithm: According to the historical status of the host, create a list of hosts that is used to receive the VMs. Select the hosts for VM placement by analyzing the historical CPU utilization and the currently available CPU utilization of the hosts in the list. This algorithm reduces energy consumption while reducing the chance of the host overloading again.

Fig. 1 shows the process of the EQVC approach.

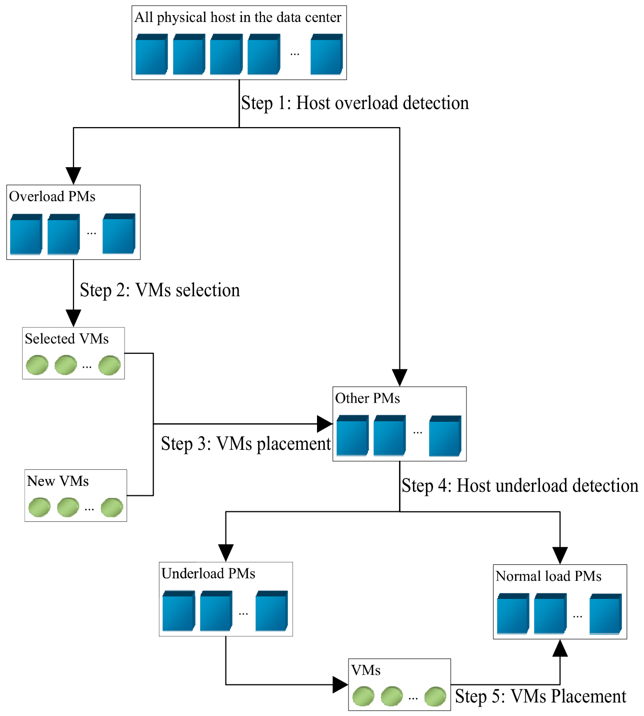


FIGURE 1. The process of EQVC approach.

A. HOST OVERLOAD DETECTION ALGORITHM

One of the first steps in the VM consolidation is to recognize the overloaded host. Our host overload algorithm aims to reduce the possibility of host overload while maintaining the normal workload of the physical hosts to protect QoS. To achieve this goal, the host detection algorithm detects the overloaded host by predicting the host's workload, to migrate its VMs to other hosts before it is overloaded. Calheiros *et al.* [28] introduced the prediction by using the ARIMA model and evaluating its accuracy of resource utilization and QoS. Their work showed that the ARIMA model had an average forecast accuracy of 91% when applied to forecast workloads. According to the research [29], [30], ARIMA accurately forecasts future workloads in large-scale heterogeneous physical hosts. Thus, we use the ARIMA model to implement the host overload detection algorithm.

In this algorithm, the ARIMA (*p, q, d*) model is fitted according to the Box-Jenkins method [31]. The history CPU utilization of the physical host is measured at regular intervals, which is consistent with the ARIMA model. The original time series must be transformed into a stationary time series. Thus, the time series must be differenced until it becomes stationary, and the number of differences is the d parameter of the ARIMA model.

According to the historical workload data, the parameters of q and p are determined by analyzing the partial autocorrelation plot and autocorrelation plot. For a host's history CPU utilization set $x_1, x_2, \ldots, x_n$, the autocorrelation function is estimated by the sample moment, which is called the autocorrelation plot.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{11}$$

$$\gamma_0 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{12}$$

$$\gamma_k = \frac{1}{n-k}\sum_{i=1}^{n-k} (x_i - \bar{x})(x_{i+k} - \bar{x}) \tag{13}$$

$$R_k = \frac{\gamma_k}{\gamma_0} \tag{14}$$

where $\bar{x}$ is the average of the data set, γ_0 is the estimation of the variance of the data set, γ_k is the biased estimation of covariance of x_i, x_{i+k}, and R_k is the estimation of autocorrelation coefficients of x_i, x_{i+k}. In the autocorrelation plot, R_k is the values on the vertical axis, and the time lags are on the horizontal axis. The autocorrelation between x_i and $x_{i-\tau}$ is the partial autocorrelation at τ, which accounts for lags above $\tau - 1$. If the partial autocorrelation plot falls below the significant level at $\tau = p + 1$, the auto regression (AR) component of the ARIMA model has order p. The value of q for the moving average (MA) component of the ARIMA is the number of lags before the autocorrelation values drop below the significant level.

The parameters *p, q,* and *d* are obtained using the above method. The host history CPU utilization data set is used to fit the model to predict the future CPU utilization of the host. For the physical host h_i, the historical CPU utilization data set is $h_i^u(1), h_i^u(2), \ldots, h_i^u(n)$, and the $\widehat{h_i^u}$ is defined as the estimated CPU utilization at the next time using the ARIMA model. Thus, if a host satisfies the (15), the host is overloaded.

$$s \times \widehat{h_i^u} \geq 1 \tag{15}$$

where s represents a safety parameter that allows the host to reserve CPU capacity to prevent SLA violations. Algorithm 1 shows the host overload detection. If n is the number of active physical hosts, the time complexity of the host overload detection algorithm is O (n).

B. VM SELECTION

When finding a set of overloaded hosts, some VMs in the hosts are migrated to guarantee QoS for the users. In this paper, we analyze the loss of CPU capacity in an overloaded host. If a VM is migrated to other hosts, the performance of the VMs degrade during the VM migration and cause the loss of CPU capacity. If the VMs are not migrated, the host

Algorithm 1 Host Overload Detection

```
Input: hostList
Output: overloadedHostList
1  overloadedHostList ← $NULL$;
2  foreach host in hostList do
3      if host satisfies equation (15) then
4          overloadedHostList.add(host);
5      end if
6  end for
7  return overloadedHostList;
```

remains overloaded, which leads to the host being unable to meet the CPU capacity of the VM requested and generates a loss of CPU capacity. In other words, whether the VMs in the overloaded host are migrated or not migrated, the loss of CPU capacity cannot be avoided. According to the above analysis, when selecting the VM to migrate, the CPU capacity loss caused by migrating the VM needs to be compared with the host overload. If the CPU capacity loss of the migrating of the VM is greater than the loss caused by host overload, the VM migration is invalid. Thus, the goal of our VM selection algorithm is to reduce the invalid VM migrations and the loss of CPU capacity, which can guarantee the QoS.

We assume h_i is an overloaded host that fails to satisfy the requested CPU capacity of a VM, which is employed by h_i. h_i^{lc} represents the lack of CPU capacity of h_i and can be calculated as follows:

$$h_i^{lc} = (s \times \widehat{h}_i^u - 1) \times h_i^{mc} \tag{16}$$

Based on the above analysis, v_j ($v_j \in VM_i$) has two cases when h_i is overloaded:

Case 1: when migrating v_j to another host, d_j represents the loss of CPU capacity of v_j during the migration and is calculated as follows:

$$d_j = v_j^{du} \times v_j^{mc} \tag{17}$$

Case 2: when v_j is not migrated, $D_{i,j}$ represents the loss of CPU capacity during the host overload and can be defined as follows:

$$D_{i,j} = \int_{t_0}^{t_0+T_j^m} h_i^{lc}(t)dt \tag{18}$$

By comparing the above two cases, the relative CPU capacity gains can be defined as follows:

$$\phi_{i,j} = D_{i,j} - d_j \tag{19}$$

where $\phi_{i,j}$ represents the loss of CPU capacity that is produced by not migrating v_j from h_i. $\phi_{i,j} > 0$ indicates that the loss of CPU capacity can be reduced by migrating v_j from h_i, and v_j is regarded as a candidate VM in our VM selection algorithm, $\phi_{i,j} < 0$ reflects that migrating v_j can increase the CPU capacity loss and the migration of v_j is undesired.

The pseudocode for the VM selection in the overloaded host is presented in Algorithm 2. The algorithm checks the list of overloaded hosts' VMs and calculates the relative CPU capacity gains for all the VMs. The VM list is sorted in decreasing order of the relative CPU capacity gains of the VMs. When selecting the migrated VM, it is selected in the sorted order and confirmed that the relative CPU capacity gains of the selected VM are greater than zero. Repeat this process until the host is not overloaded or the selected VM's relative CPU capacity gains are less than zero. If n is the number of overloaded physical hosts and m is the number of the VMs, the time complexity of the host overload detection algorithm is O (n*m).

Algorithm 2 VM Selection

```
Input: overloadedHostList
Output: selectedVmList
1  selectedVmList ← $NULL$;
2  foreach host in overloadedHostList do
3      vmList ← host.getVmList();
4      foreach vm in vmList do
5          Calculate $\phi$ for vm by using equation (19);
6          vm.updataGains($\phi$); // update $\phi$ for vm
7      end for
8      Sort vmList in decreasing order of $\phi$
9      foreach vm in vmList do
10         if vm.getGains() $>$ 0 then
11             selectedVmList.add(vm); // migrate vm
12             host.removeVm(vm);
13             if host satisfies equation (15) then
14                 continue; // host still overloaded,
                   continue remove vm
15             else
16                 break;
17             end if
18         else
19             break; // do not migrate vm
20         end if
21     end for
22 end for
23 return seletedVmList;
```

C. VM PLACEMENT

The VM placement algorithm allocates the new VMs and the selected VMs by the VM selection algorithm to idle hosts, which improves the resource utilization of the physical host and reduces the energy consumption of the data center. However, the workload of the VMs is unpredictable and fluctuating, which causes the physical host to re-overload after the VMs allocation is completed, causing the SLAV and degrading the QoS. Our VM placement algorithm provides an adaptive CPU reserve utilization for physical hosts to prevent sudden changes in the workload as well as maintain the normal load status of the hosts. Additionally, the algorithm creates the possibility of a degraded host overload, which means the number of overloaded hosts and VM migration

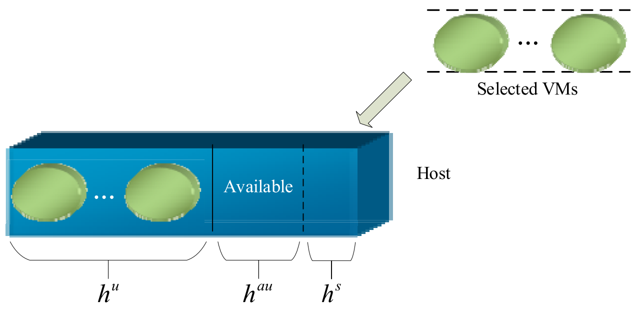


FIGURE 2. The process of VMs placement and the host model.

drops to reduce SLAV. Thus, the goal of our algorithm is to improve QoS by preventing host re-overloading while reducing energy consumption. Fig. 2 shows the process of our VM placement mechanism and the host model.

As shown in Fig. 2, h^u is the CPU utilization of the host, h^{au} is the available CPU utilization of the host, and h^s is the host's adaptive reserved CPU utilization that prevents sudden changes in the cloud workload. The MAD is a measure of statistical dispersion and is more robust than the standard deviation or sample variance [32]. Thus, For the CPU utilization history data set $h_i^u(1), h_i^u(2), \ldots, h_i^u(n)$ of h_i, the $MAD(h_i)$ is the median absolute deviation of the median of the data and is defined as follows:

$$MAD(h_i) = median(|h_i^u(k) - median(h_i^u(1), \ldots, h_i^u(n))|) \tag{20}$$

where $h_i^u(n)$ represents the nth CPU utilization in the history data. Based on the $MAD(h_i)$, we define the adaptive reserved CPU utilization as shown in (21):

$$h_i^s = \mu \cdot MAD(h_i) \tag{21}$$

where h_i^s represents the CPU usage that prevents host re-overload, and μ adjusts the reserved resources and was set to 2.5 based on research [10]. According to the above formula, h_i^{au} is calculated as follows:

$$h_i^{au} = 1 - h_i^s - h_i^u \tag{22}$$

$\Delta h_{i,j}^u$ represents the increased CPU utilization of h_i when v_j allocates to h_i and is calculated as follows:

$$\Delta h_{i,j}^u = \frac{v_j^u \cdot v_j^{mc}}{h_i^{mc}} \tag{23}$$

$\varphi_{i,j}$ represents the remaining available CPU utilization of h_i when v_j has been migrated to h_i:

$$\varphi_{i,j} = h_i^{au} - \Delta h_{i,j}^u \tag{24}$$

Our VM placement algorithm can be divided into three parts: first, this algorithm sorts the selected VMs list in decreasing order of CPU capacity. Second, this algorithm looks for the candidate hosts set that can receive the VMs in the data center. The candidate host satisfies two conditions: (1) it has not been overloaded, and (2) its available CPU utilization is greater than zero ($h^{au} > 0$). Finally, this algorithm allocates the selected VMs to the best hosts that are in the candidate list. The best host satisfies two conditions: (1) the remainder available CPU utilization of the host is higher than zero, and (2) the host has the lowest remainder available CPU utilization relative to other hosts. The pseudocode for the VM placement algorithm is presented in Algorithm 3. If n is the number of active physical hosts and m is the number of the selected VMs, the time complexity of the algorithm is O (n*m).

Algorithm 3 VM Placement

```
Input: hostList, overloadedHostList, selectedVmList
Output: allocatedHostList
1  Sort selectedVMList in decreasing order of CPU
   capacity
2  otherHostList ← hostList.remove(overloaded
   HostList); // candidate host has not been overloaded
3  candidateHostList ← NULL;
4  foreach host in otherHostList do
5      Calculate $h^{au}$ using equation (22);
6      if $h^{au} > 0$ then // candidate host's available CPU
       utilization is greater than zero
7          candidateHostList.add(host);
8      end if
9  end for
10 foreach vm in selectedVmList do
11     $\varphi_{\min} \leftarrow$ MAX;
12     allocatedHost ← NULL;
13     foreach host in candidateHostList do
14         Calculate $\varphi$ using equation (24);
15         if $\varphi > 0 \&\& \varphi < \varphi_{\min}$ then // the best host's
           remainder available CPU utilization is higher than
           zero and the lowest
16             $\varphi_{\min} \leftarrow \varphi$;
17             allocatedHost ← host;
18         end if
19     end for
20     if allocatedHost $\neq$ NULL then
21         allocatedHost.getVmList().add(vm);
22         selectedVmList.remove(vm);
23     end if
24     allocatedHostList.add(allocatedHost);
25 end for
26 return allocatedHostList;
```

D. UNDERLOADED HOST DETECTION ALGORITHM

The key step of VM consolidation is host underload detection, which is employed to select the underloaded host. All VMs of the selected host must be migrated to other hosts, and the underloaded host shut down to improve resource utilization and save energy. Our host underload algorithm sets the list of candidate hosts for underloaded hosts based on the host operating state, and then selects the underloaded host from the list according to the energy efficiency of the host. The goal of the proposed VM placement algorithm is to allocate all VMs of the underloaded host to other hosts, which can reduce power consumption without host overload. Therefore, a metric that compares the energy efficiency of the active hosts is defined as:

$$h^{ea} = \frac{P(h)}{h^u} \tag{25}$$

where h^{ea} is the ratio of the host's power consumption to the CPU capacity of the host. The higher the value of h^{ea} is the lower the energy efficiency of the host.

Algorithm 4 shows the underloaded host detection. First, this algorithm seeks the underloaded candidate hosts that must be meet two conditions: (1) the host has not migrated VM, and (2) the host has not been overloaded. Then, this algorithm checks the candidate hosts and selects a host that has the max h^{ea} as the underloaded host. If n is the number of active physical hosts, the time complexity of the algorithm is O(n).

Algorithm 4 Host Underload Detection

```
Input: hostList, overloadedHostList, allocatedHostList
Output: underloadedHost
1   $h^{ea}_{\max} \leftarrow MIN$;
2   otherHostList←hostList.remove(overloadedHostList);
    // candidate host has not been overloaded
3   otherHostList.remove(allocatedHostList); // candidate
    hosts has not migrated VM
4   candidateHostList ← $NULL$;
5   foreach host in candidateHostList do
6       Calculate $h^{ea}$ using equation (25);
7       if $h^{ea} > h^{ea}_{\max}$ then // select the underloaded host
8           $h^{ea}_{\max} \leftarrow h^{ea}$;
9           underloadedHost ← host;
10      end if
11  end for
12  return underloadedHost;
```

E. THE PROCESS OF VM CONSOLIDATION

As shown in Fig. 1, the EQVC method periodically adjusts the placement of VMs. The approach can be split into two phases. The first phase identifies overloaded physical hosts and migrates the excess VMs from the hosts. First, the method checks the list of active hosts in the data center and identifies the overloaded host by using the host overload detection algorithm. Second, the method selects VMs that need to be migrated from the overloaded hosts using the VM selection algorithm. Third, the method identifies new and suitable placement for the VM migrating from the overloaded host using the VM placement algorithm.

The second phase of the method identifies underloaded physical hosts and migrates all VMs from the hosts. The EQVC method repeatedly inspects the hosts list and looks for the underloaded host using the host underload detection algorithm. When the underloaded host is found, all VMs on the host are allocated using the VM placement algorithm. The above steps until the VMs cannot be assigned, and then the host cancels the VM migration and remains active. In the data center, the EQVC method periodically consolidates the VMs to maintain high energy efficiency and excellent QoS with less VM migrations. The process of the EQVC method is described in Algorithm 5.

Algorithm 5 Enhancing Energy-Efficient and QoS VM Consolidation

```
Input: hostList
1   overloadedHostList ← hostOverloadedDetection
        (hostList);
2   selectedVmList ← VMSelection
        (overloadedHostList);
3   allocatedHostList ← VMPlacement (hostList,
        overloadedHostList, seletedVmList);
4   otherHostList ← hostList.remove
        (overloadedHostList);
5   otherHostList.remove(allocatedHostList);
6   while otherHostList $\neq NULL$ do
7       underloadedHost ← hostUnderloadDetection
        (hostList, overloadedHostList, allocatedHostList);
8       vmList ← underloadedHost.getVmList();
9       VMPlacement(hostList, overloadedHostList,
            vmList);
10      if vmList $\neq NULL$ then
11          return;
12      else
13          otherHostList.remove(underloadedHost);
14      end if
15  end while
```

V. EXPERIMENT AND EVALUATION

In this section, we designed a series of experiments to verify the effectiveness of the proposed EQVC method in the IaaS environment.

A. EXPERIMENTAL SETTING

This paper used the CloudSim toolkit [33] as a simulation platform. CloudSim toolkit is a simulation framework for cloud computing environments. Compared to other simulation toolkits, it supports modeling of on-demand virtualized resources and application management. It also simulates energy-awareness and service applications with workloads over time. In the experiment, we simulated a heterogeneous data center that consisted of 400 HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores * 1.86 GHz) and 400 HP ProLiant ML110 G5 (Intel Xeon 3075, 2 cores * 2.26 GHz) physical hosts. Referencing the benchmark data provided by SPECpower [11], the maximum energy consumption of the two types of hosts was 117 W and 135 W. Additionally, four instances of VMs were used in the experiments:

High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB).

TABLE 1. PlanetLab trace (CPU utilization).

Date	Number of VMs	Mean(%)	St.dev.(%)
2011/03/03	1052	12.31%	17.09%
2011/03/06	898	11.44%	16.83%
2011/03/09	1061	10.70%	15.57%
2011/03/22	1516	9.26%	12.78%
2011/03/25	1078	10.56%	14.14%
2011/04/03	1463	12.39%	16.55%
2011/04/09	1358	11.12%	15.09%
2011/04/11	1233	11.56%	15.07%
2011/04/12	1054	11.54%	15.15%
2011/04/20	1033	10.43%	15.21%

To prove the feasibility of the proposed EQVC method, we conducted experiments using two workload traces from the real system: PlanetLab trace [34] and Bitbrains trace [35]. The PlanetLab trace included the CPU utilization measured every 5 minutes from more than one thousands of VMs for 10 days. Table 1 shows the PlanetLab trace information. The Bitbrains trace contained 1,750 VMs' performance metrics, which were measured from the Bitbrains distributed data center every 5 minutes for 3 months. In the experiments, we selected the CPU utilization by VMs data for 10 days. Table 2 shows the information of the Bitbrains trace. During the simulation, VMs that were consistent with the number of VMs from the workload traces were created, and these VMs were randomly assigned the workload traces from the VMs in the corresponding day.

B. PERFORMANCE METRICS

To evaluate the performance of the VM consolidation method, the six performance metrics from the research [10] were adopted: energy consumption (EC), SLA time per active host (SLATAH), performance degradation due to migrations (PDM), SLA violations (SLAV), energy and SLAV (ESV), and VM migrations (VMM). The SLATAH, PDM and SLAV were mentioned in Section III.D.

EC represents the energy consumption of all physical hosts in the data center. ESV represents a combined metric including both EC and SLAV and was defined in (26). The lower ESV indicates that the data center has higher energy efficiency and QoS.

$$ESV = EC \times SLAV \tag{26}$$

VMM represents the total number of VM migrations in the data center, and reducing VMM can improve QoS.

C. RESULT OF EXPERIMENTS

In the same experimental conditions, we compared the proposed EQVC method with the RUA method [15] and DTHMF method [20]. The parameter s of EQVC was set to 1.2. Additionally, we considered the method [11] of combining

TABLE 2. Bitbrains trace (CPU utilization).

Date	Number of VMs	Mean(%)	St.dev.(%)
2013/08/01	1171	11.85%	6.26%
2013/08/02	1169	8.06%	6.43%
2013/08/03	1171	5.40%	4.00%
2013/08/04	1156	9.06%	5.23%
2013/08/05	1151	10.12%	6.95%
2013/08/06	1141	9.33%	4.18%
2013/08/07	1143	8.25%	6.88%
2013/08/08	1131	11.52%	7.24%
2013/08/09	1130	7.55%	6.42%
2013/08/10	1123	9.28%	5.94%

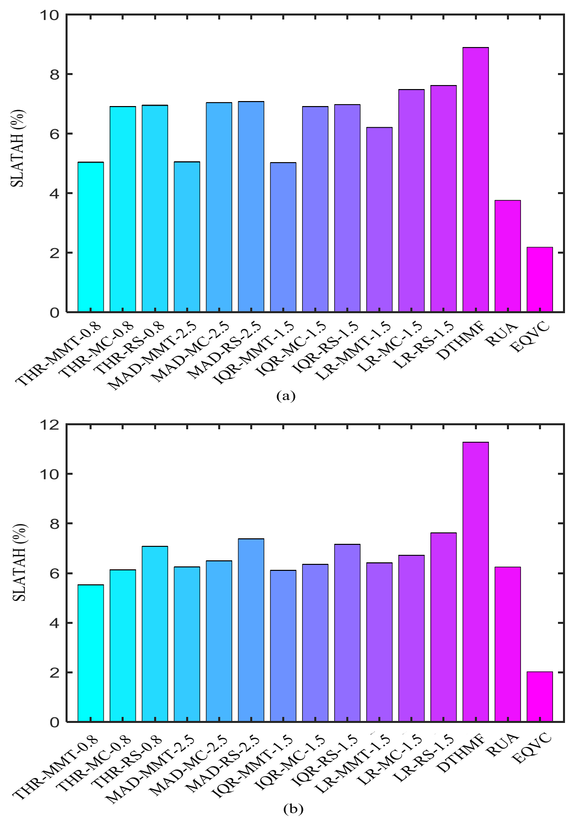


FIGURE 3. SLATAH by the different algorithms under the PlanetLab and Bitbrains trace.

four host overload detection algorithms (THR, MAD, IQR and LR) with three VM selection algorithms (MMT, MC, and RS) as the benchmark method. Tables 3 and 4 show the experimental results with different traces, such as EC, SLAV, ESV, and VMM. The values in the table are the mean of the corresponding performance indicators and the 95% confidence interval (CI) for the mean. Fig. 3 and Fig. 4 show the experimental results of SLATAH and PDM, which is a further analysis of SLAV. In the above figures and tables, the numbers following the name of each method are the parameters of the corresponding method.

The EC metrics can be used to compare the performance of different methods regarding power consumption. In Tables 3 and 4, we observe that EQVC outperforms the

TABLE 3. Results using PlanetLab trace with four metrics.

Method	EC(kWh)	SLAV(x0.0001)	ESV(x0.01)	VMM
EQVC	111.52 (96.38, 126.67)	0.0638 (0.0304, 0.0972)	0.0670 (0.0408, 0.0933)	6552 (5607, 7497)
RUA	123.39 (106.86, 139.92)	0.1201 (0.0819, 0.1583)	0.1438 (0.1087, 0.1789)	12690 (10910, 14470)
DTHMF	146.24 (125.24, 167.25)	0.3648 (0.2854, 0.4441)	0.5257 (0.4262, 0.6252)	13609 (11557, 15662)
THR-MMT-0.8	188.45 (164.61, 212.29)	0.3371 (0.3078, 0.3664)	0.6256 (0.5807, 0.6705)	26601 (23570, 29633)
THR-MC-0.8	179.33 (156.48, 202.17)	0.6989 (0.6587, 0.7391)	1.2410 (1.1321, 1.3498)	23961 (21189, 26734)
THR-RS-0.8	180.80 (157.89, 203.71)	0.6991 (0.6526, 0.7456)	1.2507 (1.1391, 1.3622)	24217 (21529, 26905)
MAD-MMT-2.5	183.43 (161.08, 205.78)	0.3348 (0.3083, 0.3613)	0.6061 (0.5598, 0.6525)	26305 (23341, 29268)
MAD-MC-2.5	173.74 (151.78, 192.70)	0.7111 (0.6688, 0.7534)	1.2252 (1.1026, 1.3478)	23420 (20684, 26154)
MAD-RS-2.5	174.96 (153.15, 196.77)	0.7111 (0.6692, 0.7530)	1.2338 (1.1135, 1.3541)	23736 (21066, 26405)
IQR-MMT-1.5	187.50 (164.50, 210.49)	0.3288 (0.2971, 0.3605)	0.6071 (0.5604, 0.6537)	26497 (23579, 29414)
IQR-MC-1.5	177.66 (155.33, 199.99)	0.6805 (0.6439, 0.7171)	1.1976 (1.0925, 1.3026)	23394 (20695, 26092)
IQR-RS-1.5	179.06 (156.85, 201.27)	0.6793 (0.6388, 0.7198)	1.2061 (1.0899, 1.3224)	23796 (21132, 26458)
LR-MMT-1.2	161.81 (141.56, 182.06)	0.4974 (0.4401, 0.5547)	0.7951 (0.7091, 0.8811)	28175 (24591, 31757)
LR-MC-1.2	148.47 (129.36, 167.58)	0.7609 (0.6806, 0.8412)	1.1185 (0.9873, 1.2496)	23931 (20651, 27210)
LR-RS-1.2	147.61 (128.98, 166.24)	0.7781 (0.6987, 0.8575)	1.1381 (1.0084, 1.2679)	23779 (20626, 26930)

TABLE 4. Results using Bitbrains trace with four metrics.

Method	EC(kWh)	SLAV(x0.0001)	ESV(x0.01)	VMM
EQVC	118.67 (99.78, 137.56)	0.0709 (0.0405, 0.1013)	0.0791 (0.0460, 0.1121)	9050 (7450,10649)
RUA	129.39 (110.53, 148.24)	0.2630 (0.1888, 0.3372)	0.3385 (0.2243, 0.4528)	14748 (12711, 16784)
DTHMF	161.81 (140.73, 182.88)	0.6603 (0.4909, 0.8297)	1.0902 (0.7533, 1.4270)	17816 (15202, 20428)
THR-MMT-0.8	191.87 (167.58, 216.16)	0.3594 (0.3129, 0.4059)	0.7059 (0.5442, 0.8675)	29612 (26738, 32484)
THR-MC-0.8	179.98 (159.66, 200.30)	0.5194 (0.4431, 0.5957)	0.9598 (0.7443, 1.1752)	25804 (24211, 27395)
THR-RS-0.8	179.96 (159.49, 200.43)	0.6095 (0.5411, 0.6779)	1.1186 (0.8963, 1.3408)	25889 (24272, 27505)
MAD-MMT-2.5	184.97 (161.55, 208.40)	0.6001 (0.5022, 0.6980)	1.1432 (0.8628, 1.4237)	37843 (32942, 42743)
MAD-MC-2.5	171.24 (153.04, 189.44)	0.6519 (0.5511, 0.7527)	1.1430 (0.8939, 1.3920)	31415 (27934, 34894)
MAD-RS-2.5	171.04 (152.21, 189.87)	0.7529 (0.6479, 0.8579)	1.3151 (1.0331, 1.5971)	30294 (27689, 32899)
IQR-MMT-1.5	193.18 (168.92, 217.44)	0.5493 (0.4670, 0.6316)	1.0853 (0.8382, 1.3323)	36297 (32113, 40480)
IQR-MC-1.5	180.21 (159.82, 200.60)	0.6098 (0.5250, 0.6946)	1.1245 (0.8882, 1.3607)	30637 (28021, 33252)
IQR-RS-1.5	180.18 (159.00, 201.36)	0.6851 (0.5995, 0.7707)	1.2590 (0.9967, 1.5213)	29669 (27142, 32194)
LR-MMT-1.2	175.61 (152.86, 198.36)	0.4883 (0.4150, 0.5616)	0.8753 (0.6634, 1.0872)	31434 (28494, 34372)
LR-MC-1.2	166.54 (146.85, 186.23)	0.5937 (0.4957, 0.6917)	1.0113 (0.7627, 1.2600)	27378 (25143, 29613)
LR-RS-1.2	165.12 (146.24, 183.99)	0.6800 (0.5910, 0.7690)	1.1350 (0.8977, 1.3722)	27014 (25260, 28768)

other approaches in energy-efficiency under the PlanetLab and Bitbrains traces. The EQVC method saved approximately 34.6% of the energy compared to the benchmark algorithm. The THR-MMT consumed the maximum energy in the PlanetLab trace, and the IQR-MMT has the maximum energy consumption in the Bitbrains trace. Further analysis finds that the power cost of the VM consolidation method that contained the MMT algorithm is consistently higher than the others. This is because the MMT algorithm selects many undesired VM migrations, resulting in great energy consumption when allocating the selected VMs. However, we see that EQVC can avoid inefficient VM migrations because the result of EQVC has the lowest EC and the least VM migrations. Consequently, the performance of the EQVC is better than the other methods.

The SLAV metrics can be applied to evaluate SLA violations while the lower SLAV represents the higher QoS. As shown in Tables 3 and 4, EQVC outperforms other methods regarding SLAV, which indicates that EQVC can efficiently reduce SLA violations and improve QoS for users. Fig. 3 and Fig. 4 show the further analysis of SLAV, which consists of both SLATAH and PDM. The SLATAH metrics reflect the possibility of overloading the physical host. Fig. 3 compares the results of SLATAH with other methods. We see that for both PlanetLab and Bitbrains traces, EQVC is much lower than the other methods. Compared with the benchmark algorithm, the probability of host overloading of the EQVC approach reduces approximately 67.97%. The SLATAH value of DTHMF is the highest in both traces, which indicates that the approach is more likely to overload. This is because when detecting the overloaded host, DTHMF relies on the temperature of the physical host according to the change in cloud workloads, reducing the accuracy of overload detection. However, EQVC can proactively migrate excess VMs from a host before the host became overloaded, reducing the overloading probability of the host. Additionally, to prevent host re-overload, EQVC also provides adaptive reserved resources for each host. Therefore, EQVC can efficiently reduce the occurrence of host overload while guaranteeing QoS requirements.

The PDM metrics reflect the loss extent of CPU capacity due to VM migration. The PDM of EQVC is lower than the benchmark method by 67.60%, as shown in Fig. 4. RUA and DTHMF perform adequately in the PDM metrics but worse than EQVC. The PDM value of the methods that contain the RS algorithm is consistently the highest because the RS randomly selects the VM for migration, and the purposeless VM migration causes the increased loss of

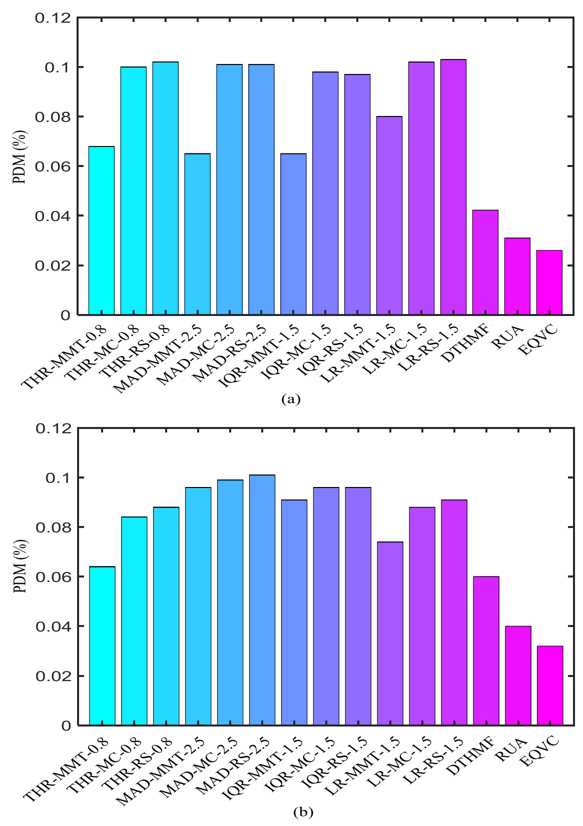


FIGURE 4. PDM by the different algorithms under the PlanetLab and Bitbrains trace.

computing resources. However, EQVC minimizes the CPU capacity loss caused by the VM migration. Thus, EQVC can improve the degraded performance of CPU capacity during VM migration, while guaranteeing QoS.

In Tables 3 and 4, the behaviors of the EC and the VMM are closely related. The VM migration not only leads to degraded VM performance but also causes energy consumption. EQVC has the lowest number of VM migrations under the PlanetLab and Bitbrains traces, as shown in Tables 3 and 4. Compared with the benchmark algorithm, the EQVC method reduces the number of VM migrations by approximately 71.85%. This is because when selecting VM to migrate, EQVC avoids inefficient VM migration. Thus, the number of VM migrations can be efficiently decreased using EQVC.

The ESV metrics can be used to measure an overall evaluation of both the EC and the SLAV, reflecting the combined level of energy consumption and QoS in the data center. As shown in Tables 3 and 4, we observe that EQVC outperforms the other approaches regarding the ESV metrics under the PlanetLab and Bitbrains traces. Additionally, the VM consolidation method that contains the RS algorithm has the highest ESV values in both traces, which demonstrates the importance of the VM selection algorithm. The experimental results show that EQVC maintains the best performance in both energy-saving and QoS guarantee with fewer VM migrations.

Overall, the results show that EQVC not only dramatically decreases the number of undesired VM migrations and saves energy but also efficiently guarantees QoS requirements. Our method can better use computing resources than the other methods, especially enabling the data center to enhance energy efficiency and QoS. Our technique allows for cloud service providers to reduce the cost of data centers and also improves the users' service experience to promote the further development of cloud computing.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a VM consolidation (EQVC) approach that can be employed in cloud data centers to reduce energy consumption and minimize the number of VM migrations when delivering guaranteed QoS. The feasibility of our method is established in four facts: (1) our approach effectively reduces the probability of host overloading; (2) our approach avoids undesired VM migrations, consequently minimizing the number of VM migrations; (3) our approach improves the utilization of host resources and prevents host re-overload; and (4) our approach has better performance under different workload traces.

The experimental results demonstrate that our method significantly outperforms other traditional methods in terms of energy-saving, guaranteed QoS levels, and electricity costs. Our approach also proves that by reducing invalid VM migrations, it is possible to improve data center QoS, which indicates that in the process of VM consolidation, we should avoid invalid VM migrations. In summary, our approach satisfies the requirements of data center low operating costs for cloud service providers, and satisfies the service experience of the users. Additionally, the method achieves a mutually beneficial situation for users and cloud service providers, which is essential to increasing the development of cloud computing.

Although CPU is one of the significant determinants of the physical host's energy consumption, other factors such as memory, disk storage, and network workload have an impact on energy cost. In future work, we will study the VM consolidation method under a multi-factorized load and will evaluate the proposed approach in a real cloud infrastructure.

REFERENCES

[1] M. N. O. Sadiku, S. M. Musa, and O. D. Momoh, "Cloud computing: Opportunities and challenges," *IEEE Potentials*, vol. 33, no. 1, pp. 34–36, Jan./Feb. 2014.
[2] J. Koomey, *Growth in Data Center Electricity Use 2005 to 2010.* Oakland, CA, USA: Analytics Press, Jul. 2011. [online] Available: http://www.analyticspress.com/datacenters.html
[3] S. Sharma and G. Sharma, "A review on secure and energy efficient approaches for green computing," *Int. J. Comput. Appl.*, vol. 138, no. 11, pp. 25–32, Jan. 2016.
[4] M. D. de Assunção, J.-P. Gelas, L. Lefèvre, and A.-C. Orgerie, "The green grid'5000: Instrumenting and using a grid with energy sensors," *Remote Instrum. eSci. Rel. Aspects*, vol. 5931, pp. 25–42, Dec. 2011.
[5] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Proc. 1st Int. Conf. Cloud Comput.*, Beijing, China, Dec. 2009, pp. 254–265.
[6] R. Nathuji and K. Schwan, "VirtualPower: Coordinated power management in virtualized enterprise systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 265–278, 2007.

[7] X. Zhu *et al.*, "1000 islands: Integrated capacity and workload management for the next generation data center," in *Proc. Int. Conf. Autom. Comput. (ICAC)*, Chicago, IL, USA, Jun. 2008, pp. 172–181.

[8] M. Mishra and A. Sahoo, "On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2011, pp. 275–282.

[9] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," *J. Netw. Comput. Appl.*, vol. 45, pp. 108–120, Oct. 2014.

[10] 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.

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

[12] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *Proc. 39th IEEE EUROMICRO Conf. Softw. Eng. Adv. Appl. (SEAA)*, Sep. 2013, pp. 357–364.

[13] R. K. Gupta and R. K. Pateriya, "Balance resource utilization BRU approach for the dynamic load balancing in cloud environment by using AR prediction model," *J. Organizational End User Comput.*, vol. 29, no. 4, pp. 24–50, Oct. 2017.

[14] Z. Li, C. Yan, X. Yu, and N. Yu, "Bayesian network-based virtual machines consolidation method," *Future Gener. Comput. Syst.*, vol. 69, pp. 75–87, Apr. 2017.

[15] G. Han, W. Que, G. Jia, L. Shu, and A. Jara, "An efficient virtual machine consolidation scheme for multimedia cloud computing," *Sensors*, vol. 16, no. 2, p. 246, 2016.

[16] J. Jiang, Y. Feng, J. Zhao, and K. Li, "DataABC: A fast ABC based energy-efficient live VM consolidation policy with data-intensive energy evaluation model," *Future Gener. Comput. Syst.*, vol. 74, pp. 132–141, Sep. 2017.

[17] M. A. Khoshkholghi, M. N. Derahman, A. Abdullah, S. Subramaniam, and M. Othman, "Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers," *IEEE Access*, vol. 5, pp. 10709–10722, 2017.

[18] Y. Wen, Z. Li, S. Jin, C. Lin, and Z. Liu, "Energy-efficient virtual resource dynamic integration method in cloud computing," *IEEE Access*, vol. 5, pp. 12214–12223, 2017.

[19] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, and C. He, "Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing," *Computing*, vol. 98, no. 3, pp. 303–317, 2016.

[20] M. A. H. Monil and A. D. Malony, "QoS-aware virtual machine consolidation in cloud datacenter," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2017, pp. 81–87.

[21] W. Qiu, Z. Qian, and S. Lu, "Multi-objective virtual machine consolidation," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 270–277.

[22] S. Y. Z. Fard, M. R. Ahmadi, and S. Adabi, "A dynamic VM consolidation technique for QoS and energy consumption in cloud environment," *J. Supercomput.*, vol. 73, no. 10, pp. 4347–4368, Oct. 2017.

[23] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proc. Conf. Power Aware Comput. Syst.*, 2008, p. 10.

[24] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2007, pp. 13–23.

[25] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 10, no. 12, pp. 33–37, 2007.

[26] K. Tsakalozos, V. Verroios, M. Roussopoulos, and A. Delis, "Live VM migration under time-constraints in share-nothing IaaS-clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2285–2298, Aug. 2017.

[27] S. Homsi, S. Liu, G. A. Chaparro-Baquero, O. Bai, S. Ren, and G. Quan, "Workload consolidation for cloud data centers with guaranteed QoS using request reneging," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 2103–2116, Jul. 2017.

[28] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct./Dec. 2015.

[29] Q. Zhang, M. F. Zhani, Q. Zhu, S. Zhang, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proc. ACM Int. Conf. Autonomic Comput. (ICAC)*, 2012, pp. 145–154.

[30] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Jan./Mar. 2014.

[31] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.

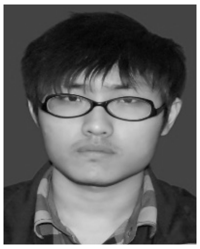
[32] P. J. Huber, *Robust Statistics*. New York, NY, USA: Wiley, 1981.

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

[34] K. Park and V. S. Pai, "CoMon: A mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, 2006.

[35] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proc. IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGrid)*, May 2015, pp. 465–474.

YAQIU LIU received the Ph.D. degree from the Harbin Institute of Technology University. He is currently a Professor with the College of Information and Computer Engineering, Northeast Forestry University. His current research interests include process control, distributed computing, cloud computing, intelligent control, soft computing, and model reconstruction.



XINYUE SUN received the B.S. degree from the North University of China. He is currently pursuing the master's degree in computer architecture with Northeast Forestry University. His current research interests include cloud computing and green computing.

WEI WEI (SM'17) received the M.S. and Ph.D. degrees from Xi'an Jiaotong University in 2005 and 2011, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Xi'an University of Technology. His academic interests are Internet of Things, big data, cloud computing, and image processing.

WEIPENG JING (M'16) received the Ph.D. degree from the Harbin Institute of Technology University. He is currently an Associate Professor with the College of Information and Computer Engineering, Northeast Forestry University. His research interests and expertise include modeling and scheduling for distributed computing systems, system reliability estimation, fault-tolerant computing, and cloud computing.

• • •