International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **2**, April **2018**

ICIC International ©2018 ISSN 1349-4198
pp. 455–467

BATCH ARRIVAL BASED PERFORMANCE EVALUATION OF A VM SCHEDULING STRATEGY IN CLOUD COMPUTING

Baoshuai Wang[1,2,3], Shunfu Jin[1,2,3,*] and Bing Qin[1,2,3]

[1]School of Information Science and Engineering
[2]Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province
Yanshan University
No. 438, Hebei Avenue, Qinhuangdao 066004, P. R. China
*Corresponding author: jsf@ysu.edu.cn

[3]Science and Technology on Communication Networks Laboratory
No. 589, West Zhongshan Road, Shijiazhuang 050081, P. R. China

Received August 2017; revised November 2017

Abstract. *With the rapid development of cloud computing, more and more people shift their workload on cloud data centers, is the energy consumption of virtual machines (VMs) is non-negligible. In order to reduce energy consumption and achieve green cloud service, we propose a novel VM scheduling strategy. All the VMs are divided into two groups: the main group and the reservation group. When the traffic load is light, the reservation group is deactivated and energy will be conserved. Considering that a job is divided into several tasks for parallel processing in multiple VMs, we establish a batch arrival queueing model with multiple VMs. By using Gauss Seidel method, we derive the steady-state distribution of the system model with two-dimensional Markov chain. Moreover, we evaluate the performance measures in terms of the blocking probability of tasks, the average response time of tasks and the energy saving rate of system. We provide numerical experiments with analysis and simulation to validate the proposed strategy and to estimate the influence of system parameters on performance measures. We establish a system cost function to trade off different performance measures, and develop an intelligent searching algorithm to optimize the system parameters.*
Keywords: Cloud computing, Data center, Energy saving, Batch arrival, Markov chain

1. **Introduction.** The growing demand for Internet services and cloud computing urges more large-scale data centers to be established. More data centers across the world are being built by cloud providers, such as Microsoft and Google, and they have more than 1 million servers in their infrastructures [1]. More data centers will cause more energy consumption. According to McKinsey's report [2], the total estimated electricity bill for data centers in 2010 was $11.5 billion, and by 2020 the worldwide carbon emission from data centers will be quadruple (commutative average growth rate (CAGR) $> 11\%$). Therefore, energy efficiency of data centers with cloud computing has been one of the hot research topics.

In [3], Dabbagh et al. introduced a prediction-based power management policy to reduce energy consumption. By classifying requests into multiple categories and estimating the number of requests in each category, the number of physical machines (PMs) could be predicted. Moreover, whether and when the idle PMs should be asleep or awake could be forecasted. In [4], in order to consolidate virtual machines (VMs) into a reduced number of active PMs while guaranteeing quality of service (QoS) requirements, Farahnakian et al. presented a novel dynamic VM consolidation approach called ant colony system based VM consolidation (ACS-VMC). With the ant colony system (ACS), the VM placement

could be adapted according to the workload. Florence et al. [5] devised an energy saving methodology based on dynamic voltage and frequency scaling scheme (DVFS). They analyzed the behavior of the given cloud request, and identified the associated type of algorithm with pattern analyzer. By using algorithm's asymptotic notations, the time complexity of the request was calculated. CPU frequency was scaled up or down using DVFS scheme to satisfy time complexity. In [6], Chen et al. proposed an adaptive DVFS scheme for multi-core embedded system to facilitate control over the tradeoff between energy and performance. Some effective energy saving strategies have been proposed in above studies. However, these studies are lack of performance evaluation and system optimization. We note that with a reasonable mathematical model, the system performance of the strategy could be improved.

In [7], Liao et al. proposed a dynamic power management policy by switching on/off a certain group of VMs. They established a mathematical model using queueing theory to determine the activation thresholds of VMs and study the energy-performance tradeoff in cloud data centers. In [8], Cao et al. developed a load distribution method with the constraints of energy and performance for cloud computing in cloud data centers. They formulated power allocation and load distribution in a cloud of clouds as optimization problems. They established a queueing model for a group of heterogeneous multicore servers with different sizes and speeds to study the energy-performance tradeoff. In [9], Kuehn and Mashaly presented an energy conservation strategy in which the data centers were controlled by a finite state machine (FSM). With a load-dependent control of server activations, VMs were allowed to be automatically consolidated in the FSM. They also established a queueing model for the study of the tradeoff between system energy and user performance which were reciprocal to each other. All the queueing models in the literature are from the view point of jobs. While in the practical cloud applications, when a job arrives at the system, this job is always divided into several tasks for parallel processing. So it is essential to model the energy saving strategies in cloud computing from the perspective of tasks.

The capacity of a cloud data center is usually planned according to the expected peak traffic load; however, the average load is about just 60% of the peak load [10]. When VMs in a cloud data center run at low utilization, such as 10% CPU utilization, the energy consumption is over 50% of the peak power [11]. In conventional data centers with static VM schedule, a great deal of energy is wasted. For this, we propose an energy saving based VM scheduling strategy with reservation VMs. All the VMs are divided into two groups: the main group and the reservation group. The main group is always activated. When the traffic load is light, the reservation group will be deactivated to conserve energy. When the traffic load is heavy, the reservation group will be activated to guarantee the user performance. In cloud data centers, when a job arrives at the system, this job is divided into several tasks for parallel processing in multiple VMs. From the point of view of tasks, we establish a batch arrival queueing model with two groups of VMs to capture the VM scheduling strategy proposed in this paper. Taking account of the number of all tasks in the system and the number of tasks in the reservation group, we constitute a two-dimensional Markov chain to evaluate the blocking probability of tasks, the average response time of tasks and the energy saving rate of system. Moreover, we provide numerical results to illustrate the tradeoff between energy consumption and user performance. We also establish a system cost function and develop an intelligent searching algorithm to optimize the VM scheduling strategy proposed in this paper.

The rest of this paper is organized as follows. In Section 2, we propose an energy saving based VM scheduling strategy and establish a type of batch arrival queueing model. In Section 3, we establish an analytical framework based on Markov chain to evaluate the

system performance. In Section 4, we present performance measures in terms of the blocking probability of tasks, the average response time of tasks and the energy saving rate of system. Numerical results are provided to verify the proposed strategy and to investigate the energy-performance tradeoff. In Section 5, we develop an intelligent searching algorithm to optimize the system threshold and the service rate with the minimum system cost. Finally, we conclude this paper in Section 6.

2. **Strategy Description and System Model.**

In this section, we propose an energy saving based VM scheduling strategy with reservation VMs in cloud data centers. Then, we establish a type of batch arrival queueing model with two groups of VMs accordingly.

2.1. **Strategy description.**

With the popularity of cloud computing, the problem of energy consumption on VMs in cloud data centers becomes more remarkable. For the purpose of saving energy and achieving green cloud service, we propose a VM scheduling strategy with reservation VMs.

All the VMs in a physical host are divided into two groups: the main group and the reservation group. The main group is always activated. The reservation group has three states: active state, deactive state and quasi-deactive state. When a job arrives at the system, this job is divided into several tasks for parallel processing in multiple VMs. When the number of tasks in the system is fewer, less VMs are necessary to execute tasks. So all the VMs in the reservation group can be closed to save energy. For this case, we say the reservation group is in deactive state. When the number of tasks in the system is greater, more VMs are necessary to execute tasks. So all the VMs in the reservation group will be opened to guarantee user performance. For this case, we say the reservation group is in active state. In order to activate/deactivate the reservation group appropriately, we introduce a threshold a as a system parameter. In practical applications, for the throughput-sensitive and delay-sensitive users, the threshold should be set lower, while from the perspective of improving energy efficiency, the threshold should be set higher. When the number of tasks in the system decreases to the threshold a, the reservation group will switch from active state to deactive state. Only when all the VMs in the reservation group are idle, can the reservation group be deactivated. We call the intermediate state before the reservation group switching from activated to deactivated as quasi-deactive state.

The state transition of the reservation group in our proposed VM scheduling strategy is shown in Figure 1.

1) When the reservation group is in deactive state, once a job with several tasks arrives at the system, the system will count the number of the tasks in the system. If there are more than a tasks in the system, the reservation group will switch to active state; otherwise, the reservation group will stay in deactive state.

2) When the reservation group is in active state, once a VM finishes its current task, the system will count the remainder tasks in the system. If the number of tasks in the system drops to the threshold a, the reservation group will be prepared to switch to deactive state, i.e., the reservation group will be in quasi-deactive state.

3) When the reservation group is in quasi-deactive state, if a VM in the main group finishes its current task and becomes available, the system will migrate one of the tasks in the reservation group to this VM. If a VM in the reservation group finishes its current task, this VM will keep idle. The tasks in the buffer will not be executed temporarily, but new jobs will arrive at the system. So the number of tasks in the buffer can only increase but not decrease. If there are more than a tasks in the system, the reservation group will

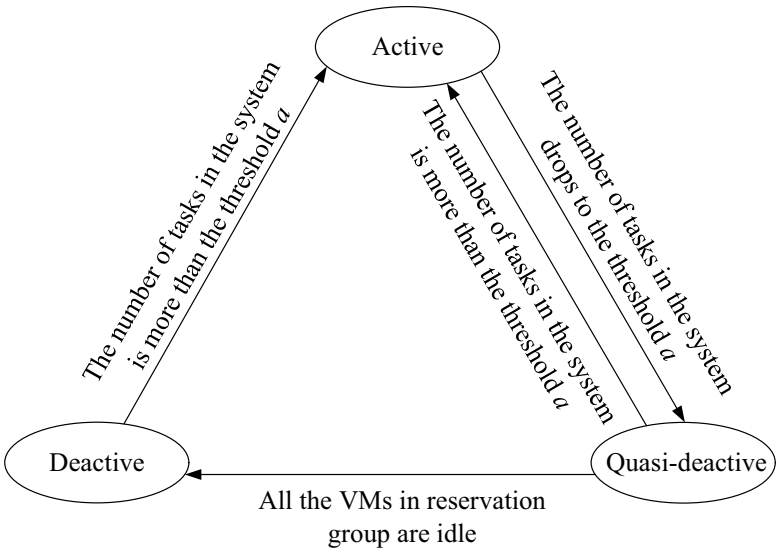


FIGURE 1. State transition of the reservation group

be activated again. If all the VMs in the reservation group are idle and the number of tasks in the system is equal to or less than the threshold a, this group will be deactivated.

2.2. **Model building.** In cloud data centers, a job is divided into several tasks for parallel processing in multiple VMs. The arrival of a job means a batch arrival of tasks. The number of tasks in a job is called as the job size. Based on the proposed VM scheduling strategy, we establish a batch arrival queueing model with two groups of VMs. In this system model, there are n VMs in the main group and m VMs in the reservation group. We assume that the jobs arrive at the system in a Poisson process with mean rate Λ and the job size ξ is geometrically distributed with parameter θ. So the job with x tasks arrives at the system in a Poisson process with mean rate $\lambda_x = \Lambda\theta(1-\theta)^{x-1}$, $x \in \{1, 2, \ldots\}$. Each VM serves only one task at a time and the service time of a task is supposed to be exponentially distributed with parameter μ. In order to ensure that all the VMs in the reservation group are busy when this group is in active state, we assume that the threshold a is no less than the sum of the numbers of VMs in both the main group and the reservation group, i.e., $a \geq n + m$. Moreover, we suppose the capacity of the buffer is r. So the capacity of the system is $N = n + m + r$.

Let random variable $I(t) = i$, $i \in \{0, 1, 2, \ldots, N\}$ be the total number of tasks in the system and $J(t) = j$, $j \in \{0, 1, 2, \ldots, m\}$ be the number of tasks in the reservation group at the time instant t. $I(t)$ is called as system level and $J(t)$ is called as system stage. With the assumptions above, $\{I(t), J(t), t \geq 0\}$ constitutes a continuous-time two-dimensional Markov chain with state space $\mathbf{\Omega} = \{(i, j) \mid i \in \{0, 1, 2, \ldots, N\}, j \in \{0, 1, 2, \ldots, m\}\}$.

For the two-dimensional Markov chain $\{I(t), J(t), t \geq 0\}$, we define the steady-state distribution $\pi_{i,j}$ as follows:

$$\pi_{i,j} = \lim_{t\to\infty} P\{I(t) = i, J(t) = j\}, \quad i \in \{0, 1, 2, \ldots, N\}, \ j \in \{0, 1, 2, \ldots, m\} \tag{1}$$

3. **Model Analysis.** In this section, we derive the queueing model established in Section 2 in steady state.

According to the working principle of the proposed VM scheduling strategy, via one step transition, the system level will decrease one, remain unchanged or increase many.

1) The system level changing from i to $i - 1$ means the system level decreases one. For the case that the system stage j fixes at 0, if $i \leq n$, the transition rate is $i\mu$; if

$n+1 \le i \le a$, the transition rate is $n\mu$. For the case that the system stage j decreases one, if $a-m+j \le i \le a$, the transition rate is $(n+j)\mu$. For the case that the system stage j fixes at m, if $a+1 \le i \le N$, the transition rate is $(n+m)\mu$.

2) The system level changing from i to $i+y$, $y \in \{1,2,\ldots,N-i\}$ means the system level increases many. The transition rate will be λ_y for following cases: the system stage j fixes at 0 and $0 < i+y \le a$; the system stage $1 \le j \le m-1$ remains unchanged and $(a-m+j \le i < a) \bigcap (a-m+j < i+y \le a)$; the system stage j fixes at m and $(a \le i < N-1) \bigcap (a < i+y \le N-1)$; the system stage j ranges from 0 to m and $(0 \le i \le a) \bigcap (a+1 \le i+y \le N-1)$; the system stage j ranges from $1 \le j \le m-1$ to m and $(a-m+j \le i \le a) \bigcap (a+1 \le i+y \le N-1)$. The transition rate will be $\sum_{k=0}^{\infty} \lambda_{y+k}$ for following cases: the system stage j fixes at m and $(a \le i \le N-1) \bigcap (i+y=N)$; the system stage j ranges from 0 to m and $(0 \le i \le a) \bigcap (i+y=N)$; the system stage j ranges from $1 \le j \le m-1$ to m and $(a-m+j \le i \le a) \bigcap (i+y=N)$.

3) The system level remaining unchanged means the number of all tasks in the system is unchanged. For the case that the system stage j fixes at 0, if the system level $i=0$, the transition rate is $-\Lambda$; if $0 < i \le n$, the transition rate is $-\Lambda - i\mu$; if $n+1 \le i \le a$, the transition rate is $-\Lambda - n\mu$. For the case that the system stage remains unchanged and $1 \le j \le m-1$, if $a-m+j \le i \le a$, the transition rate is $-\Lambda-(n+j)\mu$. For the case that the system stage j fixes at m, if $a \le i < N$, the transition rate is $-\Lambda-(n+m)\mu$; if $i=N$, the transition rate is $-(n+m)\mu$.

Based on the discussions above, the state transition of the queue model is illustrated in Figure 2.

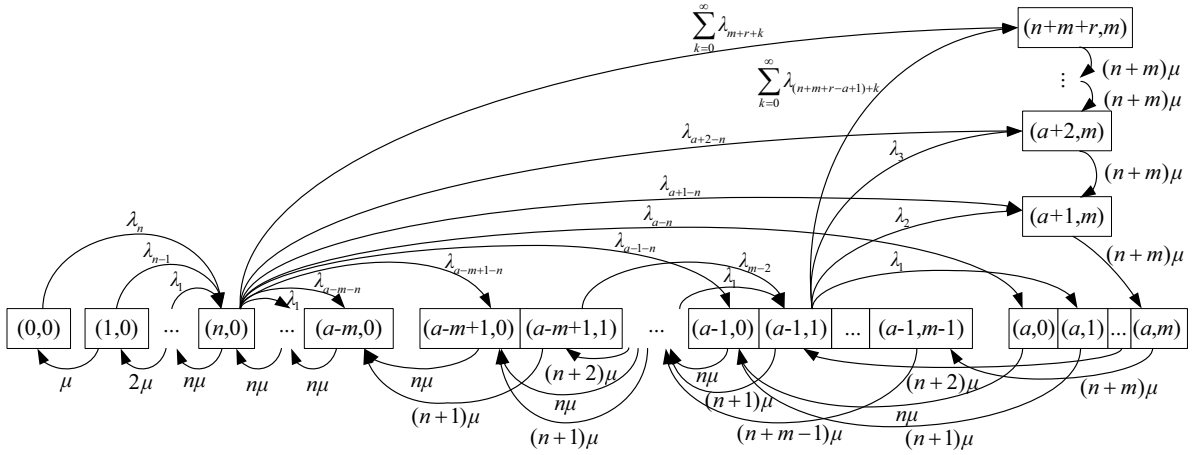


FIGURE 2. State transition of the queue model

According to different system stages j, we get the balance equations of the queue model as follows:

1) $j=0$ means the reservation group is in deactive state. For this case, the system service rate on state $(i,0)$ is $\begin{cases} i\mu, & i \le n \\ n\mu, & i > n \end{cases}$. The balance equations are given as follows:

$$\begin{cases} \mu\pi_{1,0} - \Lambda\pi_{0,0} = 0, & i = 0 \\ \sum_{k=0}^{i-1} \lambda_{i-k}\pi_{k,0} + (i+1)\mu\pi_{i+1,0} - (\Lambda + i\mu)\pi_{i,0} = 0, & 0 < i < n \\ \sum_{k=0}^{i-1} \lambda_{i-k}\pi_{k,0} + n\mu\pi_{i+1,0} - (\Lambda + n\mu)\pi_{i,0} = 0, & n \le i < a - m \\ \sum_{k=0}^{i-1} \lambda_{i-k}\pi_{k,0} + n\mu\pi_{i+1,0} + (n+1)\mu\pi_{i+1,1} - (\Lambda + n\mu)\pi_{i,0} = 0, & a - m \le i \le a \\ \pi_{i,0} = 0, & a < i \le N \end{cases} \tag{2}$$

2) $1 \le j \le m-1$ means the reservation group is in quasi-deactive state. For this case, the system service rate on state (i, j) is $(n + j)\mu$. The balance equations are given as follows:

$$\begin{cases} \pi_{i,j} = 0, & 0 \le i < a - m + j \\ (n+j+1)\mu\pi_{i+1,j+1} - (\Lambda + (n+j)\mu)\pi_{i,j} = 0, & i = a - m + j \\ \sum_{k=a-m+j}^{i-1} \lambda_{i-k}\pi_{k,j} + (n+j+1)\mu\pi_{i+1,j+1} - (\Lambda + (n+j)\mu)\pi_{i,j} = 0, & a - m + j < i < a \\ \sum_{k=a-m+j}^{i-1} \lambda_{i-k}\pi_{k,j} - (\Lambda + (n+j)\mu)\pi_{i,j} = 0, & i = a \\ \pi_{i,j} = 0, & a < i \le N \end{cases} \tag{3}$$

3) $j = m$ and $i > a$ mean the reservation group is in active state, and $j = m$ and $i = a$ mean the reservation group is in quasi-deactive state. For these two cases, the system service rate on state (i, m) is $(n + m)\mu$. The balance equations are given as follows:

$$\begin{cases} \pi_{i,m} = 0, & i < a \\ (m+n)\mu\pi_{i+1,m} - (\Lambda + (n+m)\mu)\pi_{i,m} = 0, & i = a \\ \sum_{k=0}^{i-1}\sum_{h=0}^{m} \lambda_{i-k}\pi_{k,h} - (\Lambda + (n+m)\mu)\pi_{i,m} + (m+n)\mu\pi_{i+1,m} = 0, & a < i < N \\ \sum_{k=0}^{i-1}\sum_{h=0}^{m} \pi_{k,h} \sum_{b=i-k}^{\infty} \lambda_b - (n+m)\mu\pi_{i,m} = 0, & i = N \end{cases} \tag{4}$$

By using Gauss Seidel method, solve Equations (2)-(4). Combining normalized condition $\sum_{i=0}^{N}\sum_{j=0}^{m} \pi_{i,j} = 1$, we can obtain the steady-state probability distribution $\pi_{i,j}$ recurrently.

4. Performance Measures and Experiments Results.

In this section, we present performance measures to evaluate the proposed VM scheduling strategy and provide statistical experiments to investigate the energy-performance tradeoff.

4.1. Performance measures.

With the proposed energy saving strategy, from the perspective of QoS of users, we will investigate the blocking probability and the average response time of tasks. On the other hand, from the perspective of energy efficiency of the proposed strategy, we will investigate the energy saving rate of system.

We define the blocking probability P_{block} of a task as the probability that this task cannot enter the buffer due to the finite capacity. We focus on an arbitrary task arriving at the system in batch called tagged task T, we suppose that the position of the tagged task T in a batch is uniformly distributed. The probability P_z that the tagged task T

belongs to a job with z tasks is given as follows:

$$P_z = \frac{z \times \theta(1-\theta)^{z-1}}{E[\xi]} \tag{5}$$

where $E[\xi]$ is the mean value of the job size, i.e., $E[\xi] = \frac{1}{\theta}$.

The blocking probability P_{block} of a task is given as follows:

$$P_{block} = \sum_{i=0}^{N} \sum_{j=0}^{m} \sum_{z=N-i+1}^{\infty} \pi_{i,j} \times P_z \times \frac{z-(N-i)}{z} \tag{6}$$

We define the response time of a task as the time duration from the arrival of a task to the execution termination of this task. The average response time W of tasks is the sum of the average wait time of tasks and the average execution time of tasks. By using Little's law, the average response time W of tasks is given as follows:

$$W = \frac{1}{\lambda_e}\left(\sum_{i=n+1}^{a}(i-n)\pi_{i,0} + \sum_{i=a-m+j}^{a}\sum_{j=1}^{m-1}(i-n-j)\pi_{i,j} + \sum_{i=a}^{N}(i-n-m)\pi_{i,m}\right) + \frac{1}{\mu} \tag{7}$$

where λ_e is the effective arrival rate of tasks, $\lambda_e = \Lambda \times E[\xi] \times (1 - P_{block})$.

We define the energy saving rate of system as the energy conservation per unit time in the proposed VM scheduling strategy. Only when the reservation group is in deactive state, can the energy be saved. Let C be the energy-saving degree of a VM per unit time during deactive state. The energy saving rate S of system is given as follows:

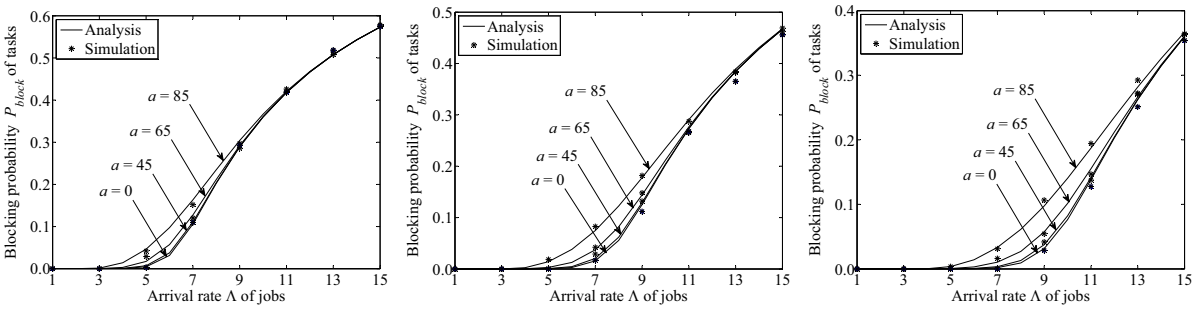
$$S = C \times m \times \sum_{i=0}^{a} \pi_{i,0} \tag{8}$$

4.2. **Statistical experiments.** In this section, we provide some statistical experiments to estimate the influence of system parameters on system performance for the proposed strategy in this paper, and give simulation results to validate the system model established in this paper. We also add numerical experiments about the conventional strategy without reservation, and conduct a comparison between the conventional strategy and our proposed strategy. The analysis results are carried out in Matlab 2010a on Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz 3.60 GHz, 6.00 GB RAM. The simulation results are obtained by averaging over 10 independent runs using MyEclipse2014. We create the TASK class with attributes in terms of UNARRIVE, WAIT, RUN, FINISH and BLOCK to record the state of a task. We also create the SERVER class with attributes in terms of OFF, IDLE and BUSY to record the state of a VM. From Figures 3-5, we can see that the analysis results match well with the simulation results. In the established mathematical model of system, the capacity of system is supposed to be finite. For any experimental parameters, the system will always achieve steady-state. The setting of the parameters will not affect the trends and laws of the experiments results. As an example, in numerical experiments, we set the parameters as follows: $n = 20$; $m = 20$; $r = 70$; $\theta = 0.2$ and $C = 3$.

By setting different service rates μ, Figure 3 demonstrates the blocking probability P_{block} of tasks versus the arrival rate Λ of jobs for different thresholds a.

In Figure 3, the lines with $a = 45$, 65 and 85 are for our proposed energy saving strategy, and the line with $a = 0$ is for the conventional strategy without reservation. We notice that the blocking probability of tasks in our proposed strategy is a bit higher than that in the conventional strategy without reservation.

From Figure 3, we observe that for the same threshold a, the blocking probability P_{block} of tasks will increase as the arrival rate Λ of jobs increases. This is because as the arrival



(a) $\mu = 0.8$ (b) $\mu = 1.0$ (c) $\mu = 1.2$

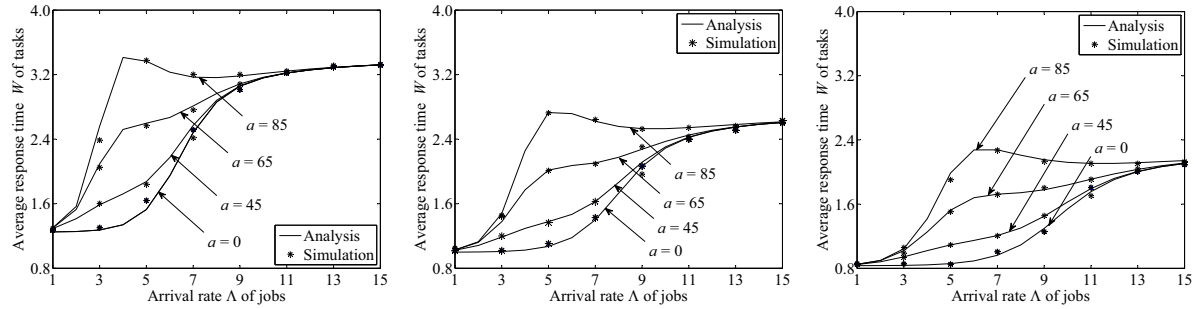
FIGURE 3. Blocking probability P_{block} of tasks

rate of jobs increases, the number of tasks in the system becomes greater, and the buffer of the system is more likely to be overflowed. So the blocking probability of tasks will increase accordingly.

For the same arrival rate Λ of jobs, we obtain following observations. When the arrival rate is smaller, all the tasks in the system can be executed. No matter whatever the system threshold is, the blocking probability of tasks always tends to 0. When the arrival rate is larger, the smaller the system threshold is, the more likely the reservation group is in active state, and the system can accommodate more tasks. So the blocking probability of tasks is lower. When the arrival rate further increases, the reservation group is more likely in active state even for a greater system threshold. So the system threshold has no impact on the blocking probability of tasks.

Combining Figures 3(a)-3(c), we notice that the blocking probability P_{block} of tasks will decrease as the service rate μ increases. This is because as the service rate increases, tasks finish service more quickly, and then the VMs can execute more tasks. So the blocking probability of tasks decreases.

By setting different service rates μ, Figure 4 illustrates the average response time W of tasks versus the arrival rate Λ of jobs for different thresholds a.



(a) $\mu = 0.8$ (b) $\mu = 1.0$ (c) $\mu = 1.2$

FIGURE 4. Average response time W of tasks

Similar to Figure 3, the lines with $a = 45,\ 65$ and 85 are for our proposed energy saving strategy, and the line with $a = 0$ is for the conventional strategy without reservation. We notice that the average response time of tasks in our proposed strategy is a bit larger than that in the conventional strategy without reservation.

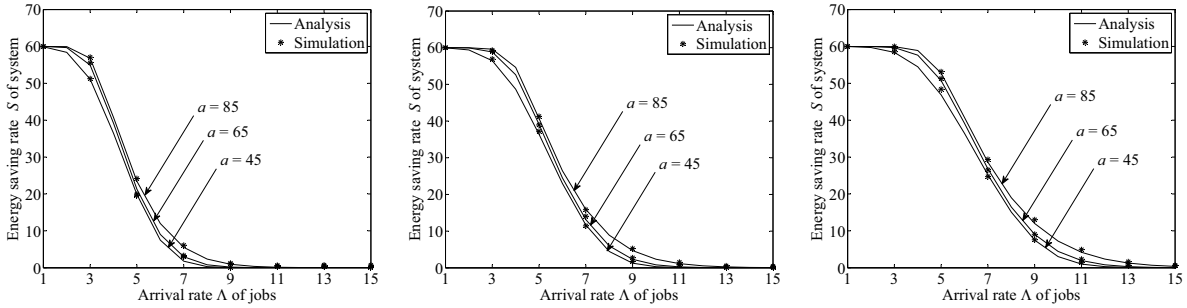
From Figure 4, we observe that for the same threshold a, as the arrival rate Λ of jobs increases, the average response time W of tasks presents three stages.

When the arrival rate of jobs is smaller, the average response time of tasks will increase as the arrival rate of jobs increases. This is because when the arrival rate of jobs is smaller, the reservation group is more likely in deactive state. As the arrival rate increases, more tasks have to wait in the buffer, and the average waiting time of tasks will increase. So the average response time of tasks will be longer. When the arrival rate of jobs is larger, as the arrival rate of jobs increases, the average response time of tasks will increase slowly, and even decrease slightly. This is because as the arrival rate of jobs increases, the reservation group is more likely in active state. The throughput of system increases, so the average response time of tasks will increase slowly, and even decrease slightly. When the arrival rate of jobs further increases, the reservation group is more likely in active state. Because of the finite capacity, the average response time of tasks will flatten and tend to a certain value.

We also can see that for the same arrival rate Λ of jobs: When $1 \leq \Lambda \leq 13$, the average response time W of tasks will increase as the threshold a increases. This is because when the system threshold is smaller, the reservation group is more likely in active state. So the average response time of tasks will be lower. When the arrival rate of jobs further increases, i.e., $\Lambda \geq 13$, the reservation group is more likely in active state even for a greater system threshold. So the system threshold has no impact on the average response time of tasks. In other words, when the arrival rate of jobs is greater enough, the average response time of tasks is almost invariable for all the system thresholds.

Combining Figures 4(a)-4(c), we notice that the average response time W of tasks will decrease as the service rate μ increases. This is because as the service rate increases, the service time of a task is shorter, and the time of tasks waiting in the buffer is shorter. So the average response time of tasks decreases. The experimental results are consistent with our predictions.

By setting different service rates μ, Figure 5 illustrates the energy saving rate S of system versus the arrival rate Λ of jobs for different thresholds a.



(a) $\mu = 0.8$ (b) $\mu = 1.0$ (c) $\mu = 1.2$

FIGURE 5. Energy saving rate S of system

From Figure 5, we observe that for the same threshold a, the energy saving rate S of system will decrease as the arrival rate Λ of jobs increases. When the arrival rate of jobs is $1 \leq \Lambda \leq 11$, as the arrival rate of jobs increases, the number of tasks in the system is greater, and the probability of the reservation group to be active will increase accordingly. For this case, the energy saving rate of system will decrease. When the arrival rate of jobs increases to a certain value, i.e., $\Lambda \geq 11$, the reservation group is more likely to be active, and the energy saving rate of system tends to 0.

We also notice that for the same arrival rate Λ of jobs, when $1 \leq \Lambda \leq 11$, the energy saving rate S of system will increase as the threshold a increases. This is because when

the system threshold is greater, the reservation group is more likely in deactive state, so the energy saving rate of system will be greater. As the arrival rate of jobs further increases, the reservation group is more likely in active state even for a greater system threshold. So the system threshold has no impact on the energy saving rate of tasks.

Combining Figures 5(a)-5(c), we notice that the energy saving rate S of system will increase as the service rate μ increases. This is because as the service rate increases, the number of tasks in the system is less possible to reach the system threshold, and then the reservation group is more likely in deactive state. So the energy saving rate of system increases.

Compared with the conventional strategy without reservation, we find that our proposed energy saving strategy performs better on saving energy consumption. However, we have to sacrifice a bit of QoS of users, such as the blocking probability and the average response time of tasks. For guaranteeing the QoS of users, the system threshold should be set lower. While for reducing energy consumption, the system threshold should be set higher. In practical cloud systems, the tradeoff among the blocking probability of tasks, the average response time of tasks and the energy saving rate of system should be investigated when setting the system threshold in our proposed VM scheduling strategy.

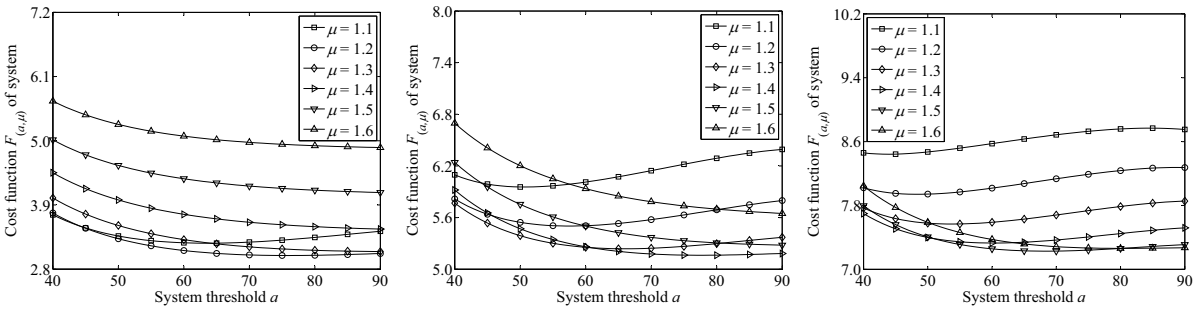
5. **System Optimization.** In this section, we establish a cost function of system to trade off different performance measures and develop an improved teaching-learning-based optimization (TLBO) algorithm to obtain the optimal combination of the system threshold and the service rate.

In cloud data centers, a higher blocking probability of tasks will make the cloud service provider lose more customers, a longer average response time of tasks will make users impatient, a larger service rate will increase the expenditure on servers. While, cloud data centers will benefit from a greater energy saving rate of system. For this, we establish a cost function $F_{(a,\mu)}$ of system as follows:

$$F_{(a,\mu)} = f_b P_{block} + f_w W + f_\mu \mu - f_s S \tag{9}$$

where f_b, f_w, f_μ and f_s are the impact factors of the blocking probability of tasks, the average response time of tasks, the expenditure on cloud data centers and the energy saving rate of system, respectively on the system cost. P_{block}, W and S are obtained in Equations (6)-(8), and μ is a system parameter to be determined.

We provide numerical experiments to investigate the change trend of the system cost function versus the system threshold, the service rate and the arrival rate of jobs. In the numerical experiments, we employ the parameters used in Subsection 4.2, and set the impact factors $f_b = 5.0$, $f_w = 0.8$, $f_\mu = 10.0$ and $f_s = 0.2$ as an example.



(a) $\Lambda = 5$ (b) $\Lambda = 6$ (c) $\Lambda = 7$

FIGURE 6. The cost function $F_{(a,\mu)}$ of system

TABLE 1. Iteration algorithm to obtain the optimal combination (a^*, μ^*)

Step 1: Set the number N of students and the maximum iterations L. Initialize the current iteration as $l = 1$, the combination of system threshold and service rate as $(a^*, \mu^*) = 0$, and the minimum cost function as $F^*_{(a,\mu)} = 0$. Set the initial system threshold as $a = 40$, the upper boundary of the system threshold as $a_{\max} = 90$.

Step 2: Initialize each student as $(a, \mu)^n$, $n \in \{1, 2, \ldots, N\}$, $\mu \in [1, 2]$ using chaotic equations and calculate the cost function $F^n_{(a,\mu)}$:

$(a, \mu)^1 = rand + 1$

calculate the cost function $F^1_{(a,\mu)}$ with $(a, \mu)^1$

For $n = 2 : N$

$(a, \mu)^n = r \times \left((a, \mu)^{n-1} - 1\right) \times \left(2 - (a, \mu)^{n-1}\right) + 1$

calculate the cost function $F^n_{(a,\mu)}$ with $(a, \mu)^n$

End

% r is the chaotic factor, $r = 3.85$.

Step 3: Calculate the mean value $(a, \mu)_{mean}$ for all students and select the student with the minimum cost function as a teacher $(a, \mu)_{teacher}$:

$$(a, \mu)_{mean} = \underset{n \in \{1,2,\ldots,N\}}{\text{mean}} \{(a, \mu)^n\},\ (a, \mu)_{teacher} = \underset{n \in \{1,2,\ldots,N\}}{\text{argmin}} \left\{F^n_{(a,\mu)}\right\}$$

Step 4: The teaching operation:

For $n = 1 : N$

$g = round(1 + rand)$

$(a', \mu')^n = w \times (a, \mu)^n + rand \times ((a, \mu)_{teacher} - g \times (a, \mu)_{mean})$

% w is the weight factor, $w = w_{\max} - (w_{\max} - w_{\min}) \times (l/L)$.

% $w_{\max}$ and $w_{\min}$ are the maximum and minimum weight factors.

If $F^n_{(a',\mu')} < F^n_{(a,\mu)}$

$(a, \mu)^n = (a', \mu')^n$

End

End

Step 5: The learning operation:

For $n = 1 : N$

randomly select another student $(a, \mu)^s$, $(s \neq n)$

If $F^n_{(a,\mu)} > F^s_{(a,\mu)}$

$(a', \mu')^n = w \times (a, \mu)^n + rand \times ((a, \mu)^n - (a, \mu)^s)$

Else

$(a', \mu')^n = w \times (a, \mu)^n + rand \times ((a, \mu)^s - (a, \mu)^n)$

End

If $F^n_{(a',\mu')} < F^n_{(a,\mu)}$

$(a, \mu)^n = (a', \mu')^n$

End

End

Step 6: Check the number of iterations:

If $l < L$

$l = l + 1$, go to **Step 3**

End

Step 7: Select the minimum cost function:

If $F^*_{(a,\mu)} = 0 \left\| \min_{n \in \{1,2,\ldots,N\}} \left\{F^n_{(a,\mu)}\right\} < F^*_{(a,\mu)}\right.$

$$F^*_{(a,\mu)} = \min_{n \in \{1,2,\ldots,N\}} \left\{F^n_{(a,\mu)}\right\},\ (a^*, \mu^*) = \underset{n \in \{1,2,\ldots,N\}}{\text{argmin}} \left\{F^n_{(a,\mu)}\right\}$$

End

If $a < a_{\max}$

$a = a + 1$, go to **Step 2**

End

Step 8: Output (a^*, μ^*)

By setting different arrival rates Λ of jobs, Figure 6 shows the change trend for the cost function $F_{(a,\mu)}$ of system in relation to the system threshold a and the service rate μ.

From Figure 6, we conclude that there is an optimal combination of the system threshold a^* and the service rate μ^* with the minimum system cost function $F^*_{(a,\mu)}$. For the batch arrival based system model established in this paper, the performance measures are difficult to be given in a close-form. The monotonicity of the system cost function is uncertain. The traditional optimization algorithms, such as the steepest descent optimization method or the Lagrange multiplier method, are inappropriate to get the optimal system parameters. So we develop an improved TLBO intelligent searching algorithm to get the optimal combination (a^*, μ^*) with the minimum system cost function $F^*_{(a,\mu)}$.

The TLBO intelligent searching algorithm is a newly developed and efficient metaheuristic optimization method which imitates the natural phenomena of knowledge dissemination [12]. The advantage of this algorithm is that it does not need any algorithm-specific parameters, and it is easy to understand. In the improved TLBO searching algorithm, we use chaotic equations to initialize a group of (a, μ) as the students. By this way, the initialization is more random. Moreover, we introduce a weight factor [13] to promote the teaching and the learning operations.

The main steps of the improved TLBO intelligent searching algorithm are given in Table 1. By employing the parameters used in Figure 6, and setting $N = 100$, $L = 50$ as an example, we obtain the optimal combination of the system threshold and the service rate. The results are summarized in Table 2.

TABLE 2. Optimal combination (a^*, μ^*) and minimum cost function $F^*_{(a,\mu)}$

Arrival rate Λ of jobs	Optimal combination (a^*, μ^*)	Minimum cost function $F^*_{(a,\mu)}$
5	$(80, 1.2254)$	3.0239
6	$(76, 1.3942)$	5.1618
7	$(71, 1.5204)$	7.2241

6. **Conclusions.** In this paper, for reducing energy consumption and achieving green cloud service, we proposed a novel VM scheduling strategy with VM reservation. Considering that jobs are divided into several tasks for parallel processing in multiple VMs, we established a batch arrival based two-dimensional Markov chain model, and derived the steady-state distribution of the system model by using Gauss Seidel method. Then we evaluated the system performance measures in terms of the blocking probability of tasks, the average response time of tasks and the energy saving rate of system. We provided statistical experiments with analysis and simulation to investigate the influence of system parameters on system performance with the proposed strategy. We note that the system threshold and the service rate have major impact on the performance measures. Accordingly, we established a system cost function to show the trade-off among different performance measures and the expenditure on VMs. By initializing the students with chaotic equations and introducing a weight factor, we developed an improved TLBO based intelligent searching algorithm, and jointly optimized the system threshold and the service rate with the minimum system cost function.

In future work, we will investigate the dependence between the service rate and the number of VMs in the reservation group with the energy saving strategy.

Acknowledgment. This work was supported in part by National Natural Science Foundation (No. 61472342), Hebei Province Natural Science Foundation (No. F2017203141) and Open Project of Science and Technology on Communication Networks Laboratory (No. KX17600025), China.

REFERENCES

[1] S. Fard, M. Ahmadi and S. Adabi, A dynamic VM consolidation technique for QoS and energy consumption in cloud environment, *The Journal of Supercomputing*, pp.1-22, 2017.

[2] A. Hamee, A. Khoshkbarforoushha, R. Ranjan et al., A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, *Computing*, vol.98, no.7, pp.751-774, 2016.

[3] M. Dabbagh, B. Hamdaoui, M. Guizani et al., Toward energy-efficient cloud computing: Prediction, consolidation and overcommitment, *IEEE Network*, vol.29, no.2, pp.56-61, 2015.

[4] F. Farahnakian, A. Ashraf, T. Pahikkala et al., Using ant colony system to consolidate VMs for green cloud computing, *IEEE Trans. Services Computing*, vol.8, no.2, pp.187-198, 2015.

[5] A. Florence, V. Shanthi and C. Simon, Energy conservation using dynamic voltage frequency scaling for computational cloud, *The Scientific World Journal*, 2016.

[6] Y. Chen, M. Chang, W. Liang et al., Performance and energy efficient dynamic voltage and frequency scaling scheme for multicore embedded system, *Proc. of IEEE International Conference on Consumer Electronics*, Las Vegas, NV, pp.58-59, 2016.

[7] D. Liao, K. Li, G. Sun et al., Energy and performance management in large data centers: A queuing theory perspective, *Proc. of International Conference on Computing, Networking and Communications*, Garden Grove, CA, pp.287-291, 2015.

[8] J. Cao, K. Li and I. Stojmenovic, Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers, *IEEE Trans. Computers*, vol.63, no.1, pp.45-58, 2014.

[9] P. Kuehn and M. Mashaly, Automatic energy efficiency management of data center resources by load-dependent server activation and sleep modes, *Ad Hoc Networks*, vol.25, pp.497-504, 2015.

[10] T. Benson, A. Akella and D. Maltz, Network traffic characteristics of data centers in the wild, *Proc. of the 10th ACM SIGCOMM Conference on Internet Measurement*, Melbourne, Australia, pp.267-280, 2010.

[11] G. Chen, W. He, J. Liu et al., Energy-aware server provisioning and load dispatching for connection-intensive Internet services, *Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, pp.337-350, 2008.

[12] R. Rao, V. Savsani and D. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *Computer-Aided Design*, vol.43, no.3, pp.303-315, 2011.

[13] H. Ouyang, L. Gao, X. Kong et al., Teaching-learning based optimization with global crossover for global optimization problems, *Applied Mathematics and Computation*, vol.265, pp.533-556, 2015.