J SYST SCI SYST ENG
Vol. 28, No. 2, April 2019, pp. 194–210
DOI: https://doi.org/10.1007/s11518-018-5401-9
ISSN: 1004-3756 (paper), 1861-9576 (online)
CN 11-2983/N

A Virtual Machine Scheduling Strategy with a Speed Switch and a Multi-Sleep Mode in Cloud Data Centers

Shunfu Jin,[a] Shanshan Hao,[a] Xiuchen Qie,[a] Wuyi Yue[b]

[a]School of Information Science and Engineering, Yanshan University, Qinhuangdao, China
jsf@ysu.edu.cn (✉), ashanshan931106@126.com, qiexiuchen@126.com
[b]Department of Intelligence and Informatics, Konan University, Kobe, Japan
yue@konan-u.ac.jp

Abstract. With the rapid growth of energy costs and the constant promotion of environmental standards, energy consumption has become a significant expenditure for the operating and maintaining of a cloud data center. To improve the energy efficiency of cloud data centers, in this paper, we propose a Virtual Machine (VM) scheduling strategy with a speed switch and a multi-sleep mode. In accordance with the current traffic loads, a proportion of VMs operate at a low speed or a high speed, while the remaining VMs either sleep or operate at a high speed. Commensurate with our proposal, we develop a continuous-time queueing model with an adaptive service rate and a partial synchronous vacation. We construct a two dimensional Markov chain based on the total number of requests in the system and the state of all the VMs. Using a matrix geometric solution, we mathematically estimate the energy saving level and the response performance of the system. Numerical experiments with analysis and simulation show that our proposed VM scheduling strategy can effectively reduce the energy consumption without significant degradation in response performance. Additionally, we establish a system utility function to trade off the different performance measures. In order to determine the optimal sleep parameter and the maximum system utility function, we develop an improved Firefly intelligent searching Algorithm.

Keywords: Cloud data center, virtual machine scheduling, speed switch, multi-sleep, matrix geometric solution, utility function, improved Firefly Algorithm

1. Introduction

The rapid development of information technology and the explosive growth in global data have generated enormous demand for cloud computing. Consequently, Cloud Data Centers (CDCs) are growing exponentially, both in number and in size, to provide universal service. International Data Corporation (IDC) predicts that the total number of CDCs deployed worldwide will peak at 8.6 million in 2017 (Hintemann et al. 2016). Currently, high energy consumption and serious environmental pollution are significant factors restricting the development of CDCs. One of the key challenges in constructing green CDCs is reducing energy consumption without seriously degrading the Quality of Service (QoS).

In CDCs, besides the necessary energy consumption produced by providing service for cloud users, a large amount of energy is wasted maintaining excess service capacity (Hameed et al. 2016, Salimian et al. 2012). All the Virtual Machines (VMs) in CDCs remain open waiting for the arrivals of cloud users, even in the night and early morning.

During those hours, the utilization of VMs is merely 5 to 10 percent. However, the energy consumption of an idle VM is 60 to 80 percent of that of a busy VM (Duan et al. 2015). In addition, inappropriate VM scheduling can also result in superfluous energy consumption. Researchers have therefore directed their focus on improving energy efficiency by reducing the amount of wasted energy in CDCs.

The energy consumption of a VM is approximately in line with the CPU utilization, so the most direct method of conserving energy is to operate all the VMs at lower voltage and frequency (Qavami et al. 2014, Farahnakian et al. 2015). One of the common techniques for optimizing energy consumption in CDCs is to engage dynamic power management (DPM). DPM refers to dynamic CPU energy consumption and CPU processing speed adjustment according to the current traffic load. In Li(2016), Li proved that if the application environment and average energy consumption are given, there is an optimal speed scheme that minimizes the average response time of requests. In Wang et al.(2011), Wang et al. presented a workload predictor based on online Bayes classifier and a novel DPM technique based on an adaptive reinforcement learning algorithm to reduce the energy consumption in stochastic dynamic systems.

In Chen et al.(2016), Chen et al. proposed a Dynamic Voltage and Frequency Scaling (DVFS) scheme based on DPM technique, by which the best fitting voltage and frequency for a multi-core embedded system are dynamically predicted. All the methods based on DPM technique mentioned above can improve energy efficiency from the perspective of reducing the energy consumption of each VM in CDCs. However, all the VMs in the CDCs remain open all the time, even though there are no requests in CDCs. Even when operating at low-speed and in low-voltage mode, the accumulated energy consumption by thousands of VMs in CDCs is significant.

In respect to the low utilization of VMs in CDCs, inducing some VMs to enter a sleep state or a power-off state during lower workload hours can also save energy (Shen et al. 2017, Dabbagh et al. 2015). In Chou et al.(2016), Chou et al. proposed a DynSleep scheme. DynSleep dynamically postpones the processing of some requests, creating longer idle periods. It says that the use of a deep sleep mode can save more energy. In Dabbagh et al.(2015), Dabbagh et al. proposed an integrated energy-aware resource provisioning framework for CDCs. This framework first predicts the number of cloud users that will arrive at CDCs in the near future, then estimates the number of VMs that are needed to serve those cloud users.

In Liao et al.(2015), Liao et al. proposed an energy-efficient strategy, which dynamically switches two backup groups of servers on and off according to different thresholds. Using the methods above, energy can be conserved by decreasing the number of VMs running in the system. However, few methods can accurately estimate the behavior of requests. Their arrivals and departures are stochastic. Pushing VMs to enter a sleep state or a power-off state based on only the predicted behavior of requests is very risky, and might lead to a significant sacrifice of the response performance.

Typically, if the traffic load is very heavy, all the VMs in CDCs will operate at a high speed so that cloud users can be served faster and the average delay can be reduced. On the other hand, if the traffic load is very light, some VMs will operate at a low speed while the remaining VMs go to sleep so that energy consumption can be greatly reduced without significant response performance degradation.

For these, in this paper, by applying a DPM technique and introducing a sleep mode, we propose a VM scheduling strategy with a speed switch and a multi-sleep mode. Accordingly, we establish a continuous-time queueing model with an adaptive service rate and a partial synchronous vacation to investigate the behavior of cloud users and all the VMs in CDCs with the proposed VM scheduling strategy. From the perspective of the total number of cloud users in the CDC and the state of all the VMs, we construct a two dimensional Markov chain to analyze the queueing model. Moreover, we mathematically and numerically

evaluate the energy saving level of the system and the average delay of requests. In order to achieve a reasonable balance between different performance measures, we establish a system utility function. Finally, we present an improved Firefly Algorithm to search the optimal sleep parameter and the maximum utility function.

The remainder of this paper is organized as follows. In Section 2, a VM scheduling strategy with a speed switch and a multi-sleep mode is proposed, and a system model is established. In Section 3, the steady-state distribution of the system model is analyzed. In Section 4, the expressions for some important performance measures in terms of the energy saving level of the system and the average delay of requests are derived. In Section 5, numerical results are provided to evaluate the system performance of the proposed VM scheduling strategy. An intelligent optimization method is presented to optimize the sleep parameter in Section 6. Finally, conclusions are drawn in Section 7.

2. Virtual Machine Scheduling Strategy and System Model

In this section, we first propose a Virtual Machine scheduling strategy for improving the energy efficiency in cloud data centers (CDCs) by using a speed switch and a multi-sleep mode. Then, we develop a continuous-time queueing model with an adaptive service rate and a partial synchronous vacation.

2.1 VM Scheduling Strategy

In conventional CDCs, all the VMs remain open regardless of traffic load. This results in a large amount of energy being wasted, which is referred to as idle energy consumption. Furthermore, inappropriate VM scheduling also generates additional energy consumption, referred to as luxury energy consumption. In order to improve the energy efficiency of CDCs, we propose a VM scheduling strategy with a speed switch and a multi-sleep mode to reduce both the idle energy consumption and the luxury energy consumption.

In the proposed VM scheduling strategy, all the VMs in the CDC are divided into one of two modules, namely, a base-line module or a reserve module. The VMs in the base-line module are always active, and their processing speed can be switched between a low speed and a high speed in accordance with the traffic load. The VMs in the reserve module can be awakened from multiple sleeps.

Based on the stochastic behavior of cloud users, as well as the operational characteristics of sleep timers, the CDC will be converted in the following three cases:

Case I : The VMs in the base-line module operate at a low speed while the VMs in the reserve module are asleep. The level of energy-conservation in the CDC is the most significant in this case.

Case II : The VMs in the base-line module operate at a high speed while the VMs in the reserve module are asleep. The level of energy-conservation in the CDC is relatively obvious in this case.

Case III : The VMs in the base-line module operate at a high speed while the VMs in the reserve module are awake and operate at a high speed. The response performance in the CDC is most ideal in this case.

To avoid frequently switching the processing speed of VMs in the base-line module, we use a dual-threshold, marked as θ_1 ($\theta_1 = 0, 1, 2, \ldots$) and θ_2 ($\theta_2 = 0, 1, 2, \ldots$), to jointly control the VMs processing speed in the base-line module, in which we set $0 < \theta_2 < \theta_1$. When the number of cloud users in the CDC exceeds the threshold θ_1, all the VMs in the base-line module will operate at a high speed. When the number of cloud users in the CDC is less than the threshold θ_2, all the VMs in the base-line module will operate at a low speed. To guarantee the QoS in the CDC even when

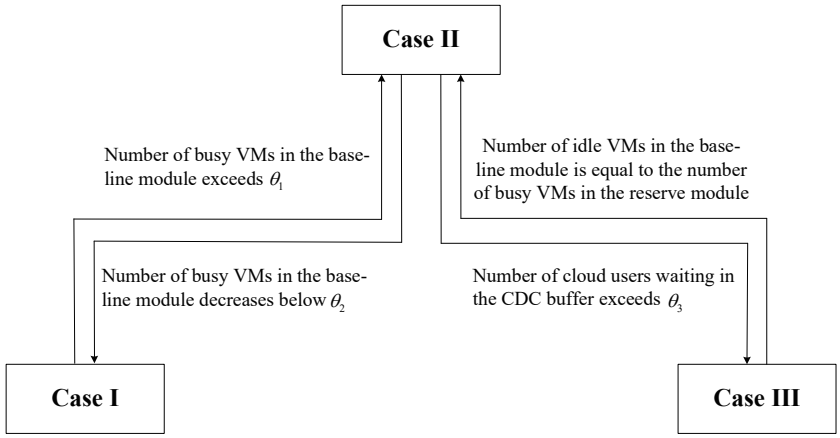


Figure 1 The Transition in the Three CDC Cases

the traffic load is heavy, we use another threshold, called the activation threshold θ_3, to wake up the VMs in the reserve module. If the number of cloud users waiting in the CDC buffer exceeds the threshold θ_3, all the VMs in the reserve module will be awakened and operate at a high speed after the sleep timer expires. Otherwise, the sleep timer will be restarted with a random duration, and all the VMs in the reserve module will go to sleep again.

For convenience of presentation, we denote the number of VMs in the base-line module as n, and the number of VMs in the reserve module as m. To avoid the appearance that all the VMs in the reserve module are awake while the VMs in the base-line module operate at a low speed, we set $(n - \theta_2) \geqslant m$. To ensure all the cloud users in the CDC buffer can be served once the VMs in the reserve module are awakened, we set $0 < \theta_3 < m$.

According to the proposed VM scheduling strategy, the transition among the three CDC cases is illustrated in Fig. 1.

In **Case I**, each cloud user is served immediately on a VM available in the base-line module at a low speed. However, with the arrivals of the cloud users, more VMs in the base-line module will be occupied. Here, we call the VMs being occupied by cloud users as busy VMs. When the number of busy VMs in the base-line module exceeds the threshold θ_1, all the VMs in the base-line module will be switched to a high speed, i.e., the CDC will be converted to the **Case II** state. The cloud users that have not received service yet will be served continuously on the same VM, but at a high speed. In this CDC case, there are no cloud users waiting in the CDC buffer. Therefore, when the sleep timer expires, this sleep timer will be restarted with a random duration, and all the VMs in the reserve module will go to sleep again.

In **Case II**, if there are idle VMs in the base-line module, the incoming cloud users will be served immediately in the base-line module at a high speed. Otherwise, the cloud users have to wait in the CDC buffer. On the one hand, with the arrivals of cloud users, more cloud users will queue in the CDC buffer. When the sleep timer expires, if the number of cloud users waiting in the CDC buffer exceeds the activation threshold θ_3, all the VMs in the reserve module will be awakened and operate at a high speed directly, i.e., the CDC will be converted to the **Case III** state. Then all the cloud users in the CDC buffer will be served immediately in the reserve module at a high speed. Otherwise, the CDC will remain in the **Case II** state. As service continues, cloud users that have finished being served depart, so fewer VMs in

the base-line module will be busy. When the number of busy VMs in the base-line module decreases below the threshold θ_2, all the VMs in the base-line module will be switched to a low speed, i.e., the CDC will be converted to the **Case I** state. The cloud users queueing in the CDC buffer will be served continuously on the same VM, but at a low speed.

In **Case III**, if there are idle VMs in either the base-line module or the reserve module, the incoming cloud users will be served immediately at a high speed. Otherwise, the cloud users will queue in the CDC buffer. However, as cloud users that have finished being served depart, fewer VMs in both the base-line module and the reserve module will be busy. When the number of idle VMs in the base-line module is equal to the number of busy VMs in the reserve module, the cloud users queueing in the CDC buffer will be migrated to the idle VMs in the base-line module and served at a high speed. Then the sleep timer will be restarted with a random duration, and all the VMs in the reserve module will go to sleep again, i.e., the CDC will be converted to the **Case II** state.

2.2 System Model

In this subsection, we establish a continuous-time queueing model with an adaptive service rate and a partial synchronous vacation to capture the related performance measures of the CDC using the proposed VM scheduling strategy. In this queueing model, the requests from cloud users are regarded as customers. Each VM is regarded as an independent server that can only serve one request at a time. A sleep is abstracted as a vacation. The system buffer is supposed to be infinite.

We assume that the arrival intervals of requests follow an exponential distribution with parameter λ ($\lambda > 0$). We assume that the service time of a request when the system is in the **Case I** state follows an exponential distribution with parameter μ_l ($\mu_l > 0$). The service time of a request when the system is in either the **Case II** state or the **Case III** state follows an exponential distribution with parameter μ_h ($\mu_h > \mu_l$). Furthermore, we assume that the energy consumption of a VM during the sleep state is J_v ($J_v > 0$), the energy consumption of an idle VM is J_o ($J_o > J_v$), the energy consumption of a busy VM operating at the low speed and the high speed are J_l and J_h ($J_h > J_l$), respectively. And, the additional energy consumption of a VM switching to a high speed from a low speed is J_a ($J_a > 0$), and that of a VM being woken up from sleep state is J_b ($J_b > 0$). In addition, we assume that the time length of a sleep timer follows an exponential distribution with parameter δ ($\delta > 0$). Here, we refer to the parameter δ as the sleep parameter.

Let random variable $N(t) = i,\ i \in \{0, 1, 2, \ldots\}$ be the total number of requests in the system at instant t, which is called the system level. Let random variable $C(t) = j,\ j \in \{1, 2, 3\}$ be the system case at instant t. $j = 1, 2, 3$ represents the system being in the state of **Case I**, **Case II** and **Case III**, respectively. $\{N(t), C(t), t \geqslant 0\}$ constitutes a two dimensional continuous-time Markov chain (CTMC) (Tian and Zhang 2006). The state-space $\mathbf{\Omega}$ of the CTMC is given as follows:

$$\mathbf{\Omega} = \{(i, j) \mid i \in \{0, 1, 2, \ldots\},\ j \in \{1, 2, 3\}\} \quad (1)$$

For the two dimensional CTMC, we define $\pi_{i,j}$ as the steady-state probability when the system level is i and the system case is j. $\pi_{i,j}$ is given as follows:

$$\pi_{i,j} = \lim_{t\to\infty} P\{N(t) = i, C(t) = j\},\quad i \in \{0, 1, 2, \ldots\},\ j \in \{1, 2, 3\} \quad (2)$$

We define $\boldsymbol{\pi}_i$ as the steady-state probability vector when the system level is i. $\boldsymbol{\pi}_i$ can be given as follows:

$$\boldsymbol{\pi}_i = \begin{cases} (\pi_{i1}, \pi_{i2}),\ i \in \{0, 1, 2, \ldots, n\} \\ (\pi_{i2}, \pi_{i3}),\ i \in \{n+1, n+2, n+3, \ldots\} \end{cases} \quad (3)$$

Table 1 The Relation between the System Levels and the System Cases

System levels	Initial system cases before one step transition	Possible system cases after one step transition
$[0, \theta_2 - 1]$	**Case I**	**Case I**
θ_2	**Case I**	**Case I**
	Case II	**Case I** or **Case II**
$[\theta_2 + 1, \theta_1 - 1]$	**Case I**	**Case I**
	Case II	**Case II**
θ_1	**Case I**	**Case I** or **Case II**
	Case II	**Case II**
$[\theta_1 + 1, n]$	**Case II**	**Case II**
$n + 1$	**Case II**	**Case II**
	Case III	**Case II** or **Case III**
$[n + 2, n + \theta_3]$	**Case II**	**Case II**
	Case III	**Case III**
$[n + \theta_3 + 1, \infty)$	**Case II**	**Case II** or **Case III**
	Case III	**Case III**

The steady-state probability distribution $\mathbf{\Pi}$ of the CTMC is composed of $\boldsymbol{\pi}_i$ $(i \geq 0)$. $\mathbf{\Pi}$ is given as follows:

$$\mathbf{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots) \tag{4}$$

3. Model Analysis

In this section, we first discuss the transition rate matrix of the two dimensional CTMC. Then, we derive the steady-state probability distribution of the system model.

3.1 Transition Rate Matrix

According to the proposed VM scheduling strategy, the system case is related to the system level. The relation between the system level and the system case is illustrated in Table 1.

Based on Table 1, we illustrate the state transition of the system model in Fig. 2.

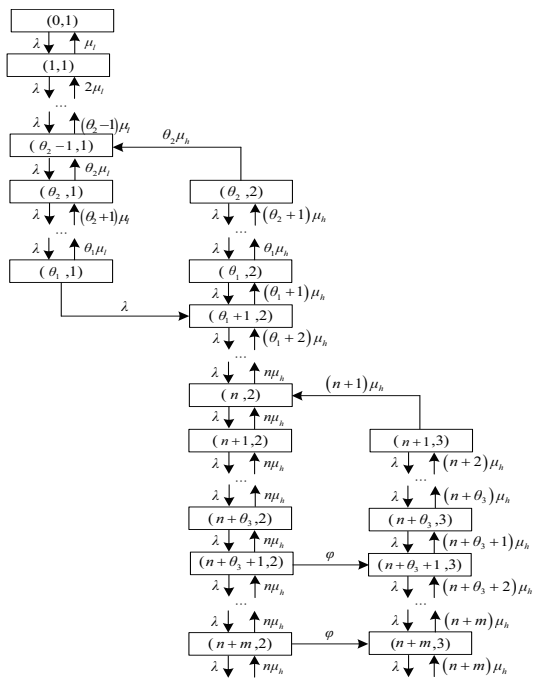


Figure 2 The State Transition of the System Model

Let $\boldsymbol{Q}$ be the one-step state transition rate matrix of the two-dimensional CTMC $\{(N(t), C(t)), t \geq 0\}$. As shown in Table 1, each system level has at most two corresponding system cases, so we separate $\boldsymbol{Q}$ into sub-matrices of 2×2 structure. Let $\boldsymbol{Q}_{u,v}$ be the one-step state transition rate sub-matrix for the system level changing from u $(u = 0, 1, 2, \ldots)$ to v $(v = 0, 1, 2, \ldots)$. For clarity, $\boldsymbol{Q}_{u,u-1}$, $\boldsymbol{Q}_{u,u}$ and $\boldsymbol{Q}_{u,u+1}$ are abbreviated as $\boldsymbol{B}_u$, $\boldsymbol{A}_u$ and $\boldsymbol{C}_u$, respectively.

For the initial system level $u = 0$, $\boldsymbol{A}_0$ indicates the state transition rates from $(0, 1)$ to $(0, 1)$, $(0, 1)$ to $(0, 2)$, $(0, 2)$ to $(0, 1)$ and $(0, 2)$ to $(0, 2)$. $\boldsymbol{C}_0$ indicates the state transition rates

from $(0,1)$ to $(1,1)$, $(0,1)$ to $(1,2)$, $(0,2)$ to $(1,1)$ and $(0,2)$ to $(1,2)$.

For the initial system level $1 \le u \le n-1$, $\boldsymbol{B}_u$ indicates the state transition rates from $(u,1)$ to $(u-1,1)$, $(u,1)$ to $(u-1,2)$, $(u,2)$ to $(u-1,1)$ and $(u,2)$ to $(u-1,2)$. $\boldsymbol{A}_u$ indicates the state transition rates from $(u,1)$ to $(u,1)$, $(u,1)$ to $(u,2)$, $(u,2)$ to $(u,1)$ and $(u,2)$ to $(u,2)$. $\boldsymbol{C}_u$ indicates the state transition rates from $(u,1)$ to $(u+1,1)$, $(u,1)$ to $(u+1,2)$, $(u,2)$ to $(u+1,1)$ and $(u,2)$ to $(u+1,2)$.

For the initial system level $u = n$, $\boldsymbol{B}_n$ indicates the state transition rates from $(n,1)$ to $(n-1,1)$, $(n,1)$ to $(n-1,2)$, $(n,2)$ to $(n-1,1)$ and $(n,2)$ to $(n-1,2)$. $\boldsymbol{A}_n$ indicates the state transition rates from $(n,1)$ to $(n,1)$, $(n,1)$ to $(n,2)$, $(n,2)$ to $(n,1)$ and $(n,2)$ to $(n,2)$. $\boldsymbol{C}_n$ indicates the state transition rates from $(n,1)$ to $(n+1,2)$, $(n,1)$ to $(n+1,3)$, $(n,2)$ to $(n+1,2)$ and $(n,2)$ to $(n+1,3)$.

For the initial system level $u = n+1$, $\boldsymbol{B}_{n+1}$ indicates the state transition rates from $(n+1,2)$ to $(n,1)$, $(n+1,2)$ to $(n,2)$, $(n+1,3)$ to $(n,1)$ and $(n+1,3)$ to $(n,2)$. $\boldsymbol{A}_{n+1}$ indicates the state transition rates from $(n+1,2)$ to $(n+1,2)$, $(n+1,2)$ to $(n+1,3)$, $(n+1,3)$ to $(n+1,2)$ and $(n+1,3)$ to $(n+1,3)$. $\boldsymbol{C}_{n+1}$ indicates the state transition rates from $(n+1,2)$ to $(n+2,2)$, $(n+1,2)$ to $(n+2,3)$, $(n+1,3)$ to $(n+2,2)$ and $(n+1,3)$ to $(n+1,3)$.

For the initial system level $n+2 \le u < \infty$, $\boldsymbol{B}_u$ indicates the state transition rates from $(u,2)$ to $(u-1,2)$, $(u,2)$ to $(u-1,3)$, $(u,3)$ to $(u-1,2)$ and $(u,3)$ to $(u-1,3)$. $\boldsymbol{A}_u$ indicates the state transition rates from $(u,2)$ to $(u,2)$, $(u,2)$ to $(u,3)$, $(u,3)$ to $(u,2)$ and $(u,3)$ to $(u,3)$. $\boldsymbol{C}_u$ indicates the state transition rates from $(u,2)$ to $(u+1,2)$, $(u,2)$ to $(u+1,3)$, $(u,3)$ to $(u+1,2)$ and $(u,3)$ to $(u+1,3)$.

All the sub-matrices in $\boldsymbol{Q}$ can be addressed according to the state transition rates shown in Fig. 2. We find that starting from the system level $(n+m+1)$, all the sub-matrices in $\boldsymbol{Q}$ are repeated forever. Then, $\boldsymbol{Q}$ is given as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C}_1 & & & & & \\ \ddots & \ddots & \ddots & & & & & \\ & \boldsymbol{B}_{n-1} & \boldsymbol{A}_{n-1} & \boldsymbol{C}_{n-1} & & & & \\ & & \boldsymbol{B}_n & \boldsymbol{A}_n & \boldsymbol{C}_n & & & \\ & & & \boldsymbol{B}_{n+1} & \boldsymbol{A}_{n+1} & \boldsymbol{C}_{n+1} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \boldsymbol{B}_{n+m} & \boldsymbol{A}_{n+m} & \boldsymbol{C}_{n+m} \\ & & & & & & \ddots & \ddots & \ddots \end{pmatrix} \tag{5}$$

The block-tridiagonal structure of $\boldsymbol{Q}$ shows that the state transitions occur only between adjacent system levels. Hence, the two dimensional CTMC $\{N(t), C(t), t \ge 0\}$ can be seen as a type of Quasi Birth-and-Death (QBD) process (Banik et al. 2007, Zhang and Hou 2011).

3.2 Steady-State Probability Distribution

For the CTMC $\{N(t), C(t), t \ge 0\}$ with the one step-state transition rate matrix $\boldsymbol{Q}$, the necessary and sufficient conditions for positive recurrence are that the matrix quadratic equation:

$$\boldsymbol{R}^2\boldsymbol{B}_{n+m} + \boldsymbol{R}\boldsymbol{A}_{n+m} + \boldsymbol{C}_{n+m} = \boldsymbol{0} \tag{6}$$

has a minimal non-negative solution $\boldsymbol{R}$ and that the spectral radius $SP(\boldsymbol{R}) < 1$, where $\boldsymbol{0}$ is a zero matrix of order 2×2.

We assume the rate matrix $\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}$, then substitute $\boldsymbol{R}$, $\boldsymbol{B}_{n+m}$, $\boldsymbol{A}_{n+m}$, and $\boldsymbol{C}_{n+m}$ into Eq. (6), so we have

$$\begin{pmatrix} n\mu_h r_{11}^2 & (n+m)\mu_h(r_{11}+r_{22})r_{12} \\ 0 & (n+m)\mu_h r_{22}^2 \end{pmatrix} + \begin{pmatrix} -(\lambda + n\mu_h + \delta)r_{11} & r_{11}\delta - (\lambda + (n+m)\mu_h)r_{12} \\ 0 & -(\lambda + (n+m)\mu_h)r_{22} \end{pmatrix} + \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{7}$$

By solving Eq. (7), we can derive r_{11}, r_{22} and

r_{12} as follows:

$$\begin{cases} r_{11} = \dfrac{(\lambda + n\mu_h + \delta) - \sqrt{(\lambda + n\mu_h + \delta)^2 - 4n\lambda\mu_h}}{2n\mu_h} \\ r_{22} = \dfrac{\lambda}{(n+m)\mu_h} \\ r_{12} = \dfrac{r_{11}\delta}{\lambda + (n+m)(1 - r_{11} - r_{22})\mu_h} \end{cases} \tag{8}$$

The rate matrix $\boldsymbol{R}$ has been given in closed-form. Note that $SP(\boldsymbol{R}) = max\{r_{11}, r_{22}\}$ and r_{11} can be proved mathematically to be less than 1. Therefore, the necessary and sufficient condition for positive recurrence of the CTMC $\{N(t), C(t), t \geq 0\}$ is equivalent to $r_{22} < 1$, that is, $\lambda < (n+m)\mu_h$.

With the rate matrix $\boldsymbol{R}$ obtained, we construct a square matrix $B[\boldsymbol{R}]$ as follows:

$$B[\boldsymbol{R}] = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C}_1 & & & & \\ \ddots & \ddots & \ddots & & & & \\ & \boldsymbol{B}_{n-1} & \boldsymbol{A}_{n-1} & \boldsymbol{C}_{n-1} & & & \\ & & \boldsymbol{B}_n & \boldsymbol{A}_n & \boldsymbol{C}_n & & \\ & & & \boldsymbol{B}_{n+1} & \boldsymbol{A}_{n+1} & \boldsymbol{C}_{n+1} & \\ & & & & \ddots & \ddots & \ddots \\ & & & & & \boldsymbol{B}_{n+m}\boldsymbol{R}\boldsymbol{B}_{n+m} + \boldsymbol{A}_{n+m} \end{pmatrix} \tag{9}$$

By using the matrix-geometric method (Latouche and Ramaswami 2000, Neuts 1981), we can give an equation set as follows:

$$\begin{cases} (\pi_0, \pi_1, \ldots, \pi_{n+m})B[\boldsymbol{R}] = \underbrace{(0, 0, \ldots, 0)}_{2(n+m+1)} \\ (\pi_0, \pi_1, \ldots, \pi_{n+m-1})\boldsymbol{e} + \pi_{n+m}(\boldsymbol{I} - \boldsymbol{R})^{-1}\boldsymbol{e}_1 = 1 \end{cases} \tag{10}$$

where $\boldsymbol{e}$ is a $2(n+m) \times 1$ vector with ones, and $\boldsymbol{e}_1$ is a 2×1 vector with ones.

We further construct an augmented matrix as follows:

$$(\pi_0, \pi_1, \ldots, \pi_{n+m}) \left(B[\boldsymbol{R}] \;\vdots\; \begin{matrix} \boldsymbol{e} \\ (\boldsymbol{I} - \boldsymbol{R})^{-1}\boldsymbol{e}_1 \end{matrix} \right) = \underbrace{(0, 0, \ldots, 0, 1)}_{2(n+m+1)} \tag{11}$$

Applying the Gauss-Seidel method (Greenbaum 1997) to solve Eq. (11), we can obtain $\pi_0, \pi_1, \ldots, \pi_{n+m}$.

From the structure of the transition rate matrix $\boldsymbol{Q}$, we know $\boldsymbol{\pi}_i$ $(i = n+m+1, n+m+2, n+m+3, \ldots)$ satisfies the matrix geometric solution form as follows:

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_{n+m}\boldsymbol{R}^{i-(n+m)}, \; i \geq (n+m) \tag{12}$$

Substituting $\boldsymbol{\pi}_{n+m}$ obtained in Eq. (11) into Eq. (12), we can obtain $\boldsymbol{\pi}_i$ $(i = n+m+1, n+m+2, n+m+3, \ldots)$. Then the steady-state probability distribution $\boldsymbol{\Pi} = (\pi_0, \pi_1, \pi_2, \ldots)$ of the system can be given mathematically.

4. Performance Measures

The energy saving level is defined as the ratio of the difference between the energy consumption of the conventional CDC and that of the CDC with the proposed VM scheduling strategy over the energy consumption of the conventional CDC.

The energy consumption C of the CDC with the proposed VM scheduling strategy is given as follows:

$$C = C_1 + C_2 + C_3 + C_4 \tag{13}$$

where C_1, C_2 and C_3 are the average energy consumption when the system is in the states of **Case I**, **Case II** and **Case III**, respectively, C_4 is the additional energy consumption caused by speed switching and VM activation.

C_1, C_2, C_3 and C_4 are given as follows:

$$C_1 = \sum_{i=0}^{\theta_1} \pi_{i,1}(iJ_l + (n-i)J_o + mJ_v),$$

$$C_2 = \sum_{i=\theta_2}^{n} \pi_{i,2}(iJ_h + (n-i)J_o + mJ_v) + \sum_{i=n+1}^{\infty} \pi_{i,2}(nJ_h + mJ_v),$$

$$C_3 = \sum_{i=n+1}^{n+m} \pi_{i,3}(iJ_h + (n+m-i)J_o) + \sum_{i=n+m+1}^{\infty} \pi_{i,3}((n+m)J_h),$$

Table 2 Experimental Parameters

Parameters	Value
Total number $(n + m)$ of VMs in the system	50
Service rate μ_l when VM operates at the low speed	0.01 ms^{-1}
Service rate μ_h when VM operates at the high speed	0.02 ms^{-1}
Dual-threshold θ_1, θ_2	20, 10
Activation threshold θ_3	15
Energy consumption level J_v of a sleeping VM	0.2 mJ
Energy consumption level J_o of an idle VM	0.4 mJ
Energy consumption level J_l of a busy VM operating at the low speed	0.45 mJ
Energy consumption level J_h of a busy VM operating at the high speed	0.5 mJ
Additional energy consumption level J_a of a VM caused by speed switch	1.0 mJ
Additional energy consumption level J_b of a VM caused by activation	2.0 mJ

$$C_4 = \sum_{i=n+\theta_3+1}^{\infty} \pi_{i,2}\delta m J_b + \pi_{\theta_1,1}\lambda n J_a.$$

In the conventional CDC, the energy consumption C' is given as follows:

$$\begin{aligned} C' &= (n+m)J_h\left(\frac{\lambda}{(n+m)\mu_h}\right) \\ &\quad +(n+m)J_o\left(1-\frac{\lambda}{(n+m)\mu_h}\right) \\ &= \frac{\lambda J_h}{\mu_h} + J_o\left(n+m-\frac{\lambda}{\mu_h}\right) \end{aligned} \tag{14}$$

Combining Eqs. (13) and (14), the energy saving level S of the CDC with the proposed VM scheduling strategy is given as follows:

$$S = \frac{C' - C}{C'} \tag{15}$$

We define the delay of a request as the duration from the instant a request arrives at the system to the instant this request is served.

Based on the steady-state probability distribution obtained in Subsection 3.2, the average number $E[L]$ of requests waiting in the system buffer is given as follows:

$$E[L] = \sum_{i=n+1}^{\infty}(i-n)\pi_{i,2} + \sum_{i=n+m+1}^{\infty}(i-(n+m))\,\pi_{i,3} \tag{16}$$

Applying Little's law, the average delay $E[W]$ of requests is given as follows:

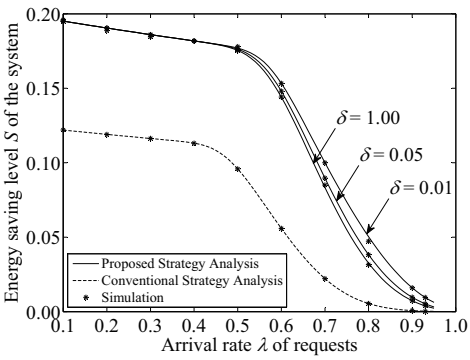
$$\begin{aligned} E[W] &= \frac{E[L]}{\lambda} \\ &= \frac{1}{\lambda}\left(\sum_{i=n+1}^{\infty}(i-n)\pi_{i,2} + \sum_{i=n+m+1}^{\infty}(i-(n+m))\,\pi_{i,3}\right) \end{aligned} \tag{17}$$

5. Numerical Experiments

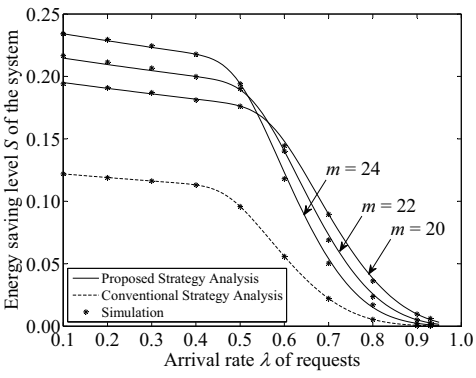
In order to evaluate the response performance and the energy saving level of the CDC with the proposed VM scheduling strategy, we provide numerical experiments with analysis and simulation. The analysis results are obtained based on Eq. (10) using Matlab 2011a. The simulation results are obtained by averaging over 10 independent runs using MyEclipse 2014. With our proposed strategy, all the system parameters satisfy $0 < \theta_2 < \theta_1$, $0 < \theta_3 < m \leqslant n - \theta_2$, $0 < J_v < J_o < J_l < J_h$, $0 < J_a < J_b$ and $0 < \lambda < (n+m)\mu_h$. The parameters set in the numerical experiments are shown in Table 2.

To elucidate the better energy saving effect of the proposed VM scheduling strategy, we carry out a numerical comparison between the proposed VM scheduling strategy and the conventional DPM strategy. In conventional DPM strategy, all the VMs are open all the time, but their processing speed can be switched between a low speed and a high speed according to the traffic load of the system at that time.

By setting the number of VMs in the reserve module $m = 20$ as an example, we examine the



(a) $m = 20$



(b) $\delta = 0.05$

Figure 3 Energy Saving Level S of the System

influence of the arrival rate λ of requests on the energy saving level S of the system for different sleep parameters δ in Fig. 3 (a). By setting the sleep parameter $\delta = 0.05$ as an example, we examine the influence of the arrival rate λ of requests on the energy saving level S of the system for different numbers m of VMs in the reserve module in Fig. 3 (b). In Fig. 3, the solid line represents the analysis results for the energy saving level with the proposed VM scheduling strategy, and the dotted line represents the analysis results for the energy saving level when using the conventional DPM strategy.

In Fig. 3, we observe that for the same sleep parameter δ and the same number m of VMs in the reserve module, the energy saving level S of the system will initially decrease gradually then decrease sharply as the arrival rate λ of requests increases. When λ is smaller (such as $\lambda < 0.5$ for $\delta = 0.05$ and $m = 20$), as λ increases, it becomes more possible that the number of requests in the system will exceed the threshold θ_1. That is, all the VMs in the base-line module will be switched to the high speed from the low speed. Since the energy consumption of a VM operating at the high speed is greater than that operating at the low speed, the energy consumption will increase, and the energy saving level will decrease as the arrival rate of requests increases. When λ is larger (such as $\lambda > 0.5$ for $\delta = 0.05$ and $m = 20$), all the VMs in the base-line module will be busy, and the incoming requests will wait in the system buffer. As λ increases, the number of requests waiting in the system buffer is more likely to exceed the activation threshold θ_3, so the VMs in the reserve module will be awakened after the sleep timer expires. Since the energy consumption of a VM operating at the high speed is greater than that of a VM being asleep, the energy saving due to the sleep mode is greater than that due to switching to the low speed from the high speed, the energy saving level will decrease as the arrival rate of requests increases.

In our proposed strategy, the energy consumption from the highest to the lowest in sequence are: high speed mode, low speed mode and sleep mode. We note that, for a VM, the energy consumption difference between sleep mode and high speed mode is greater than that between low speed mode and high speed mode. Moreover, the energy consumption of a VM in the reserve module switching from sleep mode to high speed mode is far greater than that of a VM in the base-line module switching from low speed mode to high speed mode. Therefore, the downtrend of the energy savings level at the the preceding stage is weaker than that at the following one, and the energy savings level drops abruptly from $\lambda = 0.5$.

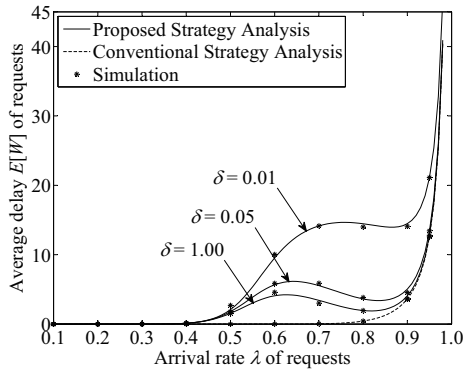
From Fig. 3 (a), we notice that for the same

arrival rate λ of requests, the energy saving level S of the system decreases as the sleep parameter δ increases. The larger the value of δ is, the more likely the VMs in the reserve module will be awake. Thus, the energy consumption of the system will increase, and the energy saving level will decrease.

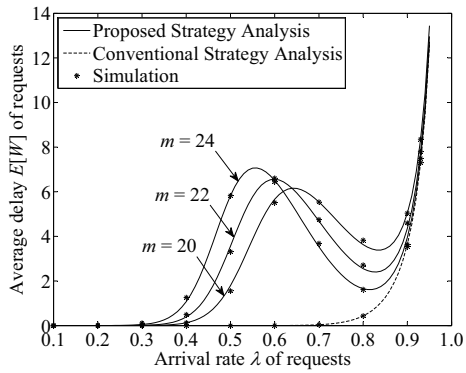
From Fig. 3 (b), we notice that for a smaller arrival rate λ of requests (such as $\lambda < 0.5$ for $\delta = 0.05$), the energy saving level S of the system increases as the number m of VMs in the reserve module increases. When λ is smaller, no matter how small the value of the number n of VMs in the base-line module is, all the VMs in the reserve module will be more likely to go to sleep again after the sleep timer expires. Thus, as the value of m increases, the energy saving level of the system will increase.

On the other hand, for a larger arrival rate λ of requests (such as $\lambda > 0.55$ for $\delta = 0.05$), the energy saving level S of the system will decrease as the number m of VMs in the reserve module increases. When λ is larger, as the number n of VMs in the base-line module decreases, the number of requests waiting in the system buffer will increase, and more likely it is that all the VMs in the reserve module will awaken. Thus, as the value of m increases, the energy saving level of the system will decrease.

By setting the number of VMs in the reserve module $m = 20$ as an example, we examine the influence of the arrival rate λ of requests on the average delay $E[W]$ of requests for different sleep parameters δ in Fig. 4 (a). By setting the sleep parameter $\delta = 0.05$ as an example, we examine the influence of the arrival rate λ of requests on the average delay $E[W]$ of requests for different numbers m of VMs in the reserve module in Fig. 4 (b). In Fig. 4, the solid line represents the analysis results of the average delay with the proposed VM scheduling strategy, and the dotted line represents the analysis results of average delay when using the conventional DPM strategy.



(a) $m = 20$



(b) $\delta = 0.05$

Figure 4 Average Delay $E[W]$ of Requests

From Fig. 4, we observe that for the same sleep parameter δ and the same number m of VMs in the reserve module, the average delay $E[W]$ of requests initially increases from 0, then decreases slightly before finally increasing sharply as the arrival rate λ of requests increases. When λ is relatively small (such as $0 < \lambda < 0.65$ for $\delta = 0.05$ and $m = 20$), the VMs in the reserve module are more likely to be asleep, only the VMs in the base-line module will be available. If all the VMs in the base-line module are busy, the incoming requests have to wait in the system buffer. Thus, the average delay of requests will increase gradually. When λ is moderate (such as $0.65 < \lambda < 0.85$ for $\delta = 0.05$ and $m = 20$), the VMs in the reserve module are more likely to be awakened after the sleep timer expires, so all the requests waiting in the system buffer can be served in the reserve module.

Furthermore, the larger the value of λ is, the earlier the VMs in the reserve module will be awakened. Thus, the average delay of requests will decrease. When λ further increases (such as $\lambda > 0.85$ for $\delta = 0.05$ and $m = 20$), the number of requests waiting in the system buffer increases rapidly, even though all the VMs in both the base-line module and the reserve module are busy. In this case, the system tends to be unsteady, so the average delay of requests will increase sharply.

From Fig. 4 (a), we notice that for the same arrival rate λ of requests, the average delay $E[W]$ of requests will decrease as the sleep parameter δ increases. A larger δ will shorten the average sleep time of VMs in the reserve module. The requests waiting in the system buffer can be served earlier in the reserve module. That is, the requests will be waiting a shorter time in the system buffer, so the average delay of requests will decrease.

From Fig. 4 (b), we notice that for a smaller arrival rate λ of requests (such as $\lambda < 0.6$), the average delay $E[W]$ of requests will increase as the number m of VMs in the reserved module increases. Note that the total number $(n+m)$ of VMs in the system is fixed. Hence, the increase in the number m of VMs in the reserve module means a decrease in the number n of VMs in the base-line module. When λ is smaller, there are fewer requests in the system, and the number of requests waiting in the system buffer is more likely to be below the activation threshold θ_3 no matter how small the value of n is. That is, the requests can only be served in the base-line module, and the average waiting time of requests will increase as the value of n decreases. Thus, in this situation, the average delay of requests will increase as the value of m increases.

On the other hand, for a larger arrival rate λ of requests (such as $\lambda > 0.65$), the average delay $E[W]$ of requests will decrease as the number m of VMs in the reserve module increases. When λ is large, there are many requests in the system, and the number of requests waiting in the system buffer is more likely to be higher than the activation threshold θ_3 no matter how great the value of n is. Furthermore, the larger the value of m is, the smaller the value of n is, and the earlier the VMs in the reserve module be awakened synchronously. That is, not only will the VMs in the base-line module be involved in the service, but also the VMs in the reserve module will be involved in the service. Thus, the average delay of requests will decrease as the value of m increases.

Concluding with the experiment results shown in Figs. 3-4, we find that, compared with the conventional CDCs, the energy saving level of CDCs using the proposed VM scheduling strategy increases significantly, while the average delay of requests increases slightly. What's more, the larger the value of the sleep parameter is, the greater the difference is between the performance measures of the conventional CDCs and those of the CDCs using the proposed strategy. Therefore, a tradeoff among the average delay of requests and the energy saving level of the system should be considered when setting the sleep parameter in our proposed VM scheduling strategy.

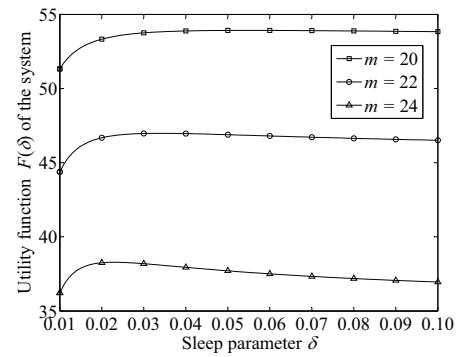
6. Optimization of Sleep Parameter

By trading off different performance measures, we establish a utility function (Chen et al. 2014) F_δ for the system as follows:

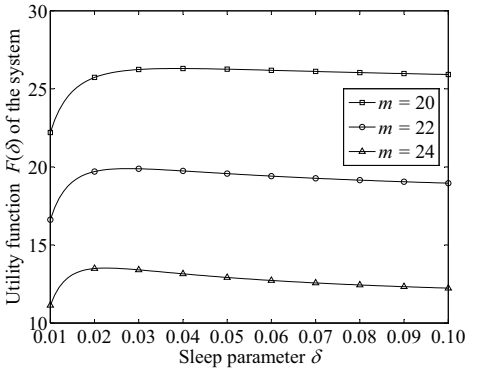
$$F(\delta) = f_s S - f_w E[W] \tag{18}$$

where f_s and f_w are the factors impacting on the system utility, that being the energy saving level of the system and the average delay of requests, respectively on the system utility. S and $E[W]$ have been obtained in Eq. (15) and Eq. (17).

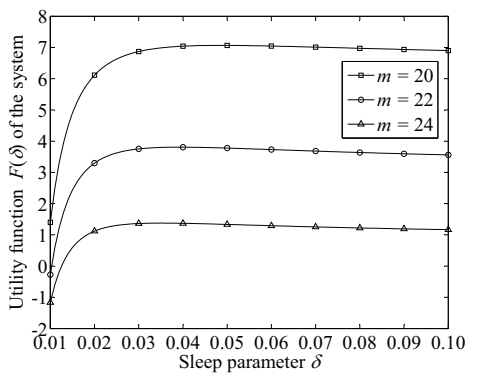
Using the parameters given in Table 2, and setting $f_s = 500$ and $f_w = 1$ as an example in Fig. 5, we illustrate how the utility function



(a) $\lambda = 0.65$



(b) $\lambda = 0.75$



(c) $\lambda = 0.85$

Figure 5 Change Trend for the Utility Function $F(\delta)$ of the System

$F(\delta)$ of the system changes along with the sleep parameter δ for different arrival rates λ of requests and numbers m of VMs in the reserve module.

As shown in Fig. 5, for all the combinations of arrival rates λ of requests and numbers m of VMs in the reserve module, the utility function $F(\delta)$ of the system firstly increases and then decreases as the sleep parameter δ increases. We recall that both of the average delay of requests and the energy saving level of the system will decrease as the sleep parameter increases. When the sleep parameter is smaller, the downward trend of the average delay of requests is bigger than that of the energy saving level of the system, and the average delay of requests is a dominant impacting factor. Hence, there is an increasing stage in Fig. 5. When the sleep parameter is greater, the downward trend of the energy saving level of the system is bigger than that of the average delay of requests, and the energy saving level of the system is a dominant impacting factor. Hence, there is a decreasing stage in Fig. 5. Therefore, there is a maximum utility function $F(\delta^*)$ when the sleep parameter is set to the optimal value δ^*.

We note that mathematical expression of the energy saving level S of the system and the average delay $E[W]$ of requests are difficult to express in a closed-form. The monotonicity of the system utility function is uncertain. In order to obtain the exact value for the optimal sleep parameter δ^* with maximum system utility function $F(\delta^*)$, we develop an improved Firefly intelligent searching algorithm.

The Firefly Algorithm (Yu et al. 2015, Gandomi et al. 2011) is a population-based algorithm to find the global optimal value of objective functions by imitating the collective behavior of fireflies. In the Firefly Algorithm, all fireflies are randomly distributed in the search space, then the less bright ones will move towards the brighter ones because the attractiveness of fireflies is proportional to their brightness. We note that the initial positions of fireflies have great influence on the searching ability of intelligent searching algorithms. In the improved Firefly Algorithm, we use chaotic equations to initialize the positions of fireflies more diversely.The main steps of the improved Firefly Algorithm are given in Table 3.

Table 3 Improved Firefly Algorithm to Obtain δ^* and $F(\delta^*)$

Step 1: Initialize the number N of fireflies, the iteration number L of each firefly's position, the number K of best solutions, the chaotic factor ξ, the light absorption coefficient γ, the maximum attractiveness β_0, the step size α. Set the sequence number of the best solutions as $x = 1$.

Step 2: Initialize the position of each firefly in the interval $[0.01, 0.10]$ using chaotic equations. δ_h is the position of the hth firefly, $h \in \{1, 2, \ldots, N\}$.

$\delta_1 = 0.09 \times rand + 0.01$

% $rand$ is the uniform random variable selected in the interval $(0, 1)$. %

for $h = 2 : N$

$\delta_h = \xi \times (\delta_{h-1} - 0.01) \times (1 - (\delta_{h-1} - 0.01)/0.90) + 0.01$

endfor

Step 3: Calculate the self-brightness $I_0(h)$ of each firefly, $h \in \{1, \ldots, N\}$:

$I_0(h) = F(\delta_h) = f_s S - f_w E[W]$

Step 4: For each firefly, update the position and the self brightness:

Initialize the current iteration as $t = 1$.

while $t \le L$

for $i = 1 : N$

for $j = 1 : N$

$I(j, i) = I_0(j) \times e^{-\gamma \times |\delta_j - \delta_i|}$

% Calculate the relative brightness $I(j, i)$ for the jth firefly to the ith firefly %

if $I(j, i) > I_0(i)$

$\beta(j, i) = \beta_0 \times e^{-\gamma \times |\delta_j - \delta_i|}$

% Calculate the attractiveness $\beta(j, i)$ of the jth firefly to the ith firefly %

$\Delta_i = \beta(j, i) \times (\delta_j - \delta_i) + \alpha \times (rand - 1/2)$

% Calculate the move distance Δ_i of the ith firefly to the jth firefly %

$\delta_i = \delta_i + \Delta_i$

$I_0(i) = F(\delta_i) = f_s S - f_w E[W]$

% Update the position δ_i and the self brightness $I_0(i)$ of the ith firefly %

$t = t + 1$

endif

endfor

endfor

endwhile

Step 5: Select the maximum self-brightness and the best position of N fireflies as one of the best solutions:

$\delta[x] = \underset{h \in \{1, \ldots, N\}}{argmax} \{I_0(h)\}$

$x = x + 1$

% $\delta[x]$ is an array that help to record K best solutions.%

Step 6: **if** $x \le K$

go to **Step** 4.

else

$\delta^* = average(\delta[x])$

$F(\delta^*) = f_s S - f_w E[W]$

endif

Step 7: Output δ^* and $F(\delta^*)$

Table 4 Optimal Sleep Parameter δ^* of the Proposed Strategy

λ	m	δ^*	$F(\delta^*)$
0.65	20	0.0550	53.9210
	22	0.0340	46.9805
	24	0.0230	38.2877
0.75	20	0.0390	26.3022
	22	0.0280	19.8894
	24	0.0230	13.5192
0.85	20	0.0490	7.0671
	22	0.0390	3.8076
	24	0.0350	1.3757

In the improved Firefly algorithm given in Table 3, we set $N = 10^4$, $L = 10^{12}$, $K = 100$, $\gamma = 1$, $\beta_0 = 0.01$, $\alpha = 0.20$. Then, we obtain the optimal sleep parameter δ^* and the corresponding maximum utility function $F(\delta^*)$ of the system in Table 4. In Table 4, the estimates of δ^* and $F(\delta^*)$ are accurate to four decimal places.

From Table 4, we observe that for the same arrival rate λ of requests, both of the optimal sleep parameter δ^* and corresponding maximum utility function $F(\delta^*)$ will decrease as the number m of VMs in the reserve module increases. For the same number m of VMs in the reserve module, as the arrival rate λ of requests increases, the optimal sleep parameter δ^* firstly decreases and then increases, while the corresponding maximum utility function $F(\delta^*)$ continuously decreases.

7. Conclusions

In this paper, we proposed a novel VM scheduling strategy with a speed switch and a multi-sleep mode in cloud data centers. By applying DPM technology and introducing a sleep mode, our proposed strategy is shown to improve energy efficiency significantly by reducing both the luxury energy consumption and the idle energy consumption. We established a continuous-time queueing model with an adaptive service rate and a partial synchronous vacation, and derived the expressions for the system performance measures in terms of the energy saving level of the system and the average delay of requests. Numerical results with analysis and simulation show that on the premise of guaranteeing the QoS of CDCs, the energy saving effect of CDCs with our proposed VM scheduling strategy is remarkably more efficient when compared with conventional CDCs. We also established a system utility function to achieve a trade off among different performance measures and designed an improved Firefly intelligent searching algorithm to obtain the optimal sleep parameter

Acknowledgments

This work was supported in part by National Natural Science Foundation (Nos. 61872311, 61472342), Hebei Province Science Foundation of China (No. F2017203141), and was supported in part by MEXT, Japan. The authors would like to thank the anonymous referees for constructive comments which greatly improved the presentation of the paper.

References

Banik A, Gupta U, Pathak S (2007). On the GI/M/1/N queue with multiple working vacations-analytic analysis and computation. *Applied Mathematical Modelling* 31(9): 1701-1710.

Chen G, Xia W, Shen L (2014). Dynamic bandwidth allocation algorithm based on transmission rate adaptation. *Journal of Communications* 35(5): 25-32 (in Chinese).

Chen Y, Chang M, Liang W, Lee C (2016). Performance and energy efficient dynamic voltage and frequency

scaling scheme for multicore embedded system. *International Conference on Consumer Electronics*, London, United Kingdom, January 18-19, 2016.

Chou C, Wong D, Bhuyan L (2016). DynSleep: Fine-grained power management for a latency-critical data center application. *International Symposium on Low Power Electronics and Design*, San Francisco, USA, August 8-10, 2016.

Dabbagh M, Hamdaoui B, Guizani M (2015). Toward energy-efficient cloud computing: Prediction, consolidation and overcommitment. *Network IEEE* 29(2): 56-61.

Dabbagh M, Hamdaoui B, Guizani M, Rayes A (2015). Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE Transactions on Network and Service Management* 12(3): 377-391.

Duan L, Zhan D, Hohnerlein J (2015). Optimizing cloud data center energy efficiency via dynamic prediction of CPU idle intervals. *International Conference on Cloud Computing*, New York, USA, June 27 - July 2, 2015.

Farahnakian F, Ashraf A, Pahikkala T (2015). Using ant colony system to consolidate VMs for green cloud computing. *IEEE Transactions on Service Computing* 8(2): 187-198.

Gandomi A, Yang X, Alavi A (2011). Mixed variable structural optimization using Firefly Algorithm. *Computers and Structures* 89(23): 2325-2336.

Gao P, Curtis A, Wong B, Keshav, S (2012). It's not easy being green. *ACM SIGCOMM Computer Communication Review* 42(4): 211-222.

Greenbaum A (1997). *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics.

Hameed A, Khoshkbarforoushha A, Ranjan R, Jayaraman P, Kolodziej J (2016). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* 98(7): 751-774.

Hintemann R, Clausen J (2016). Green cloud? The current and future development of energy consumption by data centers, networks and end-user devices. *International Conference on ICT for Sustainability*, Amsterdam, The Netherlands, Augest 29 - September 1, 2016.

Latouche G, Ramaswami V (2000). *Introduction to Matrix Analytic Methods in Stochastic Modeling, ASA-SIAM Series on Statistics and Applied Probability*. American Statistical Association.

Li K (2016). Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management. *IEEE Transactions on Cloud Computing* 4(2): 122-137.

Liao D, Li K, Sun G, Anand V, Gong Y, Tan Z (2015). Energy and performance management in large data centers: A queuing theory perspective. *International Conference on Computing, Networking and Communications*, Anaheim, California, USA, February 16-19, 2015.

Liu J, Jin S, Yue W (2018). A novel adaptive spectrum reservation strategy in CRNs and its performance optimization. *Optimization Letters* 12(6): 1215-1235.

Neuts M (1981). *Matrix-Geometric Solutions in Stochastic Models*. Johns Hopkins University Press.

Qavami H.R, Jamali S, Akbari M, Javadi B (2014). Dynamic resource provisioning in cloud computing: A heuristic Markovian approach. *International Conference on Cloud Computing*, Anchorage, USA, June 27 - July 2, 2014.

Salimian L, Esfahani F.S, Nadimi-Shahraki M.H (2012). An adaptive fuzzy threshold-based approach for energy and performance efficient consolidation of virtual machines. *Computing* 98(6): 641-660.

Shen Y, Bao Z, Qin X, Shen J (2017). Adaptive task scheduling strategy in cloud: When energy consumption meets performance guarantee. *Word Wide Web-Internet and Web Information systems* 20(2): 155-173.

Tian N, Zhang Z (2006). *Vacation Queueing Models Theory and Applications*. Springer-Verlag.

Wang Y, Xie Q, Ammari A, Pedram M (2011). Deriving a near-optimal power management policy using model-free reinforcement learning and Bayesian classification. *Design Automation Conference*, San Diego, USA, June 5-10, 2011.

Yu S, Zhu S, Ma Y, Mao D (2015). Enhancing firefly algorithm using generalized opposition-based learning. *Computing* 97(7): 741-754.

Zhang M, Hou Z (2011). Steady state analysis of the GI/M/1/N queue with a variant of multiple working vacations. *Computers and Industrial Engineering* 61(4): 1296-1301.

Shunfu Jin received the B.Eng. degree in computer science from North-East Heavier Machinery College, China, M.Eng. degree in computer science from Yanshan University, China, and Dr.Eng degree in circuits and system from Yanshan University. Now she is a professor at School of Information Science and Engineering, Yanshan University, China. Dr. Jin's research interests include stochastic modeling for telecommunication, performance evaluation for computer system and network and application for queueing system. Dr. Jin's papers have appeared in journals including Telecommunication System, Communications Networks, IEICE Transactions on Communications, Performance Evaluation, etc.

Shanshan Hao received the B.Eng. degree in computer science and technology from Hebei Normal University, Shijiazhuang, China. Now Miss. Hao is a postgraduate student at School of Information Science and Engineering, Yanshan University, China. Miss Hao's research interests

are mathematical modeling, performance evaluation and resource allocation of cloud computing.

Xiuchen Qie received the B.Eng. degree in computer science and technology from Harbin Normal University, Harbin, China. Now Miss. Qie is a postgraduate student at School of Information Science and Engineering, Yanshan University, China. Miss Qie's research interests include cloud computing, virtual machine management and stochastic modeling.

Wuyi Yue is currently a professor at the Faculty of Intelligence and Informatics, and Dean of the Graduate School of Natural Science, Konan University, Japan. Dr. Yue is a fellow of the ORSJ, a senior number of IEICE, Japan and a member of the IEEE. Dr. Yue's research interests include queueing theory, stochastic processes and optimal methods as well applied to system modeling, performance analysis and optimal resource allocation of communication networks, systems engineering, and operations research. Dr. Yue is an author (a co-author) and a co-editor of more than 10 monographs published by Springer and other Publishers, and more than 300 refereed papers published in journals such as IEEE Transactions and IEICE Transactions, and in proceedings of IEEE, ACM, LNCS, Springer.