Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 4810514, 10 pages
https://doi.org/10.1155/2017/4810514

Research Article

Ant Colony Optimization Algorithm to Dynamic Energy Management in Cloud Data Center

Shanchen Pang, Weiguang Zhang, Tongmao Ma, and Qian Gao

College of Computer & Communication Engineering, China University of Petroleum (East China), Qingdao, Shandong 266580, China

Correspondence should be addressed to Shanchen Pang; pangsc@upc.edu.cn

Received 12 July 2017; Revised 25 November 2017; Accepted 4 December 2017; Published 20 December 2017

Academic Editor: Anna Vila

With the wide deployment of cloud computing data centers, the problems of power consumption have become increasingly prominent. The dynamic energy management problem in pursuit of energy-efficiency in cloud data centers is investigated. Specifically, a dynamic energy management system model for cloud data centers is built, and this system is composed of DVS Management Module, Load Balancing Module, and Task Scheduling Module. According to Task Scheduling Module, the scheduling process is analyzed by Stochastic Petri Net, and a task-oriented resource allocation method (LET-ACO) is proposed, which optimizes the running time of the system and the energy consumption by scheduling tasks. Simulation studies confirm the effectiveness of the proposed system model. And the simulation results also show that, compared to ACO, Min-Min, and RR scheduling strategy, the proposed LET-ACO method can save up to 28%, 31%, and 40% energy consumption while meeting performance constraints.

1. Introduction

Cloud computing is a new computing model which is developed from grid computing. It includes a variety of technologies: virtualization, distributed computing, and green energy-saving technology [1]. There are mainly three types of forms to supply resources to customers: Infrastructure as a Service, Platform as a Service, and Software as a Service. It implies a service-oriented architecture, reducing information technology overhead for the end-user, great flexibility, reducing total cost of ownership, and on-demand services [2].

With the era of big data coming, computing needs have been expanding. A large number of future-generation data centers will use virtualization technology and cloud computing technology. With the expansion of the size of cloud data centers, high energy consumption of data centers becomes a serious problem. According to the relevant data, around 2016, the data centers with more than 100 blade chassis accounted for 60% of whole market of data centers (https://sanwen8.cn/p/24d2OE0.html.). The large data centers consume vast quantities of power; for example, a data center with 3000 blade chassis consumes 9000 kw per hour, and electricity charges are close to 12 million dollars every year. A lot of efforts have been made in the industry to reduce the power need in current server platforms. Therefore, how to manage the application in a cloud data center in an energy-efficient way becomes an urgent problem.

Energy consumption optimization technology for distributed parallel computing system includes three types: resource hibernation, dynamic voltage management technology, and virtualization. Resource hibernation is mainly used to reduce energy cost of idle machines, but it takes a long time to start. Dynamic voltage scaling (DVS) is a power management technique in computer architecture, where the voltage used in a component is increased or decreased, depending upon circumstances (https://en.wikipedia.org/wiki/Dynamic_voltage_scaling), but, with the decrease of voltage, processors' performance will decline. Dynamic voltage and frequency scaling (DVFS) changes the frequency and voltage of the cores, scaling performance, and power simultaneously.

Virtualization can improve the efficiency of computers and reduce costs and energy consumption. Through virtualization technology, multiple tasks run on different virtual machines, reducing energy consumption by increasing the utilization of computer resources and reducing the number of computers required.

However, the above methods have their limitations:

(1) They do not consider scheduling problems to reduce energy consumption.

(2) There are few integrated management strategies to reduce power consumption.

(3) They are applicable to implemented systems or system prototypes only.

Based on the above discussions, this paper tackles the problem of dynamic energy management in data centers. The contributions of our work to existing literature can be summarized as follows:

(1) An integrated management strategy is developed to solve the power consumption and the dynamic energy management system model for cloud data centers is built.

(2) A task-oriented resource allocation method (LET-ACO) is proposed in order to adapt to the dynamic and real-time situation and solve the issue of cloud computing resource scheduling and decrease the power consumption of data center.

2. Related Works

2.1. Optimization of Energy Consumption in Cloud Data Center. A large number of data centers use virtualization technology and cloud computing technology. The integration of traditional power management technology and virtualization technology provides new solutions to solve the power management issue of cloud data center [3–5].

Operation and power management in virtualization are major challenges in cloud platform. Firstly, the virtual resources and physical resources managed by the virtualization platform are separated from each other; therefore, how to implement the energy management strategy of application-level virtual machine is a challenging problem. Cloud data center is a constantly changing platform; considering the requirements of isolation and independence of virtual machines, the energy management strategy of virtual machine should be flexible. Secondly, the energy consumption of the virtual machine cannot be measured directly from the hardware; most energy consumption models are not accurate.

Nathuji and Schwan proposed VirtualPower approach to integrate power management mechanisms and policies with the virtualization technologies [6]. This approach supports the isolated and independent operation assumed by guest virtual machines (VMs) running on virtualized platforms and globally coordinates the effects of the diverse power management policies applied by these VMs to virtualized resources. Yang et al. used the open-source codes and PHP web programming to implement a resource management system with power-saving method for virtual machines in [7]. They proposed a system integrated with open-source software, such as KVM and Libvirt, to construct a virtual cloud management platform.

High performance can be achieved in cloud computing systems using appropriate energy management algorithms in virtual cloud platform. Beloglazov and Buyya proposed novel adaptive heuristics for dynamic consolidation of VMs based on an analysis of historical data from the resource usage by VMs in [8]. The proposed algorithms significantly reduce energy consumption, while ensuring a high level of adherence to the service level agreement. Escheikh et al. proposed workload-aware power management (PM) performability analysis of server-virtualized system (SVS) in [9]. This modeling approach delivers a precise description of different entities and features of the SVS and provides effective support for dynamic time-based PM policy enabling opportunistic selection of suitable power states of power manageable component (PMC).

Dynamic voltage scaling is a power management technique in computer architecture, where the voltage used in a component is increased or decreased, depending upon circumstances [10]. Rossi et al. proposed a novel dynamic voltage scaling (DVS) approach for reliable and energy efficient cache memories in [11]. They also developed a design exploration framework allowing us to evaluate several possible trade-offs between power consumption and reliability. Chen et al. presented a method for dynamic voltage/frequency scaling of networks-on-chip and last level caches in multicore processor designs, where the shared resources form a single voltage/frequency domain in [12]. These techniques reduce energy delay product by 56% compared to a state-of-the-art prior work. Moons and Verhelst generalized the Dynamic Voltage Accuracy Scaling concept to pipelined structures and quantified its energy overhead in [13]. DVAS technology includes three parts: energy saving through voltage scaling, accuracy scaling through bit width reduction, and voltage scaling through bit width reduction. DVAS is finally applied to a JPEG image processing application, demonstrating large system level gains. Arroba et al. proposed a DVFS policy that reduces power consumption while preventing performance degradation and a DVFS-aware consolidation policy that optimizes consumption, considering the DVFS configuration that would be necessary when mapping virtual machines to maintain Quality of Service in [14]. Han et al. proposed an off-line dynamic voltage scaling (DVS) scheme that can be integrated with EDF, which is a global real-time scheduling algorithm for symmetric multiprocessor systems in [15]. However, these techniques are unsatisfactory in minimizing both schedule length and energy consumption.

Resource hibernation is the setting or prediction of the closing/sleeping time for computer processing components. There are many challenges for resource hibernation technology in cloud computing systems with many cloud computing resources. For example, the shortcomings of the traditional scheduling strategies can lead to computer load imbalance; besides, using resource hibernation technology will obviously seriously affect the performance of the entire system.

However, the above researches discuss energy-saving methods only from the system/hardware management; the appropriate task scheduling strategies can also achieve the goal of saving energy in cloud data centers.

2.2. Task Scheduling Optimization. Cloud computing task scheduling refers to the allocation and management of resources in a specific cloud environment according to certain resource usage rules. Task scheduling problems are related to the efficiency of all computing facilities and are of paramount importance [16]. Cloud computing task scheduling is a NP-complete problem; it can be solved in different methods: traditional deterministic algorithms and heuristic intelligent algorithms [17–25]. However, those methods don not take energy consumption into account, and, to overcome this limitation, researchers have proposed some approaches. Li et al. proposed a heuristic energy-aware stochastic task scheduling algorithm called ESTS to solve energy-efficient task scheduling problem in [26]. Zhang et al. presented new energy-efficient task scheduling algorithms for both continuous and discrete processor speeds with detailed simulation and analyses in [27]. Changtian and Jiong adopted the genetic algorithm to parallel find the reasonable scheduling scheme in [28].

Nevertheless, all these studies explored different ways of energy conservation in cloud data center and did not consider the integrated management strategy. Our approach explores a set of energy-saving schemes, and the task scheduling algorithm is innovated. Since the task scheduling model is a NP-complete problem in cloud computing and considering the supremacy of ant colony optimization algorithm (ACO) for solving task scheduling optimization in the cloud and grid environment [29–31], we select ACO in this paper to find the schema for task scheduling and achieve energy-consumption reduction in the cloud data centers.

3. System Model

The dynamic energy management system model is shown in Figure 1. This system is composed of DVS Management Module, Load Balancing Module, and Task Scheduling Module. The cloud data center accepts a task arrival flow that enters a waiting queue, DVS Management Module observes the load of each machine and gives commands to optimize power consumption, and the load state of the machines and the queue state are transferred to Load Balancing Module, which gets the available virtual machine resources. Task Scheduling Module accepts task information and assigns tasks to available virtual machines.

(1) DVS Management Module. DVS technologies can manage the applications in a cloud data center in an energy-efficient way. This module monitors the running state of each machine and gives commands to optimize power consumption, and the state of machine i can be calculated as follows:

$$\text{State}(i) = \frac{\text{RunVM}(i)}{\text{MaxVM}(i)}, \tag{1}$$

where RunVM(i) represents the number of VM instances running on machine i and MaxVM(i) represents the maximal number of VM instances a machine can host; it needs to consider the situation of physical server, virtual machine load, and so on.

A threshold value ξ ($0 < \xi < 1$) is set to 0.5. When State(i) is higher than ξ, the DVS Management Module issues commands to improve machine speed level. When it is lower than ξ, it issues downscaling commands.

(2) Load Balancing Module. We calculate the load rate of each machine according to the status data returned by the DVS Management Module. Considering the structural characteristics of cloud computing system, the load rate of machine i is used as an index of resource utilization, and its calculation method is as follows:

$$\text{Load}(i) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\text{thisMI} - \text{sysMI}\right)^2}, \tag{2}$$

where thisMI represents the utilization of machine i, sysMI denotes the utilization of system, and n is the total number of machines.

A threshold value θ ($0 < \theta < 1$) is set to 0.65; when Load(i) is lower than θ, virtual machines carried on machine i are added to standby queue.

(3) Task Scheduling Module. Task Scheduling Module accepts task information and assigns tasks to available virtual machines. The proposed LET-ACO algorithm is applied to task scheduling; the goal is to reduce the power consumption of data center effectively on the premise of performance guarantee, and specific method will be shown in Sections 4 and 5.

4. Problem Formulation

To further analyze the problem, we set up the following cloud task scheduling model.

Definition 1. A set of tasks $T_i = \{T_1, T_2, \ldots, T_m\}$; it indicates m tasks in the current queue; a set of virtual machines $\text{VM}_j = \{\text{VM}_1, \text{VM}_2, \ldots, \text{VM}_n\}$; it indicates n available computing resources; the distribution of tasks is defined as an $m \times n$ matrix $R_{m\times n}$:

$$R_{m\times n} = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ R_{21} & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & & \vdots \\ R_{m1} & R_{m2} & \cdots & R_{mn} \end{pmatrix}, \tag{3}$$

where R_{ij} is the number of T_i running on VM_j.

For a complex task, the task system first decomposes it into several smaller tasks [32] and then allocates tasks to appropriate computing resources; finally the calculation results are summarized. Task decomposition should be followed by a number of principles:

(1) Independence: maintaining relative independence between subtasks

(2) Hierarchy: the tasks are decomposed in layers according to certain order: parameter, test object, and function

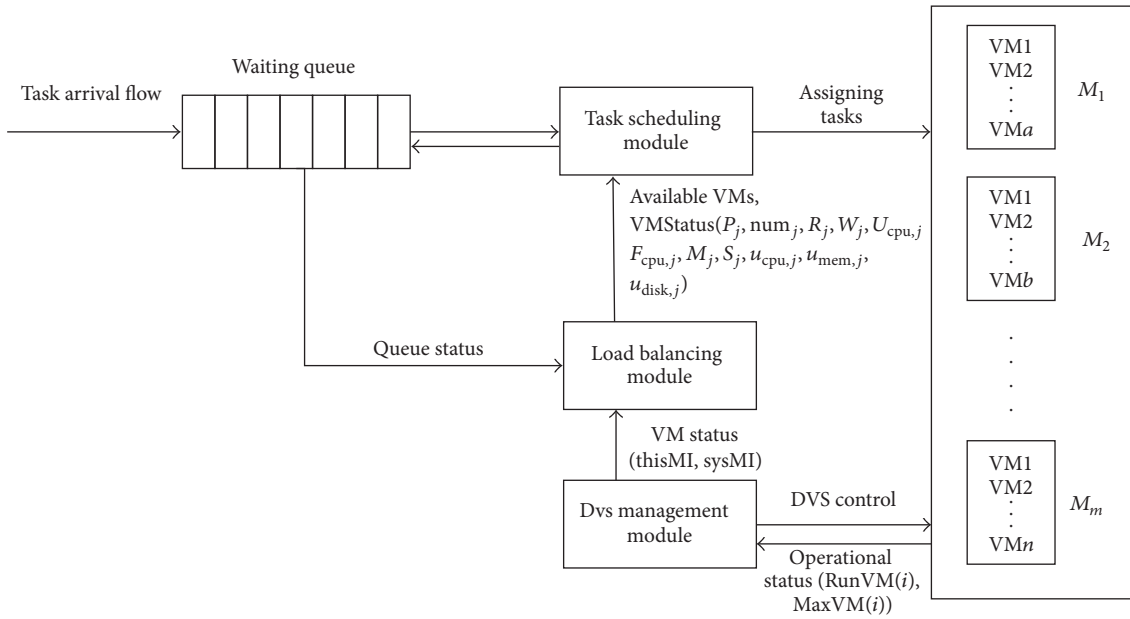

Figure 1: Dynamic energy management system model.

(3) Uniformity: the granularity of subtasks is homogeneous

(4) Similarity: the decomposed subtasks are as similar as possible

To illustrate the task scheduling process, we use Stochastic Petri Net (SPN) [33] to build the model. The task scheduling process for the cloud data center is shown in Figure 2.

Definition 2. A Petri Net corresponds to a 6-tuple:

$$\text{SPN} = (P, T, F, W, M_0, \lambda), \tag{4}$$

where P and T are disjoint sets of places and transitions, respectively, F belongs to $(S\times T)\cup(T\times S)$, $WF \to N^+$ is the set of arc functions, $M_0 : P \to \{0, 1, 2, \ldots\}$ is the initial marking, and λ is the average implementation rate of transitions.

Tables 1 and 2 present the meaning of P (place) and T (transition).

Table 1: The meaning of place.

Place	Explanation
p	Sets of tasks
$p_{a1}, p_{a2}, p_{a3}, \ldots, p_{am}$	Tasks of decomposition
$p_{b1}, p_{b2}, p_{b3}, \ldots, p_{bm}$	Tasks to be assigned
$p_{c1}, \ldots, p_{cn}$	Computing resources
$p_{d1}, \ldots, p_{dn}$	Intermediate result sets
p_e	Output result sets

Table 2: The meaning of transition.

Transition	Explanation
t	Segmenting tasks
$t_{a1}, t_{a2}, t_{a3}, \ldots, t_{am}$	Inputting tasks
$t_{b11}, \ldots, t_{b1n}; t_{b21}, \ldots, t_{b2n}; t_{b31}, \ldots, t_{b3n}; t_{bm1}, \ldots, t_{bmn}$	Matching resources
$t_{c1}, \ldots, t_{cn}$	Outputting intermediate result sets
t_d	Summary processing

5. Resource Preallocation

In this section, we describe our optimization model for task scheduling using LET-ACO method.

5.1. Prediction

5.1.1. Task Execution Time Prediction Model. The resources in the cloud data centers are full of uncertainty; in one aspect, the hardware's abilities of different resources are different, and the CPU load and network load are varying in every minute; even the same task has different execution time in the same resource if it is submitted at different time; in the other aspect, if two different tasks were executed in two resources with the same station, the execution times are different, too. The uncertainty in the two aspects makes it complex to predict a task's execution time in cloud data centers.

According to the features of heterogeneity and dynamic changes in the cloud computing environment, Dinda presented a detailed statistical analysis of the dynamic load information in [34], time series analysis of the traces shows that load is strongly correlated over time, and the relationship between them is almost entirely linear. Therefore, we design a task execution time prediction model based on linear regression model. In order to enhance the reliability of the prediction model, we need to make a hypothesis about the application of prediction:

(1) The task priority being basically the same; it can ensure the accuracy of prediction

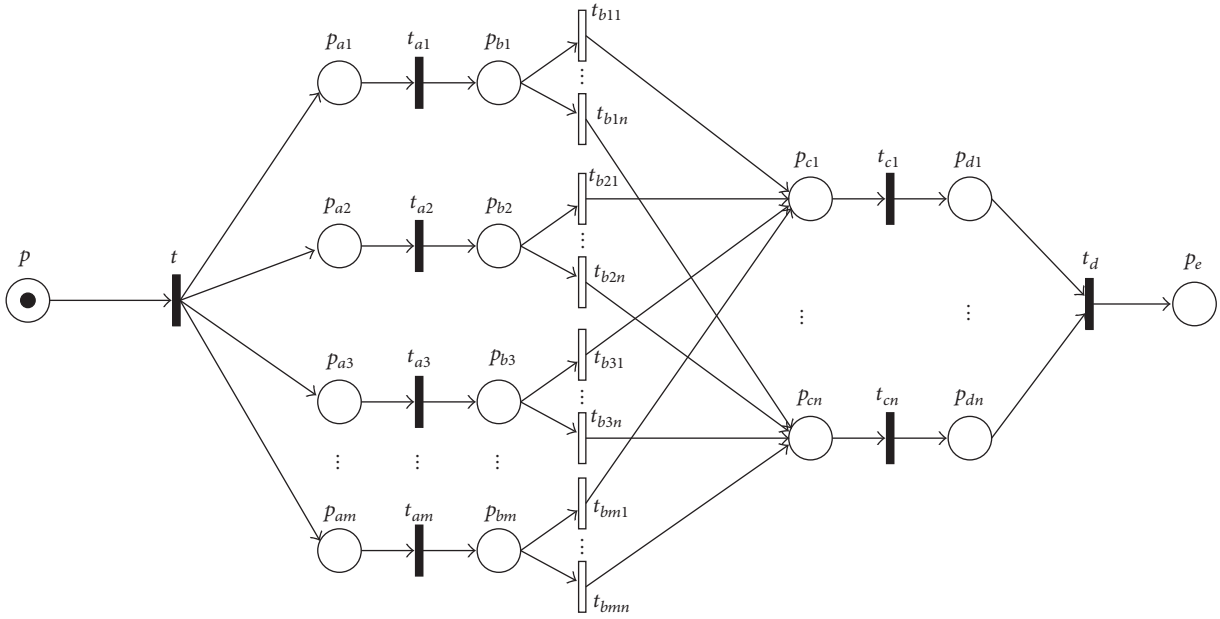

Figure 2: Stochastic Petri Nets model of task scheduling.

(2) Experiment with tasks of the same type (the amount of resources used by the tasks)

Since the current state of the calculated node is known, we use the following model to predict next task's execution time on VM_j:

$$\mathrm{Time}_j(t) = \alpha_0 + \alpha_1 U_{\mathrm{cpu},j} + \alpha_2 F_{\mathrm{cpu},j} + \alpha_3 M_j + \alpha_4 S_j + \varepsilon, \tag{5}$$

where $U_{\mathrm{cpu},j}$ represents the CPU utilization, $F_{\mathrm{cpu},j}$ represents the CPU basic frequency, M_j denotes the memory capacity, and S_j denotes network bandwidth; α_0, α_1, α_2, α_3, and α_4 are the regression coefficient; ε represents random error.

This paper uses statistical methods, so we get a great amount of data: task execution time, CPU utilization, CPU basic frequency, memory capacity, and network bandwidth to compute regression coefficient. The experimental results show that the average relative error of task execution time is maintained at less than 6%.

5.1.2. Energy Consumption Prediction Model. In the real cloud computing system, an important aspect of energy management technology is the visibility of energy usage. However, we cannot directly obtain the energy consumption state data of hardware due to the existence of virtualization layer. Therefore, we need to build an indirect energy-consumption measurement mechanism for virtual machines. This paper employs the energy-consumption measurement model of virtual machine used in [35]. The system's energy consumption is mainly composed of CPU, memory, disk, and system-idle energy consumption; it can be expressed as follows:

$$\begin{aligned} E_{\mathrm{sys},j} &= E_{\mathrm{cpu},j} + E_{\mathrm{Mem},j} + E_{\mathrm{Disk},j} + E_{\mathrm{static},j} \\ &= \alpha_{\mathrm{cpu}} u_{\mathrm{cpu},j} + \alpha_{\mathrm{mem}} u_{\mathrm{mem},j} + \alpha_{io} u_{\mathrm{disk},j} + \gamma, \end{aligned} \tag{6}$$

where $u_{\mathrm{cpu},j}$ denotes processor utilization, $u_{\mathrm{mem},j}$ represents the number of LLC (last level cache) misses for a VM across all cores used by it during time period, and $u_{\mathrm{disk},j}$ represents the sum of bytes read and written; α_{cpu}, α_{mem}, α_{io}, and γ are model-specific constants.

Processor utilization can be easily obtained from the processor usage in the operating system; most processors have the LLC count function; Intel Nehalem processor provides this functionality on each core, tracking I/O operation in Hypervisor to get the sum of bytes read and written. After obtaining a series of experimental data, we use linear regression with ordinary least-squares estimation to obtain parameters to establish the model. The experimental results show that average relative error of the system's energy consumption is kept within 10%.

5.2. Task Scheduling Algorithm Based on LET-ACO. The heterogeneous computing platform meets the computational demands of diverse tasks. One of the key challenges of such heterogeneous processor systems is effective task scheduling [30]. The task scheduling problem is generally NP-complete. Many real-time applications are discrete optimization problems. There is no polynomial time algorithm to solve combinatorial optimization problems that are NP-hard. Heuristic algorithm can solve this problem better. In this paper, the improved ant colony algorithm is adopted to solve task scheduling problem.

Ant colony algorithm, which has the advantages of positive feedback, distributed parallel computer, more robustness, and being easy to combine with other optimization algorithms, is a heuristic algorithm with group intelligent bionic computing method. Ant colony algorithm does well in finding out the appropriate computing resources in the unknown network topology.

5.2.1. System Initialization. At the initial time of the system, ants are randomly placed on VM_j, which needs to provide CPU basic frequency P_j, the number of CPU num_j, memory capacity R_j, and network bandwidth W_j. System initializes the value of the pheromone on each virtual machine, using the following equation. b_{cpu}, b_{mem}, and b_{net} are model-specific constants according to the proportion of each factor; $b_{cpu} + b_{mem} + b_{net} = 1$.

$$\tau_j(0) = b_{cpu}\left(num_j \times P_j\right) + b_{mem}R_j + b_{net}W_j. \quad (7)$$

5.2.2. The State Transfer Probability. Every ant chooses virtual machines judging by pheromone and heuristic information. During the iterative process, ${P_{ij}}^k(t')$ (state transition probability) relies on both the amount of information (virtual machine processing capability and energy-consumption information) and the existing heuristic information on the path. Individual ants represent decomposed tasks. At time t', the kth ant chooses VM_j for the next task i with a probability as shown in the following equation:

$${P_j}^k\left(t'\right) = \begin{cases} \dfrac{\left[\tau_j\left(t'\right)\right]^{\alpha}\left[\eta_j\right]}{\sum_{l=1}^{n}\left[\tau_l\left(t'\right)\right]^{\alpha}\left[\eta_l\right]} & \text{if } l, j \in 1, \ldots, n \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

where α is the information heuristic factor that indicates the importance of the resource pheromone; $\tau_j(t')$ represents the pheromone concentration of VM_j at time t'; η_j represents the expectation that next task will be executed on VM_j, $\eta_j = I_j(t)$, and $I_j(t)$ is the profit function that represents VM_j profit when next task is running on VM_j; its calculation method is as follows:

$$I_j(t) = \frac{1}{\lambda \mathrm{Time}_j(t)} - \lambda' E_j(t), \quad (9)$$

where λ and λ' are set according to the experimental data; then our algorithm is used to get the experimental results and the parameters are adjusted. ${P_j}^k(t')$ is to be zero when the virtual machine j has been selected or the virtual machine j fails.

5.2.3. Pheromone Updating. Pheromone updating is done by all the ants that come up with feasible schedules in the following manner as shown in the following equation:

$$\tau_j\left(t' + 1\right) = (1 - \rho)\,\tau_j\left(t'\right) + \Delta\tau_j, \quad (10)$$

where $\tau_j(t' + 1)$ represents the pheromone concentration of VM_j at the next moment; ρ is volatile factor: $0 \le \rho < 1$; $\Delta\tau_j$ represents pheromone increment, and its calculation method is as follows:

$$\Delta\tau_j = D \cdot \max\left(\mathrm{RES}\left(I_{k,r}\right)\right), \quad (11)$$

where D is the intensity of pheromone and it affects the rate of convergence; $\mathrm{RES}(I_{k,r})$ represents the task assignment scheme in which the kth ant has searched in the rth iteration whose value is determined by the profit function value of the allocation scheme.

If the ant k completes the round search (a task allocation scheme has been found), all the virtual machines on this path will update the local pheromone. If all ants complete the round search, finding the optimal path in this iteration, the virtual machines on the optimal path will update the local pheromone.

5.2.4. LET-ACO Algorithm Flow. The proposed LET-ACO algorithm is described according to the following steps.

Step 1. DVS Management Module obtains the running status information of each physical host and virtual machine. Then, it uses DVS technology to control hosts according to State(i).

Step 2. The load state of machines and the queue state are transferred to Load Balancing Module which can get the available virtual machine resources.

Step 3. The profit matrix is obtained by calculating $I_j(t)$.

Step 4. Task Scheduling Module sets initial pheromone for available computing nodes.

Step 5. Initialize the ants and place them randomly on the available virtual machines.

Step 6. Calculate the transition probability of ant k according to the profit matrix, and choose next node.

Step 7. If the ant k completes the round search, local pheromone will be updated; if not, return to Step 6.

Step 8. If all ants complete the round search, global pheromone will be updated; if not, return to Step 5.

Step 9. If the number of iterations is required, Task Scheduling Module outputs the optimal allocation scheme; if not, return to Step 5.

Step 10. Determine whether there is any task to be allocated in the task queue, and if so, return to Step 1, and if not, end this task assignment.

LET-ACO runs periodically; Figure 3 depicts the main flow of the algorithm.

6. Experiments and Performance Analysis

To evaluate our proposed method, the simulation experiments are operated on CloudSim to analyse the effects. We

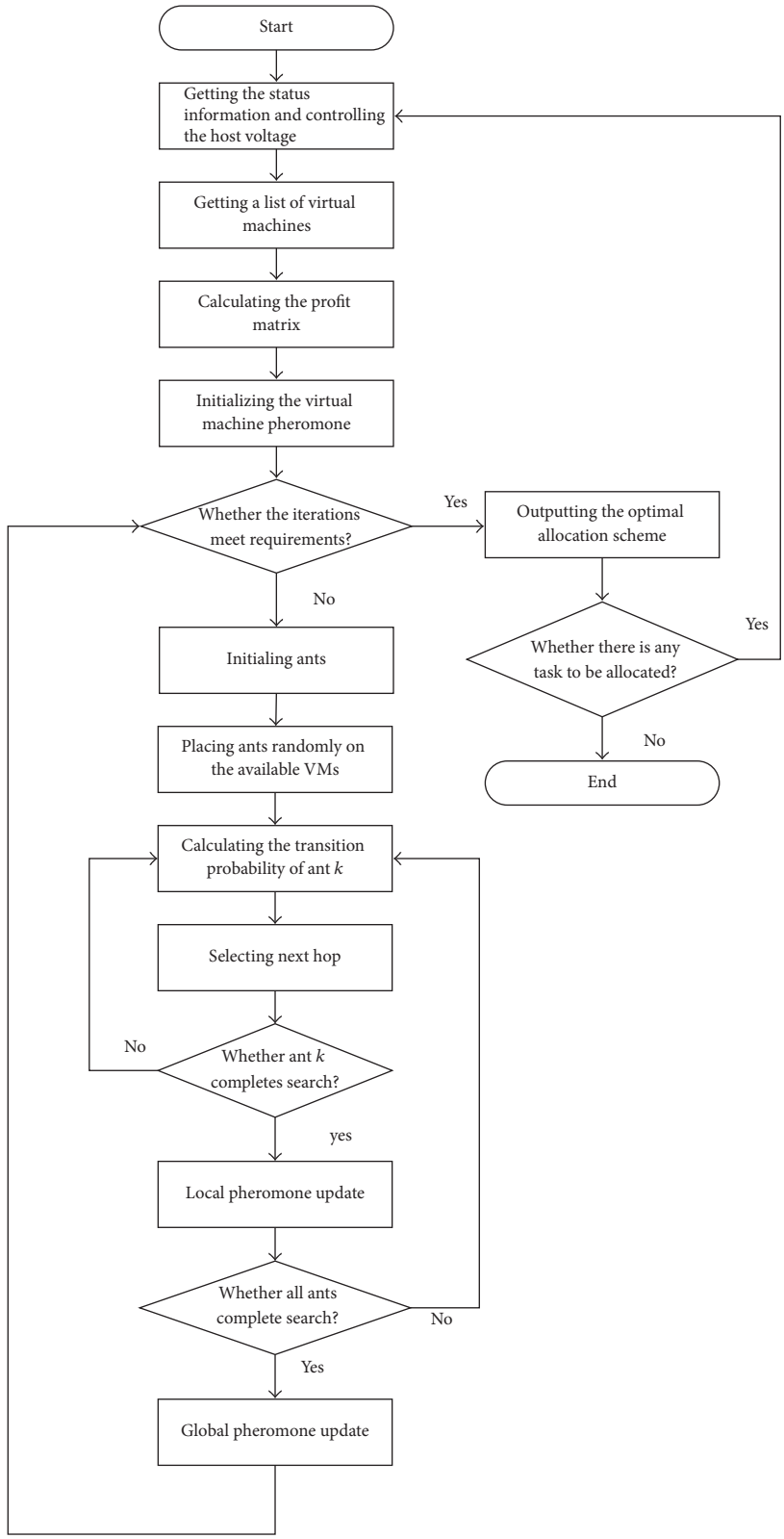

Figure 3: Main flow of the LET-ACO algorithm.

Table 3: Parameters of CloudSim.

Type	Parameter	Value
Data center	Number of data centers	1
	Number of hosts per data center	6
Virtual machine (VM)	Total number of VMs	30
	MIPS of PE	500–1500 (MIPS)
	Number of PEs per VM	2–8
	VM memory	512–2048 (MB)
	Bandwidth	500–1000 bit
Task	Total number of tasks	100–400
	Length of task	5000 (MI)

Table 4: Parameters of LET-ACO algorithm.

Parameter	Value
Number of ants in colony	25
Number of iterations	20
α	0.2
ρ	0.5

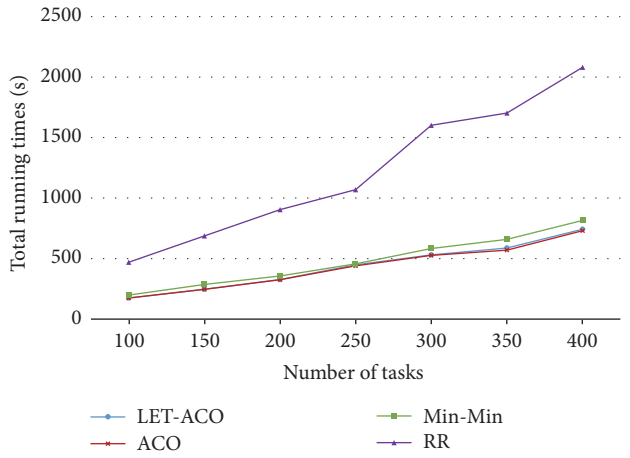

Figure 4: Execution times of tasks.

design the simulation environment by assuming that there are 6 hosts, 30 VMs, and 100–400 tasks. Data and information about hosts, VMs, and tasks are summarized in Table 3.

In this experiment, the parameters of the LET-ACO algorithm are set in Table 4.

Based on the parameters' settings in Tables 3 and 4, results of three metrics are obtained and illustrated in Figures 4–7.

Figure 4 shows the comparison of task total running time using LET-ACO, ACO, Min-Min, and Round-Robin (RR) algorithms. ACO algorithm [36] is the basic ant colony algorithm involving time factor. Min-Min algorithm [37] and RR algorithm [38] are the classic algorithms employed by process and network schedulers in computing. It is shown that the task completion time gets longer with the increase of tasks number. Task execution time is the longest by using RR algorithm; ACO and LET-ACO algorithm are obviously superior to the other algorithms. These results suggest that

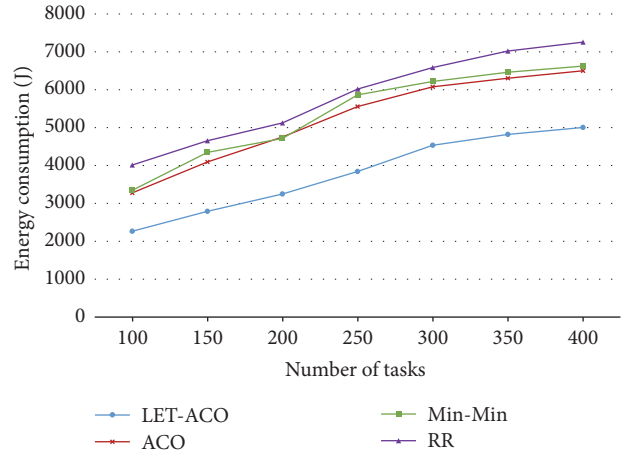

Figure 5: Energy consumption of tasks running in system.

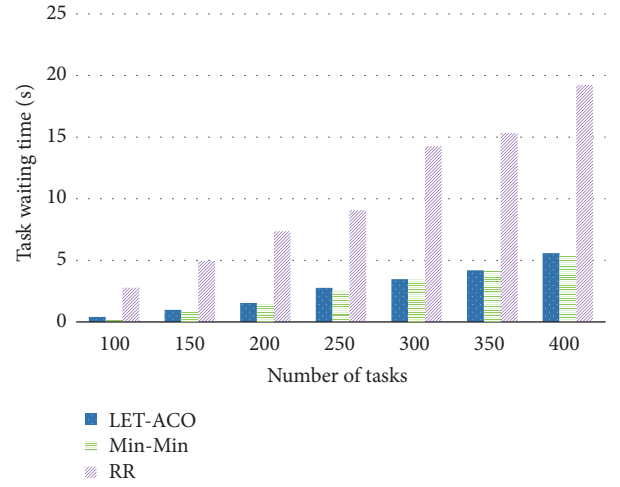

Figure 6: Tasks' average waiting time.

LET-ACO is very effective in finishing the tasks with small VM time, and it takes power into consideration, but it does not bring any increase of the makespan.

In Figure 5, it is shown that the energy that the data center consumes increases with the number of tasks added. Compared with the other three algorithms, the proposed

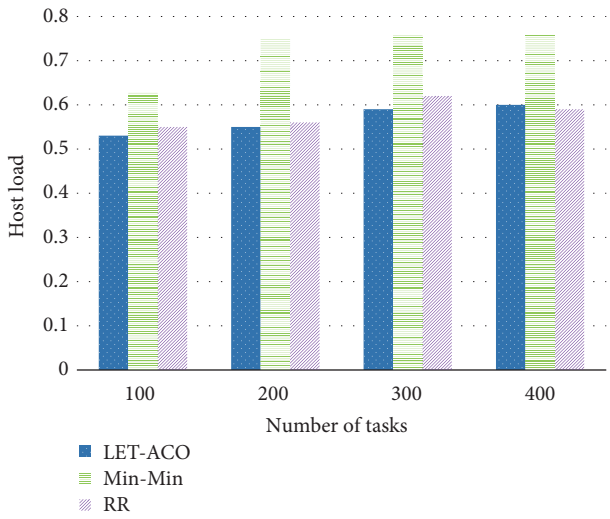

Figure 7: Host load.

algorithm can significantly reduce the system's energy consumption. To analyze the reason, it is just because ACO, Min-Min, and RR algorithms focus solely on the completion time of the task without considering energy consumption.

Figure 6 shows the task average waiting time for different algorithms. The results show that the task waiting time based on LET-ACO algorithm is shorter, which can meet the needs of users better.

Figure 7 shows the host load using different algorithms. The cloud system load has been maintained in a relatively balanced state by using LET-ACO algorithm. Min-Min algorithm can complete the task in a short time, but the load balancing of Min-Min algorithm is the worst. The resources with strong processing ability are always in the working state, while other resources are idle all the time; Min-Min algorithm cannot reflect the advantages of distributed system.

Taken together, LET-ACO algorithm can reduce the power consumption of data center effectively on the premise of performance guarantee.

7. Conclusion

This work proposes an integrated management strategy to solve the data center power consumption problem. The resource scheduling system model is built for cloud data center and this system is analysed by Stochastic Petri Net. A task-oriented resource allocation method (LET-ACO) is proposed; it can reduce the power consumption of data center effectively on the premise of performance guarantee. To validate the effectiveness of our proposed method, the simulation experiments are operated on CloudSim.

In the future work, we will try to build more detailed system model to describe the data center or to build a new kind of model which can be greatly effective for analysing particular problems in cloud data center including heterogeneous tasks scheduling and fault diagnosing, and we may take more factors into consideration; for example, not only time and power consumption but also the state of the hosts can influence the energy of the data center; another promising future work direction is to try to use other biocomputing methods to solve some problems in cloud data center.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grants nos. 61572522, 61572523, and 61402533) and special fund project for work method innovation of Ministry of Science and Technology of China (no. 2015IM010300).

References

[1] C. Weinhardt, W. A. Anandasivam, B. Blau et al., "Cloud computing—a classification, business models, and research directions," *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, 2009.

[2] M. Vouk, "Cloud computing issues, research and implementations," *Journal of Computing and Information Technology*, vol. 16, no. 4, pp. 235–246, 2008.

[3] K.-J. Ye, Z.-H. Wu, X.-H. Jiang, and Q.-M. He, "Power management of virtualized cloud computing platform," *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 35, no. 6, pp. 1262–1285, 2012.

[4] Y. Zhang, W. Pan, Q. Wang, K. Bai, and M. Yu, "Hypebios: enforcing vm isolation with minimized and decomposed cloud tcb," Tech. Rep., Virginia Commonwealth University, 2012.

[5] Y. Zhang, M. Li, B. Wilder, M. Yu, K. Bai, and P. Liu, "NeuCloud: enabling privacy-preserving monitoring in cloud computing".

[6] R. Nathuji and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems," in *Proceedings of the SOSP'07: 21st ACM Symposium on Operating Systems Principles*, pp. 265–278, October 2007.

[7] C.-T. Yang, J.-C. Liu, S.-T. Chen, and K.-L. Huang, "Virtual machine management system based on the power saving algorithm in cloud," *Journal of Network and Computer Applications*, vol. 80, pp. 165–180, 2017.

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

[9] M. Escheikh, K. Barkaoui, and H. Jouini, "Versatile workload-aware power management performability analysis of server virtualized systems," *The Journal of Systems and Software*, vol. 125, pp. 365–379, 2017.

[10] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, "A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 73–83, 2015.

[11] D. Rossi, V. Tenentes, S. Khursheed, and B. M. Al-Hashimi, "BTI and leakage aware dynamic voltage scaling for reliable low power cache memories," in *Proceedings of the 21st IEEE International On-Line Testing Symposium, IOLTS 2015*, pp. 194–199, July 2015.

[12] X. Chen, Z. Xu, H. Kim et al., "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *Proceedings of the 50th Annual Design Automation Conference, DAC 2013*, p. 114, June 2013.

[13] B. Moons and M. Verhelst, "DVAS: dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *Proceedings of the 20th IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED 2015*, pp. 237–242, July 2015.

[14] P. Arroba, J. M. Moya, J. L. Ayala, and R. Buyya, "Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers," *Concurrency Computation: Practice and Experience*, vol. 29, no. 10, Article ID e4067, 2017.

[15] S. Han, M. Park, X. Piao, and M. Park, "A dual speed scheme for dynamic voltage scaling on real-time multiprocessor systems," *The Journal of Supercomputing*, vol. 71, no. 2, pp. 574–590, 2015.

[16] F. Ramezani, J. Lu, and F. K. Hussain, "Task-based system load balancing in cloud computing using particle swarm optimization," *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 739–754, 2014.

[17] E. Pacini, C. Mateos, and C. G. Garino, "Distributed job scheduling based on swarm intelligence: a survey," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 252–269, 2014.

[18] T. Ma, Q. Yan, W. Liu, D. Guan, and S. Lee, "Grid task scheduling: algorithm review," *IETE Technical Review*, vol. 28, no. 2, pp. 158–167, 2011.

[19] S. Pang, T. Ma, and T. Liu, "An improved ant colony optimization with optimal search library for solving the traveling salesman problem," *Journal of Computational and Theoretical Nanoscience*, vol. 12, no. 7, pp. 1440–1444, 2015.

[20] T. Song and L. Pan, "Spiking neural P systems with request rules," *Neurocomputing*, vol. 193, pp. 193–200, 2016.

[21] T. Song, F. Gong, X. Liu, Y. Zhao, and X. Zhang, "Spiking neural P systems with white hole neurons," *IEEE Transactions on NanoBioscience*, vol. 15, no. 7, pp. 666–673, 2016.

[22] T. Song, P. Zheng, M. L. D. Wong, and X. Wang, "Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control," *Information Sciences*, vol. 372, pp. 380–391, 2016.

[23] X. Wang, T. Song, P. Zheng, S. Hao, and T. Ma, "Spiking neural P systems with anti-spikes and without annihilating priority," *Science and Technology*, vol. 20, no. 1, pp. 32–41, 2017.

[24] Y. Wen, H. Xu, and J. Yang, "A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system," *Information Sciences*, vol. 181, no. 3, pp. 567–581, 2011.

[25] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, 2014.

[26] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2867–2876, 2014.

[27] L. M. Zhang, K. Li, D. C.-T. Lo, and Y. Zhang, "Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds," *Sustainable Computing: Informatics and Systems*, vol. 3, no. 2, pp. 109–118, 2013.

[28] Y. Changtian and Y. Jiong, "Energy-aware genetic algorithms for task scheduling in cloud computing," in *ChinaGrid Annual Conference (ChinaGrid)*, pp. 43–48, 2012.

[29] W.-T. Wen, C.-D. Wang, D.-S. Wu, and Y.-Y. Xie, "An ACO-based scheduling strategy on load balancing in cloud computing environment," in *Proceedings of the 9th International Conference on Frontier of Computer Science and Technology, FCST 2015*, pp. 364–369, August 2015.

[30] H. Sun and J. Zhu, "Design of task-resource allocation model based on Q-learning and double orientation aco algorithm for cloud computing," *Computer Measurement & Control*, 2014.

[31] G. Lin, Y. Bie, M. Lei, and K. Zheng, "ACO-BTM: a behavior trust model in cloud computing environment," *International Journal of Computational Intelligence Systems*, vol. 7, no. 4, pp. 785–795, 2014.

[32] C. Mateos, A. Zunino, and M. Campo, "A survey on approaches to gridification," *Software: Practice and Experience*, vol. 38, no. 5, pp. 523–556, 2008.

[33] H. He, S. Pang, and Z. Zhao, "Dynamic scalable stochastic petri net: a novel model for designing and analysis of resource scheduling in cloud computing," *Scientific Programming*, vol. 2016, Article ID 9259248, 13 pages, 2016.

[34] P. A. Dinda, "The statistical properties of host load," *Scientific Programming*, vol. 7, no. 3-4, pp. 211–229, 1999.

[35] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10)*, pp. 39–50, June 2010.

[36] C. Mateos, E. Pacini, and C. G. Garino, "An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments," *Advances in Engineering Software*, vol. 56, pp. 38–50, 2013.

[37] T. D. Braunt, H. J. Siegel, N. Beck et al., "A comparison study of eleven static heuristics for mapping a class of independent tasks onto ileterogeneous distributed computing systems," 2000.

[38] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Arpaci-Dusseau Books, 2015.