Research Article

Design of Virtual Machine Scheduling Algorithm in Cloud Computing Environment

Bin Liang,1 Ruifeng Liu,2 and Dongfeng Dai1

1School of Electromechanical and Automotive Engineering, Jiuzhou Polytechnic, Xuzhou 221116, China
2School of Public Administration, Sichuan University, Chengdu, Sichuan 610000, China

Correspondence should be addressed to Bin Liang; liangb@jzp.edu.cn

Received 14 March 2022; Revised 2 April 2022; Accepted 15 April 2022; Published 9 May 2022

Academic Editor: Han Wang

Copyright © 2022 Bin Liang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the application of cloud computing services in more and more fields, it will undertake more computing tasks and storage tasks. The problem of high energy consumption in data centers will become more serious. Virtualization technology is very important in cloud computing, which can improve the utilization rate of resources. At the same time, it has flexibility in resource scheduling and can integrate multiple virtual machines to achieve power efficiency. Using online virtual machine migration technology for energy-saving planning in cloud environment is a hot research topic in academic circles. The scheduling strategy proposed in this paper can reduce the server downtime and the number of server hosts, so as to achieve the maximum use of resources. The main research work of this paper includes the following aspects: Firstly, the energy consumption in the process of virtual machine energy-saving design is modeled, and the relationship between energy consumption and resource usage under different load conditions is analyzed, and the problems are abstracted, e.g., packaging box problem. Secondly, this paper uses genetic algorithm to solve the problem of high energy consumption. Finally, based on the target allocation scheme of virtual machines obtained by the above method, the migration problem of virtual machines is abstracted as the problem of finding the maximum weighted independent set of graphs. And a greedy algorithm is designed to solve this problem. In this paper, CloudSim simulation platform is used to verify the effectiveness of the proposed algorithm. Experiments show that the proposed algorithm can reduce data energy consumption and avoid frequent migration of virtual machines.

1. Introduction

If the virtual machine scheduling problem is not considered carefully, the system load will be unbalanced. To solve this problem, this paper proposes a scheduling strategy based on genetic algorithm to reduce dynamic migration and achieve the best load balance [1]. Because cloud computing has many difficult problems to solve, so how to effectively solve these problems and improve efficiency, this paper proposes a dual fitness genetic algorithm. The research shows that this method can not only shorten the total task time but also shorten the average task time [2]. Big data technology cannot solve the problems in data mining efficiently. Therefore, this paper puts forward the model of cloud computing, which can provide more benefits. It has the characteristics of flexible cost and more flexibility in computing resources [3]. In cloud computing, providing appropriate and reliable application services is the basic requirement. The lightweight virtual machine scheduler of metascheduler proposed in this paper can improve resource utilization and can be further expanded to achieve adaptive load balancing [4]. Genetic algorithm can solve the load balancing problem well. Experiments show that this strategy solves the problems of unbalanced load and high migration cost to a great extent [5]. At present, the offline method cannot be used to solve the problems in uncertain environment, because this method can only solve the certain problems. Efficient cloud computing scheduling method can reduce the energy consumption of cloud computing platform and improve resource utilization [6]. Cloud computing has high requirements on basic equipment and resources. Efficient load balancing can improve the utilization of resources. In order to
solve different cloud environments, this paper proposes a variety of load balancing schemes [7]. Experimental results show that ant colony optimization resource allocation method be able to meet the needs of cloud computing environment [8]. In order to solve the problems in cloud security, this paper proposes keystroke dynamics method, which can provide a very effective solution [9]. Aiming at the problem of resource utilization in cloud computing, this paper proposes a virtual machine scheduling algorithm based on K-means clustering, which can make full use of resources and has strong stability [10]. In this paper, we propose a virtual machine scheduling algorithm to solve the mixed load scheduling problem, which uses processors to evaluate the recent energy consumption problem of virtual machines. Experiments show that this method is used in scheduling deviation and normalized energy efficiency [11]. Virtual machine has been widely used because it can improve the utilization rate of resources. With the change of cloud computing scale, the requirements for resource utilization are getting higher and higher, so a more efficient method is needed to improve the allocation of resources [12]. Sustainable green intelligent computer can solve the problem of high energy consumption in cloud computing. It only needs to pay attention to the scheduling of cloud virtual machines, consider the actual configuration of servers, and avoid overheating in the central local location [13]. Task scheduling is the focus of cloud data research. At present, the long execution time means that its cost is also very high, and reducing the execution time means reducing the cost, so this paper proposes a task scheduling algorithm with short average execution time to solve this problem [14]. There are many problems to be solved in resource scheduling of virtual machines. The algorithm proposed in this paper combines genetic algorithm to continuously adjust the host load, so as to achieve the optimal balance [15].

2. Features of Cloud Computing

2.1. Virtualization. The basic feature of cloud computing is virtualization technology. The resources retrieved by users in cloud services are virtualized, not real resources. Cloud services take advantage of virtualization technology to a certain extent, and eliminate the differences between different server architectures by protecting the details of specific hardware and virtualizing IT resources. In cloud services, virtualization of CPU resources is the purpose of virtualization. These resources are usually provided to users in the form of virtual machines, which is convenient for users to use. For cloud providers, resource virtualization helps build an easy-to-use management platform.

2.2. High Availability. Cloud computing platform provides reliable services according to users’ needs. In the cloud server center, the stable operation of the whole system mainly depends on the centralized management of large-scale resources under the unified platform. Data redundancy technology and load balancing technology provide reliable guarantee for the high availability of services. Hadoop’s heartbeat mechanism can monitor the survival of nodes at any time. Once a node goes down, it will be replaced immediately.

2.3. Low Cost. Cloud computing does not require high-performance computing components and professional storage devices. Users only need to pay a very low rent and a terminal that can connect to the Internet, and they can get high-performance computing power and large-scale storage capacity. Cloud application service providers are responsible for maintaining the software and hardware in the cloud, which greatly saves the purchase and maintenance costs of users’ software and hardware.

2.4. On-Demand Service. Cloud service providers charge rent according to the actual usage of users, which can allocate resources in time to increase the actual service capacity of users when the number of resource requests suddenly increases and can also release the resources occupied by users when the number of resources required by users decreases to avoid wasting resources.

2.5. Scalability. Cloud computing architecture and scale have certain scalability, allowing computing nodes to join and exit computer clusters dynamically, which can not only build a usable high-performance computing platform in a short time but also facilitate data center maintenance personnel to maintain equipment.

2.6. Safe and Reliable. Data center uses virtual machines to schedule tasks and uses the isolation of virtual machines to provide users with good security. At the same time, the redundancy technology of the data center ensures the security of the data stored in the cloud. Even if a certain node or a certain area is paralyzed, the standby node in another area can replace the failed node at the first time.

3. Resource Model and Scheduling Algorithm Model

3.1. Basic Model. The basic model includes the virtual machine used in the algorithm and the host model, in which the virtual machine acquires the resources in the host and receives the tasks to execute. In this paper, the virtual machine set is expressed as Formula (1).

\[ V = \{v_0, v_1, v_2, \ldots, v_{n-1}\}, \]  

where \( V \) represents a virtual machine set, and each element in the set represents a virtual machine; \( N \) is the number of virtual machines, where each virtual machine can be expressed as Formula (2).

\[ v_j = \{v_{j, ID}, v_{j, storage}, v_{j, pes}, v_{j, RAM}\}, \]  

where \( v_j \) represents the \( J \)-th virtual machine; \( v_{j, ID} \) represents the ID of this virtual machine; \( v_{j, storage} \) represents the mirror size of this virtual machine; \( v_{j, pes} \) represents the number of CPU cores (Pe number) owned by the virtual machine; \( v_{j, RAM} \) represents the memory size of this virtual machine.
Each host has certain resources, and the host set is defined as Formula (3).

\[
H = \{h_0, h_1, h_2, \cdots, h_s\},
\]

where \(H\) represents the host set and \(S\) represents the total number of hosts. Each host can be expressed as Formula (4).

\[
h_k = \{h_k^{ID}, h_k^{storage}, h_k^{mips}, h_k^{RAM}, h_k^{band}, h_k^{pes}\},
\]

where \(h_k^{ID}\) represents the ID of this host; \(h_k^{storage}\) represents the storage size of this host, which determines how many virtual machines this host can accommodate; \(h_k^{RAM}\) represents the memory size of this host; \(h_k^{pes}\) represents the number of cores owned by this host.

### 3.2. Resource Request Model.

In order to keep the load of the host stable and reduce the risk caused by excessive load of the host, the virtual machine scheduling algorithm in this paper limits the load to less than 80%, that is to say, it is necessary to limit the amount of virtual machine resource requests. If the virtual machine is placed on a host, so that the resource load of the host exceeds 80%, then the virtual machine will not be placed on the host, and other hosts need to be selected for placement, so the resource request amount of the virtual machine is limited here. The algorithm in this paper improves the host resource limitation of the original OpenStack algorithm. According to Equations (2) and (4), the three sources of host Pe number, memory size, and storage space are limited, as shown in Formula (5).

\[
\begin{align*}
&\sum_{j=0}^{n-1} v_{j, pes} \cdot \text{flag}_{j, k} + v_{s, pes} \leq h_k^{pes} \cdot 80\%, \\
&\sum_{j=0}^{n-1} v_{j, RAM} \cdot \text{flag}_{j, k} + v_{s, RAM} \leq h_k^{RAM} \cdot 80\%, \\
&\sum_{j=0}^{n-1} v_{j, storage} \cdot \text{flag}_{j, k} + v_{s, storage} \leq h_k^{storage} \cdot 80%,
\end{align*}
\]

where \(v_{s, pes}\), \(v_{s, RAM}\), and \(v_{s, storage}\) denote various resource values of virtual machine \(v_s\) to be placed on host \(h_k\), and \(0 \leq x \leq n-1\), and \(\text{flag}_{j, k}\) is defined as Formula (6).

\[
\begin{align*}
\text{flag}_{j, k} &= 0, \text{ virtual machine } v_j \text{ not placed on host } h_k, \\
\text{flag}_{j, k} &= 1, \text{ virtual machine } v_j \text{ not placed on host } h_k.
\end{align*}
\]

The above Formulas (5) and (6) indicate that when creating a virtual machine in host \(h_k\), the corresponding sum of the various types of resources already allocated thereto plus the value of the corresponding resources to be allocated thereto cannot exceed 80% of the total value of the corresponding resources that this host can accommodate. This filtering can be called Filter filtering.

### 3.3. Host Resource Load Model.

When a virtual machine is created, Pe resources, memory resources, and storage resources on the host need to be used. This will cause the host to produce the corresponding resource load. Therefore, this paper will consider the Pe load, memory load, and virtual machine storage load of the host here. The load model can be understood as follows: the corresponding resources that have been used are higher than the total corresponding resources, and the load model is established as shown in Formulas (7)–(10).

\[
\begin{align*}
\text{Pe}_{j, k, \text{load}} &= \frac{\sum_{j=0}^{n-1} v_{j, pes} \cdot \text{flag}_{j, k}}{h_k^{pes}} \cdot 100\%, \\
\text{RAM}_{j, k, \text{load}} &= \frac{\sum_{j=0}^{n-1} v_{j, RAM} \cdot \text{flag}_{j, k}}{h_k^{RAM}} \cdot 100\%, \\
\text{Storage}_{j, k, \text{load}} &= \frac{\sum_{j=0}^{n-1} v_{j, storage} \cdot \text{flag}_{j, k}}{h_k^{storage}} \cdot 100\%, \\
\text{h}_{k, \text{average, load}} &= \frac{\text{Pe}_{j, k, \text{load}} + \text{RAM}_{j, k, \text{load}} + \text{Storage}_{j, k, \text{load}}}{3} \cdot 100\%,
\end{align*}
\]

where \(\text{Pe}_{j, k, \text{load}}\), \(\text{RAM}_{j, k, \text{load}}\), \(\text{Storage}_{j, k, \text{load}}\), and \(\text{h}_{k, \text{average, load}}\) represent the Pe number load, memory load, virtual machine storage load, and the average load of host \(h_k\), respectively. It can be seen that the more virtual machines placed on a host, the greater its load value. Therefore, it is necessary to choose the placement of virtual machines reasonably to keep the load of various resources of the host in a stable range.

### 3.4. Scheduling Target Model.

After building the above model, we also need to build a scheduling target model. The scheduling target model is the ultimate goal of the algorithm developed in this paper. And it is also an important index to evaluate the performance gap between the two algorithms. The goal of this resource-sensitive virtual machine scheduling algorithm is to balance the resource load of each host as much as possible. That is to say, you cannot overload or underload resources on individual hosts. To minimize the difference between the resource loads of all hosts and keep the resource loads of all hosts at a balanced level, in order to achieve this goal, we first need to know the average load of all hosts, which can be obtained according to Equation (10), and then, we can calculate the load variance and load standard deviation of all hosts. In addition, because we have limited Formula (5) for resource requests of virtual machines, therefore, it is necessary to analyze the success rate of creating virtual machines in all kinds of hosts, which can show the satisfaction of scheduling algorithm to the request of creating virtual machines (the higher the success rate, the better the satisfaction). Therefore, this paper establishes Formulas (11)–(14).

\[
\text{h}^{\text{average, load}} = \frac{\sum_{k=0}^{n-1} \text{h}^{\text{average, load}}}{s} \cdot 100%,
\]

The above Formulas (5) and (6) indicate that when creating a virtual machine in host \(h_k\), the corresponding sum of the various types of resources already allocated thereto plus the value of the corresponding resources to be allocated thereto cannot exceed 80% of the total value of the corresponding resources that this host can accommodate. This filtering can be called Filter filtering.
\[
\text{Variance}_{\text{host}} = \frac{1}{n} \sum_{j=0}^{n-1} (h_{k, \text{average load}} - h_{k, \text{all average load}})^2,
\]
(12)

\[
\text{Std}_{\text{host}} = \sqrt{\text{Variance}_{\text{host}}},
\]
(13)

\[
\text{Ratio}_{\text{vm success}} = \frac{\text{Num}_{\text{vm success}}}{n} \cdot 100\%.
\]
(14)

In the above formula, \(h_{k, \text{average load}}\) represents the average load of all hosts, \(\text{Variance}_{\text{host}}\) represents the load variance of all hosts, \(\text{Std}_{\text{host}}\) represents the load standard deviation of all hosts, and \(\text{Ratio}_{\text{vm success}}\) represents the success rate of creating virtual machines. The load dispersion of each host can be clearly expressed by variance and standard deviation.

3.5. Host Estimates Remaining Resources. According to Formula (2), the resource requests of virtual machines are generally PE, RAM, and storage, which will not be introduced here. As for the estimated remaining resources of the host, it represents the resource remaining situation of the host calculated by the algorithm when the virtual machine is scheduled, which is the estimated remaining resources of the host. This paper improves the definition of remaining resources in the original algorithm of OpenStack (the original algorithm only considers the virtual machine resources created by the host species). An estimation mechanism is added, which considers not only the created virtual machine in the host, but also the resource request of the virtual machine to be created. According to the Formulas (2), (4), and (6), the following Formula (15) is constructed.

\[
\begin{align*}
\text{h}_k\text{ free}_{\text{PE}} &= \text{h}_k\text{ Pe} - \sum_{j=0}^{n-1} v_j\text{ Pe} \cdot \text{flag}_{jk} - v_{x}\text{ Pe}, \\
\text{h}_k\text{ free}_{\text{RAM}} &= \text{h}_k\text{ RAM} - \sum_{j=0}^{n-1} v_j\text{ RAM} \cdot \text{flag}_{jk} - v_{x}\text{ RAM}, \\
\text{h}_k\text{ free}_{\text{storage}} &= \text{h}_k\text{ storage} - \sum_{j=0}^{n-1} v_j\text{ storage} \cdot \text{flag}_{jk} - v_{x}\text{ storage},
\end{align*}
\]
(15)

where \(\text{h}_k\text{ free}_{\text{PE}}\), \(\text{h}_k\text{ free}_{\text{RAM}}\) and \(\text{h}_k\text{ free}_{\text{storage}}\) respectively, represent the estimated remaining PE, estimated remaining memory and estimated remaining storage of host \(h_k\), and their values are equal to the total corresponding resources owned by this host minus the corresponding resources of all virtual machines created on it and then minus the corresponding resources of virtual machines to be placed on it. Combined with Equation (5), it can be seen that the more remaining resources the host estimates, the more likely the scheduling algorithm is to put virtual machines on it.

3.6. Predicted Average Weight of Host Computer. In this paper, the estimated residual resources are used to calculate the corresponding estimated weights. Firstly, the maximum estimated residual resources and the minimum estimated residual resources of each resource need to be calculated. Then, the maximum estimated residual resource value and the minimum estimated residual resource value are used to calculate the estimated residual resource difference of each resource. Finally, using the estimated residual resource difference, the host residual resource value and the minimum estimated residual resource value, the estimated weight value corresponding to each resource is calculated, and the estimated weight value is added to take the average as the final estimated average weight value of the host.

(1) Calculate the estimated maximum residual resource value and the estimated minimum residual resource value of each resource. The purpose of calculating this value is to obtain the available residual resources on the host, so as to judge which host is more suitable for creating a new virtual machine. According to Formula (15), the following Formula (16) is defined as

\[
\begin{align*}
\text{max}_{\text{free PE}} &= \text{Max}\{h_0\text{ free}_{\text{PE}}\}, h_{j=1}\text{ free}_{\text{PE}}
\end{align*}
\]
\[
\begin{align*}
\text{min}_{\text{free PE}} &= \text{Min}\{h_0\text{ free}_{\text{PE}}\}, h_{j=1}\text{ free}_{\text{PE}}
\end{align*}
\]
\[
\begin{align*}
\text{max}_{\text{free RAM}} &= \text{Max}\{h_0\text{ free}_{\text{RAM}}\}, h_{j=1}\text{ free}_{\text{RAM}}
\end{align*}
\]
\[
\begin{align*}
\text{min}_{\text{free RAM}} &= \text{Min}\{h_0\text{ free}_{\text{RAM}}\}, h_{j=1}\text{ free}_{\text{RAM}}
\end{align*}
\]
\[
\begin{align*}
\text{max}_{\text{free storage}} &= \text{Max}\{h_0\text{ free}_{\text{storage}}\}, h_{j=1}\text{ free}_{\text{storage}}
\end{align*}
\]
\[
\begin{align*}
\text{min}_{\text{free storage}} &= \text{Min}\{h_0\text{ free}_{\text{storage}}\}, h_{j=1}\text{ free}_{\text{storage}}
\end{align*}
\]
(16)

where \(\text{max}_{\text{free PE}}\) denotes an estimated maximum value of the remaining PE resource, \(\text{min}_{\text{free PE}}\) denotes an estimated minimum value of the remaining PE resource, \(\text{max}_{\text{free RAM}}\) denotes an estimated maximum value of the remaining memory resource, \(\text{min}_{\text{free RAM}}\) denotes an estimated minimum value of the remaining memory resource, \(\text{max}_{\text{free storage}}\) denotes an estimated maximum value of the remaining memory resource, and \(\text{min}_{\text{free storage}}\) denotes an estimated minimum value of the remaining memory resource.

(2) Calculating the estimated residual resource difference of each resource (which is equal to the estimated maximum residual resource value minus the estimated minimum residual resource value), the purpose of calculating the value is to obtain the difference between the estimated residual resources of all hosts and calculate the estimated weight value of each host accordingly. According to Formula (16), Formula (17) is defined as

\[
\begin{align*}
\text{range}_{\text{PE}} &= \text{max}_{\text{free PE}} - \text{min}_{\text{free PE}}, \\
\text{range}_{\text{RAM}} &= \text{max}_{\text{free RAM}} - \text{min}_{\text{free RAM}}, \\
\text{range}_{\text{storage}} &= \text{max}_{\text{free storage}} - \text{min}_{\text{free storage}},
\end{align*}
\]
(17)
where range\textsubscript{pes} represents the estimated remaining Pe resource difference of all hosts, range\textsubscript{RAM} represents the estimated remaining memory resource difference of all hosts, and range\textsubscript{storage} represents the estimated remaining storage resource difference of all hosts, which can be used to calculate the estimated average weight of each host.

(3) This paper improves the calculation method of host weights in OpenStack’s original algorithm (the original algorithm only considers the weights of a single resource), calculates the estimated average weights of the whole host by using the weights of multiple resources, and defines Formula (18) according to Formulas (15)–(17).

$$h_{jk}\text{average\ weight} = \frac{1}{3} \left( \frac{h_{free\ pes} - \min h_{free\ pes}}{\text{range}_{jk\ pes\ weight}} + \frac{h_{free\ RAM} - \min h_{free\ RAM}}{\text{range}_{jk\ RAM\ weight}} \right) + \frac{h_{free\ storage} - \min h_{free\ storage}}{\text{range}_{jk\ storage\ weight}}$$

where $h_{jk, average\ weight}$ represents the estimated average weight generated by host $h_k$ when creating virtual machine $v_j$. The estimated weight value of each host is equal to the difference between the estimated remaining resources and the estimated minimum remaining resources and the estimated remaining resources, and then, the estimated weights of the three resources are added and averaged to obtain the estimated average weight, and the one with the highest estimated average weight is more likely to create virtual machines. The purpose of this is to reduce the impact of excessive resources on the weight of the end host and make the weight consideration more diversified.

4. Experiment of Virtual Machine Placement Algorithm

4.1. Experimental Simulation Environment. In order to test the performance of the algorithm, 10000 MIPS CPU, 50 GB memory, 1 TB storage, and 10 G bandwidth are set up. Specific settings of virtual machine and algorithm parameters are shown in Tables 1 and 2.

In order to analyze the performance of the algorithm, the virtual machines with loads of 100, 200, 300, 400, and 500 are run 10 times, respectively, and the average value of 10 times is taken as the final data of experimental evaluation.

4.2. Experimental Analysis. In order to verify the performance and efficiency of IGGA proposed in this paper, the virtual machine placement algorithm based on FFD is compared with the traditional virtual machine placement algorithm based on GA. FFD algorithm is a commonly used heuristic algorithm, which is generally used to solve the problem of virtual machine placement, while GA algorithm is the basis of improved genetic algorithm, and is often used to evaluate the performance of improved algorithm.

As shown in Figures 1 and 2, under the same virtual machine request scale, IGGA algorithm is obviously superior to FFD algorithm and GA algorithm in terms of the number of physical machines activated and the comprehensive utilization rate of resources. Compared with the other two algorithms, the number of physical machines activated is reduced by about 13% and 7%, respectively, and the resource utilization rate is increased by about 30% and 14%. Moreover, the number of activated physical machines is closely related to the request scale of virtual machines. At the same time, the request scale of virtual machine also affects the comprehensive utilization of resources, which fluctuates in a certain range.

As shown in Figure 3, IGGA algorithm is superior to FFD algorithm and GA algorithm in system energy consumption, and the energy consumption is reduced by about 12% and 6%, respectively. It has the least number of physical machines, the highest comprehensive utilization rate of resources, and the minimum energy consumption required by the system. Therefore, the less the number of physical machines started by the system, the higher the resource utilization rate, and then, the energy consumption of the system can be reduced.

As shown in Figure 4, it reflects the fluctuation of load imbalance of the three algorithms under different virtual machine sizes. Under the same virtual machine request size, IGGA algorithm is less unbalanced than FFD algorithm and GA algorithm, which shows that the improved algorithm is better than them in load balancing performance. The performance of FFD algorithm is the worst in load balancing and energy consumption. The algorithm reduces the number of active physical machines to a great extent, which makes the load of individual hosts too high, and the unbalanced load leads to waste of resources and energy consumption. Traditional GA algorithm has the disadvantages of high redundancy and repeated coding, resulting in poor performance of the algorithm. The algorithm proposed in this paper is optimized in energy consumption and load

| Virtual machine type | CPU (MIPs) | Memory (GB) | Storage (GB) | Bandwidth (G) |
|----------------------|-----------|-------------|--------------|---------------|
| 1                    | 1000      | 4           | 20           | 1             |
| 2                    | 2000      | 8           | 50           | 2             |
| 3                    | 3000      | 16          | 100          | 2             |
| 4                    | 5000      | 24          | 200          | 4             |

Table 2: Setting algorithm parameters.

| Algorithm parameters | Value |
|----------------------|-------|
| Population size $S$  | 50    |
| Number of iterations $G$ | 100 |
| Crossing rate $R_c$  | 0.8   |
| Variation rate $R_m$ | 0.02  |
balancing and improves the energy efficiency of the data center as a whole.

How to improve the overall energy efficiency of data center is a problem worthy of people’s attention. By establishing the equilibrium index and combining with the traditional algorithm, the genetic algorithm proposed in this paper can be better optimized. Experiments show that the optimized algorithm can minimize data energy consumption.

4.3. Experimental Simulation Environment. The experiment extends the cloud simulation platform and compares it with set-top box.

In the experiment, 500 physical machines are selected to form a data center. Each physical machine contains four resources with the same performance to verify the algorithm. The experiment also sets virtual machine parameters, specific data center, and algorithm, as shown in Tables 3 and 4.

The data processing method is the same as that in Tables 1 and 2.

4.4. Experimental Results and Analysis. In order to build a green data center, we should try to avoid the adverse effects of virtual machine migration, and this algorithm also considers minimizing unnecessary migration of virtual machines. In the algorithm, the statistics of migration times of virtual machines are mainly based on the situation that virtual machines are not in the source physical machines after integration.

As shown in Figure 5, the migration number of virtual machines in different virtual machine sizes is compared between DTMA algorithm and STMA algorithm, and both algorithms set resource load threshold intervals of 0.4 and 0.8. Compared with DTMA algorithm, the number of virtual machines migrated in STMA algorithm fluctuates greatly, which is due to the frequent migration of virtual machines caused by the changing load of physical machines reaching instantaneous peak and valley values. However, DTMA algorithm is lower than STMA algorithm in the number of virtual machine migration under all sizes of virtual machines, which is reduced by about 19%. This is because DTMA algorithm considers the overall load of the data center, dynamically changes the threshold interval of resource load, and deals with unnecessary migration caused by instantaneous peak and valley according to the load of the data center.

The system energy consumption is the sum of the energy consumption of all physical machines in the data center during the system operation, as shown in Figure 6, which reflects the energy consumption of DTMA algorithm and STMA algorithm under different virtual machine scales, and the system energy consumption increases with the increase of the number of virtual machines. Under the same
In this paper, a virtual threshold migration algorithm is proposed to solve the problem of unreasonable use of resources. This method can predict load, select migration opportunity, object, and destination physical machine, dynamically changes the threshold value, and migrate in combination with policies, thus reducing the number of servers, thus reducing the power consumption of the data center to the lowest. Through simulation experiments, this method can consider the load change of the whole data center and the dynamic adjustment of threshold interval, avoid unnecessary migration times of virtual machines, and play an important role in reducing energy consumption of data center.

This chapter mainly compares and analyzes the simulation results. Firstly, the simulation results of server number, energy consumption, and load imbalance between the proposed virtual machine placement algorithm and other algorithms are compared and analyzed, and then, the simulation results of virtual machine migration times and energy consumption between the proposed virtual machine migration algorithm and other algorithms are compared and analyzed, respectively.

### 5. Conclusion

In recent years, virtual machine scheduling based on virtual machines has become an important supporting technology for power management and dynamic expansion of cloud servers. Virtual machine resource scheduling problems can be divided into two categories: deployment and migration. Virtual machine deployment is the process of checking and selecting the appropriate physical target machine from the empty cloud service center according to the virtual machine type and resource request, which has been proved to be a NP-hard problem, similar to the compression problem. In the virtual machine placement stage, the virtual machine placement mainly adopts the corresponding greedy or heuristic virtual machine placement strategy, which is difficult to ensure the balanced utilization of resources and lacks consideration of the data center load type. The purpose of migration is to study the process of data change. The whole process must be handled according to the load of the computer room. In the phase of virtual machine migration, we need to consider the following aspects, choose the migration timing, object, and target physical machine, and adjust the source physical machine according to the load.

### Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

### Conflicts of Interest

The authors declared that they have no conflicts of interest regarding this work.

### Acknowledgments

This work was sponsored in part by the 2020 Teaching Research and Scientific Research Project of Jiuzhou Polytechnic (Yjx202003).
References

[1] J. Hu, J. Gu, G. Sun, and T. Zhao, “A scheduling strategy on load balancing of virtual machine resources in cloud computing environment,” in 2010 3rd International symposium on parallel architectures, algorithms and programming, vol. 3no. 7, pp. 432–468, Liaoning, China, Dec. 2010.

[2] L. I. Jian-Feng and J. Peng, “Task scheduling algorithm based on improved genetic algorithm in cloud computing environment,” Journal of Computer Applications, vol. 31, no. 1, pp. 184–186, 2011.

[3] A. Fernández, S. del Río, V. López et al., “Big data with cloud computing: an insight on the computing environment, Map Reduce, and programming frameworks [J]. Wiley Interdisciplinary Reviews,” Data Mining & Knowledge Discovery, vol. 4, no. 5, pp. 380–409, 2014.

[4] R. Jeyarani, R. V. Ram, and N. Nagaveni, "Design and implementation of an efficient two-level scheduler for cloud computing environment," in IEEE/ACM International Conference on Cluster, vol. 32no. 18, pp. 189–264, Kottayam, India, Oct. 2009.

[5] J. Gu, J. Hu, T. Zhao, and G. Sun, “A new resource scheduling strategy based on genetic algorithm in cloud computing environment,” Journal of Computers, vol. 7, no. 1, pp. 42–52, 2012.

[6] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, “Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment,” Journal of Software, vol. 99, no. 7, pp. 20–35, 2015.

[7] M. Katyal and A. Mishra, “A comparative study of load balancing algorithms in cloud computing environment,” Computer Science, vol. 6, no. 1, pp. 25–36, 2014.

[8] X. Y. Hua, J. Zheng, and H. U. Wen-Xin, “Ant colony optimization algorithm for computing resource allocation based on cloud computing environment,” Journal of East China Normal University (Natural Science), vol. 3, no. 2, pp. 120–192, 2010.

[9] X. Kai, T. Yan, and J. Hu, "Correlation keystroke verification scheme for user access control in cloud computing environment," Computer Journal, vol. 54, no. 10, pp. 1632–1644, 2011.

[10] H. Wei, Z. Wen, and C. Chu, “Virtual machine scheduling algorithm based on K-means clustering in cloud computing,” Journal of Nanjing University of Science & Technology, vol. 37, no. 6, pp. 807–812, 2013.

[11] P. Xiao, D. B. Liu, and X. L. Qu, "An virtual machine scheduling algorithm based on energy-consumption ratio model in cloud computing," Acta Electronica Sinica, vol. 43, no. 2, pp. 305–311, 2015.

[12] R. S. Chang, Y. C. Chang, and R. C. Ye, “A virtual machine scheduling algorithm for resource cooperation in a private cloud,” in Computer Science and its Applications, no. 7 pp. 207–215, Springer, Dordrecht, 2012.

[13] X. Xu, Q. Zhang, S. Maneas, S. Sotiriadis, C. Gavan, and N. Bessis, "VMSAGE: a virtual machine scheduling algorithm based on the gravitational effect for green cloud computing," Simulation Modelling Practice and Theory, vol. 93, pp. 87–103, 2019.

[14] B. Liang, X. Dong, and X. Zhang, "A heuristic virtual machine scheduling algorithm in cloud data center," in 2019 IEEE 3rd information technology, networking, electronic and automation control conference (ITNEC), vol. 42no. 56, pp. 239–310, Chengdu, China, March 2019.

[15] H. Shu, J. B. Yang, and H. J. Ding, "Research on GA-DPSO virtual machine scheduling algorithm based on comprehensive utilization of host resource," Advanced Materials Research, vol. 791, pp. 1373–1376, 2013.