Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8769674, 13 pages
https://doi.org/10.1155/2022/8769674

Research Article

An Energy-Saving Task Scheduling Model via Greedy Strategy under Cloud Environment

Shuaishuai Liu,[1] Xinyu Ma,[1] Yuanfei Jia,[2] and Yue Liu[3]

[1]*School of Computer Science and Information Engineer, Harbin Normal University, Harbin 150025, China*
[2]*Maple Leaf International School-Jinan, Jinan 250000, China*
[3]*Navigation College, Dalian Maritime University, Dalian 116026, China*

Correspondence should be addressed to Yue Liu; liu2020yue@126.com

Received 7 February 2022; Revised 15 March 2022; Accepted 29 March 2022; Published 15 April 2022

Academic Editor: Yinghua Shen

Cloud computing, an emerging computing paradigm, has been widely concerned due to its high scalability and availability. An essential stage of cloud computing is cloud resource management. Currently, the existing research about cloud computing technology has two prevalent disadvantages: high energy consumption and low resource utilization. Considering greedy scheduling is an effective strategy for cloud resource management technology in cloud computing, particularly in improving resource utilization and reducing energy consumption, we consider the heterogeneous characteristics of resources to save energy consumption of datacenter when tasks are the fundamental element of cloud datacenter. Meanwhile, granular computing is a complex problem-solving strategy through a granulation method. Thus, we introduce granular computing theory into cloud task scheduling and propose a greedy scheduling strategy based on different information granules, dividing the tasks into three types (i.e., CPU, memory, and hybrid type). Finally, we assign various scheduling strategies for cloud tasks with different characteristics. All the numerical experiments on the CloudSim platform show that our method has significant effects on energy consumption optimization and is a practical task scheduling algorithm.

1. Introduction

Cloud computing is an emerging distributed computing paradigm, which can be regarded as a pool of accessible and virtualized resources via Internet technology. This relatively new computing paradigm has been widely concerned and successfully applied in many engineering fields, such as big data analysis, medical diagnosis, and financial transactions [1]. Cloud task scheduling is an integral part of cloud resource management, which can directly affect the overall performance of cloud datacenter [2]. Thus, how to carry out efficient task scheduling has attracted the attention of many researchers. Currently, as the number of cloud computing customers increases, scheduling becomes quite tricky, requiring the use of appropriate scheduling algorithms. Some scheduling algorithms are developed in the early stages under the grid computing environment [3].

Recently, in cloud computing, where users may use hundreds or thousands of virtualized resources, everyone cannot assign every task manually. Due to commercialization and virtualization and the complexity of cloud computing in handling task scheduling at the virtual machine layer, scheduling plays a critical role in the cloud datacenter, requiring efficient and effective allocation of resources to each task. Decker et al. [4] use two granular computing methods, namely, fuzzy-set-based evolving modeling (FBeM) and evolving granular neural network (eGNN) to model and monitor data. A progressive granular neural network (PGNNs) can improve the classifier performance [5]. Shi et al. [6] propose a task duplication and insertion algorithm based on list scheduling to dynamically schedule tasks by predicting the completion time. An improved whale optimization algorithm (IWC) in paper [7] can effectively prove task scheduling efficiency.

According to the above discussion, we can see that some exiting works only consider the task scheduling optimization or the multitask scheduling problem, but they ignore various types of task impacts on the scheduling. Moreover, in the application of actual hardware of a big scale, the slight adjustment is likely to bring enormous impact meter platform of the cloud for the hardware. Thus, this paper considers the multitype task scheduling problems and proposes a cloud energy-saving scheduling strategy. The basic idea of our method is to divide resource requests into CPU type, memory type, and hybrid type according to the three-way decision theory.

Tasks have many attributes, but many types are not suitable to cluster, and the most significant impact on scheduling is CPU and memory. So, we consider the three indicators (i.e., CPU, memory, and hybrid type). Meanwhile, combined with different scheduling strategies, tasks with different characteristics are placed on more suitable computing resources to achieve the effect of energy-saving.

The remainder of this paper is organized as follows. Section 2 reviews the basic notions of three-way decision and the cloud task scheduling framework. Section 3 gives a cloud task clustering method based on three-way decision (TWD-CTC), which adopts the corresponding energy-saving scheduling strategy. Section 4 gives the experimental analysis results. Finally, this paper concludes with a suggestion for future research in Section 5.

2. Preliminaries

2.1. Three-Way Decision in Cloud Computing. Three-way decision is an effective data-information-knowledge-wisdom (DIKW) processing theory proposed by Yao [8], whose essential elements include trisecting, acting, and outcome, i.e., TAO model. Three-way decision has been widely used in various disciplines such as computer science, management science, mathematical science, and decision science [9–11]. We introduce the three-way decision into cloud task scheduling, and it is necessary to introduce the basic notions of three-way decision.

Definition 1 (see [12]). Let U be a finite nonempty subset and R be an equivalence relation on U. A pair of $\mathrm{apr}_{(\alpha,\beta)} = (U, R)$ is called an approximate space of fuzzy probabilistic rough set. For $X \subseteq U$ and $0 \le \beta < \alpha \le 1$, the upper and lower approximate set can be defined as follows:

$$\begin{aligned} \underline{\mathrm{apr}}_{(\alpha,\beta)}(X) &= \{x \in U \mid \Pr(X|[x]) \ge \alpha\}, \\ \overline{\mathrm{apr}}_{(\alpha,\beta)}(X) &= \{x \in U \mid \Pr(X|[x]) > \beta\}, \end{aligned} \tag{1}$$

where $\Pr(X|[x]) = |[x] \cap X| / |[x]|$ denotes the conditional probability of classification and $|\cdot|$ denotes the cardinality of the set.

Given a pair of thresholds (a,b), U will be divided into the positive region $\mathrm{POS}_{(\alpha,\beta)}(X)$, boundary region $\mathrm{BND}_{(\alpha,\beta)}(X)$, and negative region $\mathrm{NEG}_{(\alpha,\beta)}(X)$, and we call it trisection of three-way decision, which is defined as follows:

$$\begin{aligned} \mathrm{POS}_{(\alpha,\beta)}(X) &= \{x \in U \mid \Pr(X|[x]) \ge \alpha\}, \\ \mathrm{BND}_{(\alpha,\beta)}(X) &= \{x \in U \mid \beta < \Pr(X|[x]) < \alpha\}, \\ \mathrm{NEG}_{(\alpha,\beta)}(X) &= \{x \in U \mid \Pr(X|[x]) \le \beta\}. \end{aligned} \tag{2}$$

According to decision-theoretic rough set, the rule generated from $\mathrm{POS}(X)/\mathrm{POS}_{(\alpha,\beta)}(X)$ represents object x that belongs to X; the rule generated from $\mathrm{NEG}(X)/\mathrm{NEG}_{(\alpha,\beta)}(X)$ represents object x that does not belong to X; the rule generated from the$\mathrm{BND}(X)/\mathrm{BND}_{(\alpha,\beta)}(X)$ represents uncertain object x that belongs to X.

Definition 2 (see [13]). Let U be a finite nonempty universal set and$S$ be a finite standard set, which divide the U into three pairwise disjoint subsets, POS, BND, and NEG, which are denoted as $\pi = \{\mathrm{POS}, \mathrm{BND}, \mathrm{NEG}\}$ and the following statement hold:

(1) Pairwise disjoint: $\mathrm{POS} \cap \mathrm{BND} = \varnothing$, $\mathrm{POS} \cap \mathrm{NEG} = \varnothing$, and $\mathrm{BND} \cap \mathrm{NEG} = \varnothing$

(2) Covering the universal set: $\mathrm{POS} \cup \mathrm{BND} \cup \mathrm{NEG} = U$

For these subsets, the complement is constructed as follows:

$$\begin{aligned} \mathrm{POS}^C &= \mathrm{BND} \cup \mathrm{NEG}, \\ \mathrm{BND}^C &= \mathrm{POS} \cup \mathrm{NEG}, \\ \mathrm{NEG}^C &= \mathrm{POS} \cup \mathrm{BND}. \end{aligned} \tag{3}$$

As a decision theory, the three-way decision model provides an incremental thinking method to solve complex problems. In the first stage, the universe will be divided into the three reasonable regions. In the second stage, the optimal strategy is formulated according to the three regions, and the different strategies will be applied in these regions [14, 15].

With the in-depth study of cloud computing, it is easy to find that there are many phenomena about "3" in cloud computing, which can be called the three elements of cloud computing. For example, the task time can be represented as three parts: the long-time, medium-time, and short-time tasks; the virtual machine operation can be represented as three parts: merging, migrating, and shutting; the host state can be represented as three parts: working, sleeping, and shutting. Note that with so many phenomena about "3," it is not difficult to assume there exists a granular computing model based on a three-way decision under the cloud computing environment. We draw on the basic ideas of the three-way decision and many theoretical achievements to study some interesting cloud computing problems in this paper.

The main idea of three-way decision is to divide the whole into three independent parts and apply different processing methods to the different parts, and it also provides an effective strategy for solving complex problems. This paper proposes an energy-saving task scheduling model based on three-way decision with greedy strategy. The cloud tasks will be divided into CPU type, memory type, and hybrid type by the characteristics of resource requests, and different scheduling strategies are adopted for the different types of tasks to save energy consumption.

2.2. Cloud Task Scheduling. Task scheduling is one of the important components of cloud computing, which allocates tasks with appropriate computing resources for execution reasonably. The effect of scheduling strategy directly affects the effectiveness of the entire cloud computing system, including but not limited to operational efficiency, stability, flexibility, user service quality, system load balancing, and system energy consumption. Figure 1 shows the process of task scheduling in cloud computing.

As shown in Figure 1, the steps of task scheduling are shown as follows: at first, users submit tasks by their resource requirements, and tasks will be added to the task queue. The task scheduler obtains the available resource information mainly on virtual machines through scheduling technology; then, tasks are reasonably allocated to appropriate virtual machines to execute by scheduling strategy; finally, running the tasks on the virtual machine and after the tasks are completed, it will be summarized and feedback to the users.

3. Proposed Algorithm

This section presents a greedy scheduling model with saving energy consumption based on the characteristics of cloud tasks.

3.1. Cloud Task Scheduling System. The architecture diagram of the task scheduling system is shown in Figure 2, in which VM denotes virtual machine. The structure diagram gives the process of the cloud task scheduling system from the perspective of three-way decision and analyzes the user requests by historical data, including data structure, data attributes, and amount of data. The system data preprocessing module filters irrelevant attributes and normalizes valuable raw data. In order to achieve effective task scheduling, the tasks will be divided into three types (CPU type, memory type, and hybrid type) by the data characteristics and threshold, and the corresponding scheduling strategies are applied in different types. The tasks with different attributes will be placed in the more suitable host.

Alam et al. [16] pointed out that tasks have three modes for clustering problems about tasks. According to the resource requests and the number of requests of users, tasks can be represented as three parts: long-time, short-time, and medium-time tasks by clustering. Long-time tasks have more resource requests and fewer occurrences, but they always request for more resources. In particular, there are many requests for CPU resources, so they are called the CPU-intensive tasks; short-time tasks have fewer resource requests, but they have the most frequent occurrences; medium-time tasks have the middle number of occurrences and resource requests, and most requests are memory resource requirements, so they are called the memory-intensive tasks.

This paper uses a mix of multiple job types to cluster all tasks. The task dimensions have requests for CPU resources, memory resources, task waiting time, etc. Zhang et al. [17] pointed out that task resource usage is a more reliable metric than the task waiting time. So, CPU and memory resource requests are chosen as task dimensions. On the other hand, task type and priority are also essential considerations. However, there are nine types of tasks, and the details between types are complicated. Da et al. [18] pointed out the completion status of the tasks: 73% of tasks can be completed normally, 26% of tasks are terminated, less than 1% of tasks are in other states, and almost all tasks cannot be completed in the terminated state. Since the state of tasks is different, some tasks cannot be paused, postponed, or stopped midway once they are started, such as timers. There are 12 priorities, and too many priorities are not suitable for analysis and clustering. Finally, the basic dimensions of the task are summarized into two: CPU resource request and memory resource request.

3.2. The Cloud Task Clustering Based on Three-Way Decision. In this paper, tasks will be divided into three types via three-way decision: CPU type, memory type, and hybrid type. Given $T = \{t_1, t_2, \cdots, t_n\}$ be a task set, where $t_i = \{\text{cpu}, \text{ram}\}$ represents the dimension of task, which includes CPU and memory resources request. Based on the basic idea of three-way decision, the expressions for task clustering is $T = \text{POS}(T) \cup \text{BND}(T) \cup \text{NEG}(T)$, where $\text{POS}(T)$ represents CPU type, $\text{NEG}(T)$ represents memory type, and $\text{BND}(T)$ represents hybrid type, and they satisfy the following properties:

(1) There is no overlap between the three types of tasks: $\text{POS} \cap \text{BND} = \varnothing$,$\text{POS} \cap \text{NEG} = \varnothing$, and $\text{BND} \cap \text{NEG} = \varnothing$

(2) Three types of tasks including all tasks: $\text{POS} \cup \text{BND} \cup \text{NEG} = T$

It is assumed that the selection of iterative centroid is to find the largest value of CPU and the smallest value of the memory in each cluster $x_j (j = 1, 2, \cdots, k)$, which form cluster centroidscentroid$_{x_j} = \{t_{\max\,(\text{cpu})}, t_{\min\,(\text{ram})}\}, (j = 1, 2, \cdots, k)$.

Definition 3. Let t_i^{cpu} be CPU of the task $i\,(i = 1, 2, \cdots, n)$, t_i^{ram} be memory of the task $i\,(i = 1, 2, \cdots, n)$. Given $d(t_i^{\text{cpu}}, \text{centroid}_{x_j}^{\text{cpu}})$ represents the distance between task $i\,(i = 1, 2, \cdots, n)$ in cluster x_j and cluster centroid of CPU attribute, and $d(t_i^{\text{ram}}, \text{centroid}_{x_j}^{\text{ram}})$ represents the distance of memory attribute. α_j represents the average of CPU attribute and β_j represents the average of memory attribute of tasks in cluster x_j. Given the universe of discourse $x_j (j = 1, 2, .., k)$ is a finite nonempty subset and R is an indistinguishable relation on the universe of discourse,$\text{apr} = (x_j, R)$ is the approximation space of rough set. If $x_j \in X$, upper and lower approximate set can be defined as follows:

$$\overline{\text{apr}}(X) = \cup\left\{t_i \in x_j \middle| \|d\|\left(t_i^{\text{cpu}}, \text{centroid}_{x_j}^{\text{cpu}}\right)\Big|_j\right\},$$
$$\underline{\text{apr}}(X) = \cup\left\{t_i \in x_j \|d\|\left(t_i^{\text{cpu}}, \text{centroid}_{x_j}^{\text{cpu}}\right)| < \alpha_j \wedge |d\left(t_i^{\text{ram}}, \text{centroid}_{x_j}^{\text{ram}}\right)| \geq \beta_j\right\}. \tag{4}$$

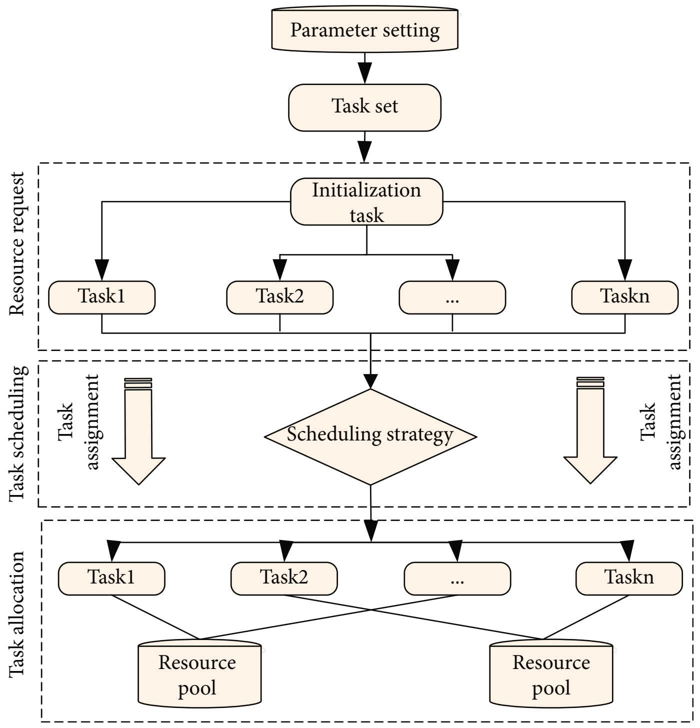

Figure 1: General process of task scheduling in cloud environment.

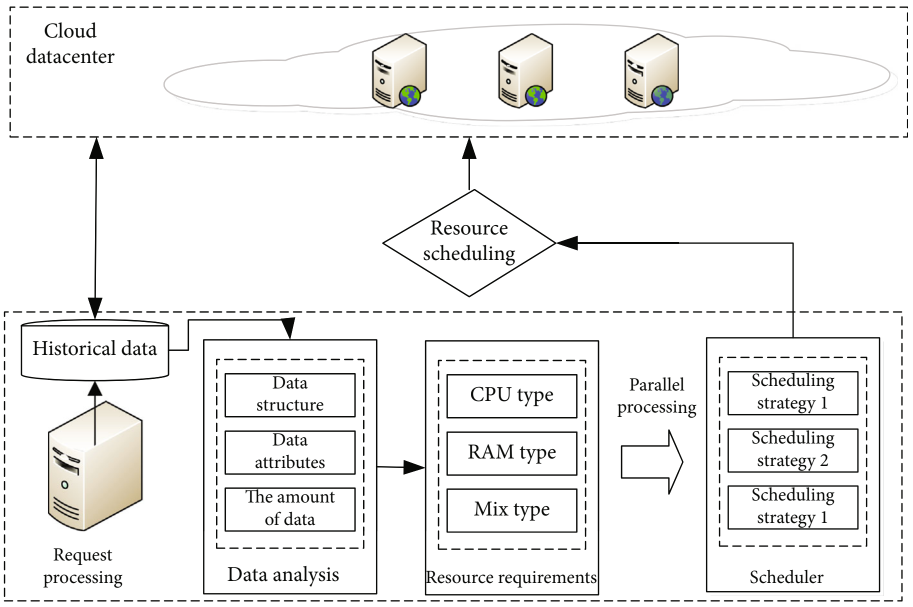

Figure 2: Cloud computing task scheduling architecture diagram.

All object $x_j \in X$ will be divided into core region $\mathrm{POS}(X)$, boundary region $\mathrm{BND}(X)$, and trivial region $\mathrm{NEG}(X)$ based on upper and lower approximate set, which is defined as follows:

$$\mathrm{POS}(X) = \underline{\mathrm{apr}}(X) = \cup\left\{ t_i \in x_j \| d \| \left(t_i^{\mathrm{cpu}}, \mathrm{centroid}_{x_j}^{\mathrm{cpu}} \right) | < \alpha_j \wedge | d \cdot \left(t_i^{\mathrm{ram}}, \mathrm{centroid}_{x_j}^{\mathrm{ram}} \right) | > \beta_j \right\},$$

$$\mathrm{BND}(X) = \overline{\mathrm{apr}}(X) - \underline{\mathrm{apr}}(X) = \cup\left\{ t_i \in x_j \middle| \| d \| \left(t_i^{\mathrm{cpu}}, \mathrm{centroid}_{x_j}^{\mathrm{cpu}} \right) | < \alpha_j \wedge | d \cdot \left(t_i^{\mathrm{ram}}, \mathrm{centroid}_{x_j}^{\mathrm{ram}} \right) \middle| \le \beta_j \right\},$$

$$NEG(X) = 1 - \overline{\mathrm{apr}}(X) = \cup\left\{ t_i \in x_j \middle| \| d \left(t_i^{\mathrm{cpu}}, \mathrm{centroid}_{x_j}^{\mathrm{cpu}} \right) \middle| \ge \alpha_j \right\}. \tag{5}$$

The clustering algorithm firstly carries out k-means on the objects. Then, the tasks of each cluster were divided into three parts, which are core region $\mathrm{POS}(X)$, boundary region $\mathrm{BND}(X)$, and trivial region $\mathrm{NEG}(X)$. The algorithm flow mainly has the following six steps:

(1) Input dataset and number of clusters k

(2) Randomly select k objects in the dataset as cluster centers

(3) Calculate the Euclidean distance between the object and the cluster center and assign the objects to the smallest cluster of Euclidean distance

(4) In each cluster $x_j (j = 1, 2, \cdots, k)$, use the largest CPU value and the smallest memory value as the new cluster center, and return to step 3 until the cluster center no longer change

(5) Calculate the threshold $\alpha_j = 1/|x_j| \sum_{t_i \in x_j} t_i^{\mathrm{cpu}}$ and $\beta_j = 1/|x_j| \sum_{t_i \in x_j} t_i^{\mathrm{ram}}$ for each cluster $x_j (j = 1, 2, \cdots, k)$, and cope with each cluster $x_j (j = 1, 2, \cdots, k)$. According to the upper and lower approximate set conditions of Definition 3, the objects of each cluster are divided into POS region (CPU type), NEG region (memory type), and BND region (hybrid type)

(6) Finally, POS region, NEG region, and BND region of all clusters will be merged

Based on the above discussion in Subsection 3.1, we propose a cloud task clustering method based on three-way decision (TWD-CTC) by analyzing cloud tasks. The specific algorithm is as follows.

```
Input: $T = \{t_1, t_2, \cdots, t_n\}$, the number of clusters $k$
Output: $T = \{\mathrm{POS}(T) \cup \mathrm{NEG}(T) \cup \mathrm{BND}(T)\}$
$\mathrm{centroid}_{x_j} \longleftarrow \mathrm{ChooseCentroid}(T, k)$;
while $\mathrm{centroid}_{x_j}$ is changed do:
        $\mathrm{DivideCluter}(t_{i,\mathrm{centroid}_{x_j}})$;
        $\mathrm{Update}(\mathrm{centroid}_{x_j})$;
endwhile;
foreach $j \in k$ do:
    if $(t_i \in \underline{\mathrm{apr}}(x_j))$ do:
            $\mathrm{POS} = \mathrm{POS} \cup \mathrm{getPOS}(x_j)$;
    else if $(t_i \in (\overline{\mathrm{apr}}(x_j) - \underline{\mathrm{apr}}(x_j)))$ do:
            $\mathrm{BND} = \mathrm{BND} \cup \mathrm{getBND}(x_j)$;
    else if $(t_i \in 1 - \overline{\mathrm{apr}}(x_j))$ do:
        $\mathrm{NEG} = \mathrm{NEG} \cup \mathrm{getNEG}(x_j)$;
    endif;
endfor;
Output: $T = \{\mathrm{POS}(T) \cup \mathrm{NEG}(T) \cup \mathrm{BND}(T)\}$
```

Algorithm 1: TWD-CTC.

3.3. The Energy-Saving Task Scheduling with Greedy Strategy Based on TWD-CTC. This subsection proposes an energy-saving scheduling model with greedy scheduling based on three-way decision.

3.3.1. Greedy Strategy. The greedy task scheduling with energy-saving model is shown in Figure 3 and VM is the virtual machine. The system structure diagram presents a greedy scheduling model based on TWD-CTC from a macroscopic perspective.

The process is as follows:

(1) The different types of resource requests for tasks are divided into core tasks (CPU type), trivial tasks (memory type), and boundary tasks (hybrid type) by cluster

(2) The greedy strategy is used to allocate tasks, and tasks are allocated to VM with optimal resources each time, so as to achieve better overall allocation efficiency

3.3.2. Scheduling Algorithm. The greedy algorithm is efficient and straightforward on finding the optimal solution to solve some problems. The basic idea is first to find the current local optimal solution and then gradually find the optimal global solution to save time to find the optimal solution and get the overall best solution.

The greedy-based task scheduling strategy firstly sorts the tasks in descending order by length of task and sorts VMs in ascending order according to MIPS. Then, calculate the estimated time time$[i][j]$ of task t_i run in VM v_i, and the estimated time will be as the greedy object to iterate for assigning each task to VM with optimal resources. After each allocation, the task scale will be reduced, and each iteration will generate the most suitable resources for the current assigning task. When the iteration is completed, the global task allocation method will be optimal when the iteration is completed.

Based on the above discussion in this subsection, the specific algorithm of the TWD-CTC greedy scheduling model is given as follows.

4. Performance Evaluation

4.1. Experiment Setup. The experiment uses the CloudSim platform designed to help create and manage multiple, independent, and collaborative virtualized services on datacenter nodes and enable flexible switching between time-sharing and space-sharing when allocating processing cores' virtualized services.

We evaluate our algorithm based on the Google traces from [19], which is gathered from Google MapReduce Cloud trace logs, and this version of the cloud computing environment handles information of 25 million tasks that span for nearly one month. Our simulations have been conducted on a computer having Intel® Core™ i7-10750H CPU 2.60GHz 2.59 GHz; 32 GB RAM, and 64-bit Windows 10 Operating System.

The CPU/MIPS of task is [0,1], the Memory/MB of task is [100,500], and the file size/MB of task is [400,1000]; the VmMips of VM is [312,512,800,920,1000,1500]; the Host

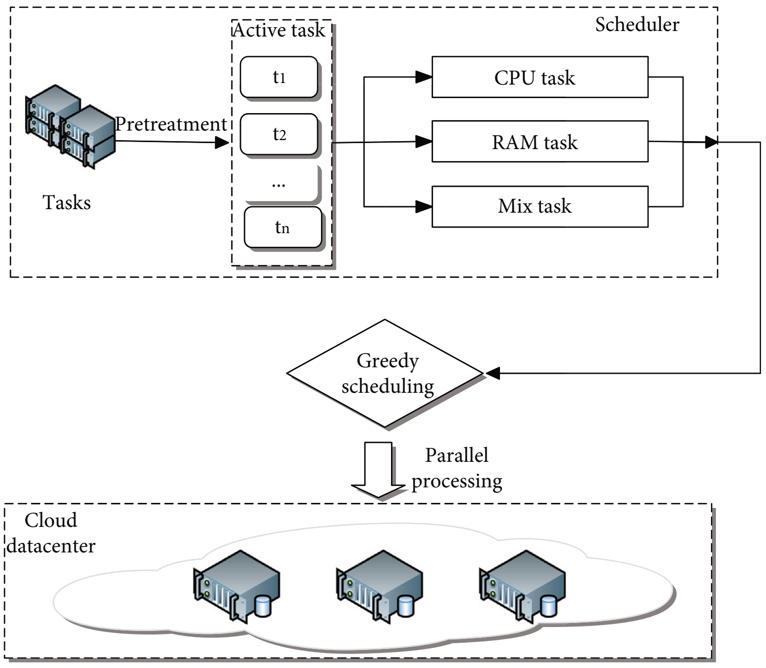

Figure 3: Framework diagram of cloud task greedy scheduling model.

```
Input: Tasks of $n$,VMs of $m$
Output: Total energy consumption of cloud computing $EC$
  SortTask(Tasks,$n$);
  SortVM(VMs,$m$);
  foreach $i \in n$ do :
    foreach $j \in m$ do :
      time$[i][j]$⟵ ComputeTime$(t_i, v_i)$;
  minTime⟵0;
  foreach $i \in n$ do :
    foreach $j \in m$ do :
      if ($v_i$ ⟵ suitbale$(t_i)$ and
  time$[i][j]$ < minTime) do :
      MapList = MapList ∪ Map($vi$, $ti$);
      minTime ⟵ time$[i][j]$;
      endif;
    endfor;
  endfor;
  $EC$⟵ SchedulVmTask (MapList);
Output:$EC$
```

Algorithm 2: TWD-CTCG.

Mips of Host is 3720. The relationship between the host and energy consumption is shown in Figure 4.

The performance of TWD-CTCG algorithm is evaluated by two indicators including energy consumption and load balancing. The comparative experiment implements min-max-min [20], min-max [21], SJF [22], and FCFS [22] task scheduling algorithms. Our algorithm compares them in performance by energy consumption and load balancing.

4.2. Energy Consumption Comparison Experiment

4.2.1. Different Task Experiment Comparison

(1) Less Task Experiment Comparison. In the case of less task experiment, there are 100, 200, 300, 400, and 500 tasks. The compared results of experiments are shown in Table 1 and Figure 5.

From the analysis of the experimental data in Table 1 and Figure 5, when the number of tasks is 100, the cases of min-max and min-max-min are essentially similar and FCFS is relatively higher, and SJF has the lowest energy

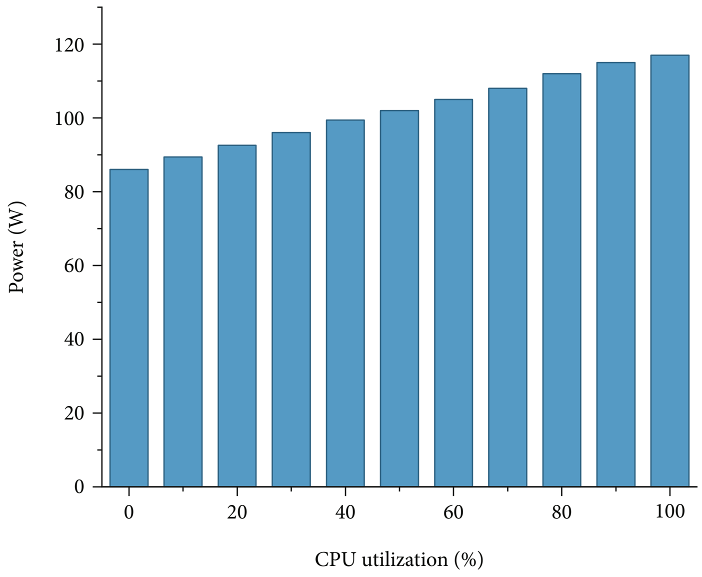

Figure 4: The relationship between host and energy consumption.

consumption, while TWD-CTCG performs slightly worse. As the number of tasks increases, the energy consumption of SJF increases dramatically, and TWD-CTCG has the lowest energy consumption in other tasks type.

(2) More Task Experiment Comparison. In the case of more task experiments, there are 1000, 1250, 1500, 1750, and 2000 tasks. The compared results are shown in Table 1 and Figure 6.

According to the analysis of the experimental data in Table 1 and Figure 6, as the number of tasks increases, the energy consumption generated by the min-max and min-max-min are nearly the same, and SJF generates relatively higher and FCFS has been growing, but the growth has not been remarkable. The TWD-CTCG algorithm increases relatively small in energy consumption.

By comparing the experimental results with less tasks and more tasks, it can be seen that the TWD-CTCG is better than the other algorithms in less task case except for task number is 100 and TWD-CTCG is better than others in more task case.

4.2.2. Different Host Experiment Comparison

(1) Less Host Experiment Comparison. In the case of less host experiment, there are 10, 20, 30, 40, and 50 hosts. The compared results of experiments are shown in Table 2 and Figure 7.

From the analysis of the experimental data in Table 2 and Figure 7, with the number of hosts increasing, the energy consumption generated by min-max-min and min-max is almost the same, while the SJF algorithm is better than FCFS, and in the case of less hosts, TWD-CTCG is better than other algorithms.

(2) More Host Experiment Comparison. In the case of more host experiment, there are 100, 150, 200, 250, and 300 hosts. The compared results are shown in Table 2 and Figure 8.

According to the analysis of the experimental data in Table 2 and Figure 8, with the continuous increase of the number of hosts, the energy consumption generated by min-max-min and min-max is still basically the same in the case of more hosts, but the effect of FCFS is better than that of SJF, indicating that the FCFS algorithm will be better in the case of more hosts, and the effect of TWD-CTCG is best.

By comparing the experimental results with less tasks and more tasks, it can be seen that the algorithm proposed in this paper is better than other algorithms in terms of energy-saving effect and works well in both cases.

4.3. Load Balancing Comparison Experiment

4.3.1. Different Task Experiment Comparison

(1) Less Task Experiment Comparison. Similar to Subsection 4.2, the experiment has 100, 200, 300, 400, and 500 tasks and then compares the load balancing generated by the min-max and min-max-min, SJF, and FCFS algorithms under different numbers of tasks. The compared results are shown in Table 3 and Figure 9.

From the experimental data analysis in Table 3 and Figure 9, in the case of less tasks, the load balancing of the TWD-CTCG algorithm is not well at the beginning, but as the number of tasks increases, the load balancing gradually improves. However, the difference between min-max and min-max-min algorithms of load balancing is not very big and SJF keeps growing, while FCFS grows steadily.

(2) More Task Experiment Comparison. This section is a multitask load balancing comparison experiment, which

Table 1: Energy consumption of different tasks.

Algorithm	Tasks					Tasks				
	100	200	300	400	500	1000	1250	1500	1750	2000
Min-max-min	407498	435538	461987	481285	502318	8135374	8537300	8972264	12048045	12724574
Min-max	406966	434132	459521	478851	493559	8116462	8530405	8929579	12020748	12660304
SJF	175715	466374	765637	501018	947789	3600783	8939643	8720300	12103006	14688714
FCFS	313803	331713	536364	555324	572319	6697937	7461169	7783518	11028896	11567902
TWD-CTCG	340211	162602	194826	384688	380961	2564681	3547325	4228897	5460826	6868261

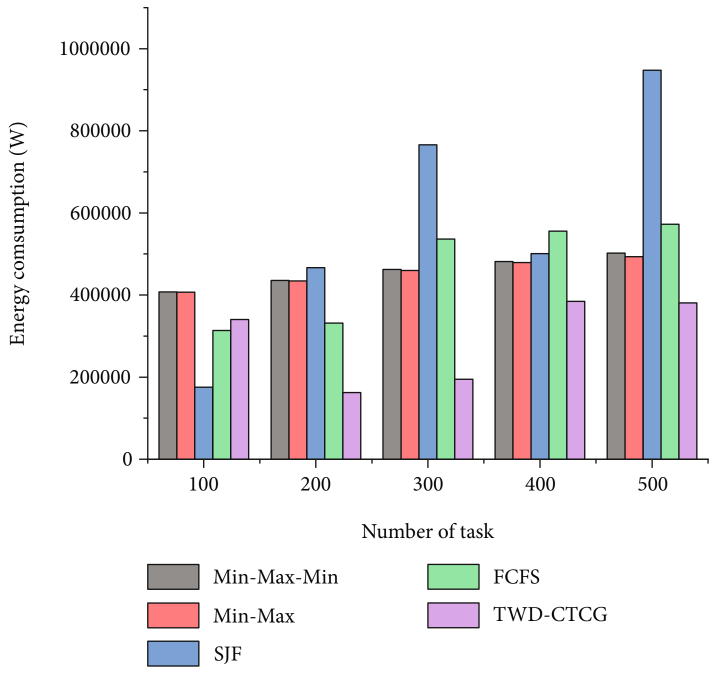

Figure 5: Energy consumption comparison of less tasks.

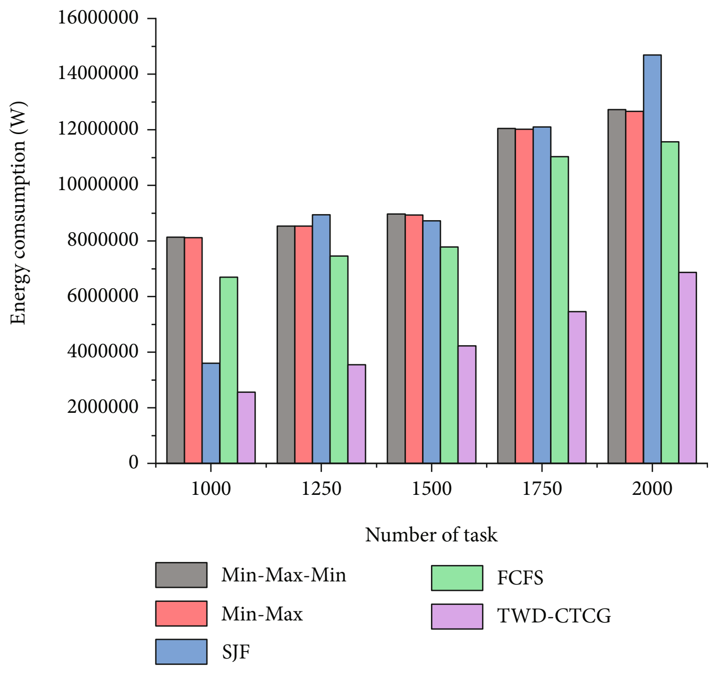

Figure 6: Energy consumption comparison of more tasks.

Table 2: Energy consumption of different hosts.

Algorithm	Hosts 10	20	30	40	50	Hosts 100	150	200	250	300
Min-max-min	187243	288223	389203	490183	591163	5982662	7517162	9051662	10586162	12120662
Min-max	186500	287480	388460	489440	590420	5975722	7510222	9044722	10579222	12113722
SJF	94856	146336	197816	249296	300776	5201153	6636653	8072153	9507653	10943153
FCFS	142428	221628	300828	380028	459228	4891548	6231348	7571148	8910948	10250748
TWD-CTCG	53573	76448	100215	124853	147917	1961099	2469103	2970899	3478323	3980699

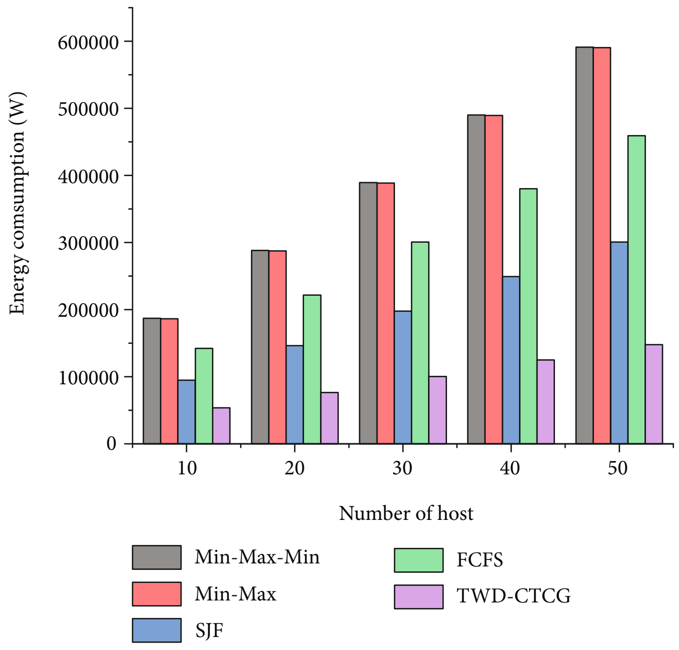

Figure 7: Energy consumption comparison of less hosts.

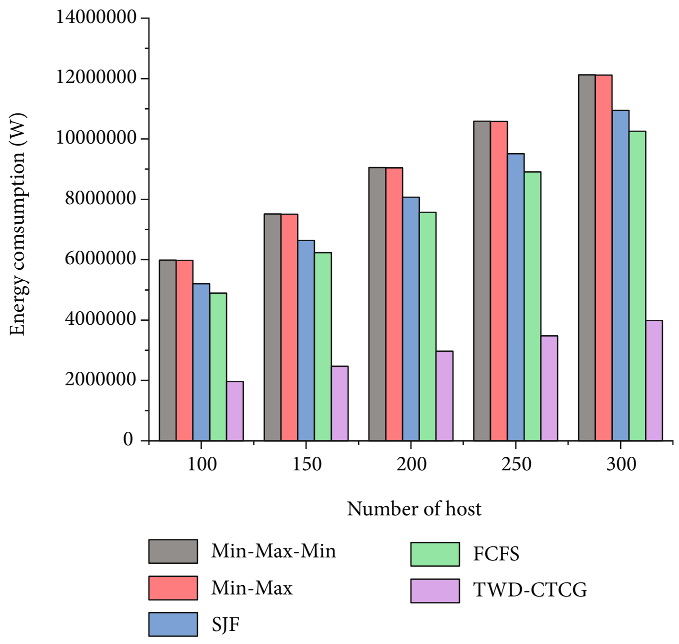

Figure 8: Energy consumption comparison of more hosts.

Table 3: Load balancing of different tasks.

Algorithm	Tasks					Tasks				
	100	200	300	400	500	1000	1250	1500	1750	2000
Min-max-min	0.8870	1.0112	1.1474	1.2725	1.4191	6.7664	7.3118	7.9606	9.7981	10.7014
Min-max	0.8825	1.0131	1.1504	1.2738	1.3978	6.7531	7.2962	7.8984	9.7751	10.6658
SJF	0.5579	0.9607	1.2821	1.1657	1.5770	4.4062	6.9543	7.1081	8.7733	10.1426
FCFS	0.82749	0.82749	1.0932	1.1884	1.2830	5.8229	6.4006	6.8467	8.4989	9.2735
TWD-CTCG	0.87934	0.65183	0.7146	0.9415	0.9281	4.3969	5.1152	5.4756	6.2591	6.8133

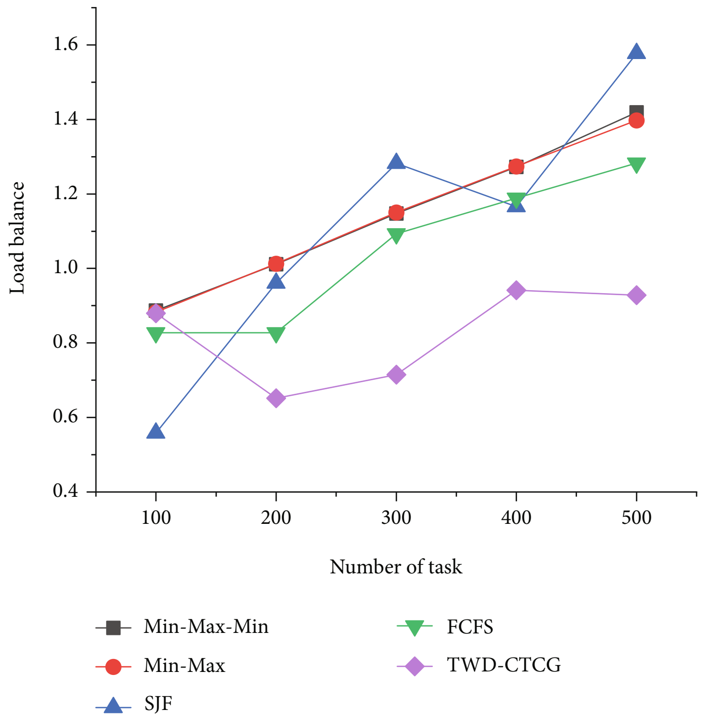

Figure 9: Load balancing comparison of less tasks.

has 1000, 1250, 1500, 1750, and 2000 tasks. The load balancing of more tasks is shown in Table 3 and Figure 10.

According to the analysis of the experimental data in Table 3 and Figure 10, in the case of more tasks, with the increasing of the number of tasks, the load balancing of SJF becomes larger and larger and exceeds FCFS. The load balancing of min-max-min is basically the same as min-max. The TWD-CTCG remains low in the whole process, except that the SJF is the same as TWD-CTCG when the number of task is 1000.

From the above experimental results comparing less tasks and more tasks, it can be seen that the algorithm proposed in this paper is better than other algorithms in terms of load balancing.

4.3.2. *Different Host Experiment Comparison*

(1) Less Host Experiment Comparison. The experiment has 10, 20, 30, 40, and 50 hosts and then compares the load balancing generated by the min-max and min-max-min, SJF, and FCFS algorithms under different hosts. The compared results are shown in Table 4 and Figure 11.

From the experimental data analysis in Table 4 and Figure 11, in the case of less hosts, the load balancing of all algorithms is similar at first, but as the number of hosts increases, the load balancing of min-max-min and min-max becomes larger and larger, and they are relatively similar. The TWD-CTCG algorithm has been kept at a low level, and the effect of FCFS is better than that of SJF in the case of less hosts.

(2) More Host Experiment Comparison. This is a more host load balancing comparison experiment, which has 100, 150, 200, 250, and 300 hosts. The compared results are shown in Table 4 and Figure 12.

According to the analysis of the experimental data in Table 4 and Figure 12, in the case of more hosts, min-

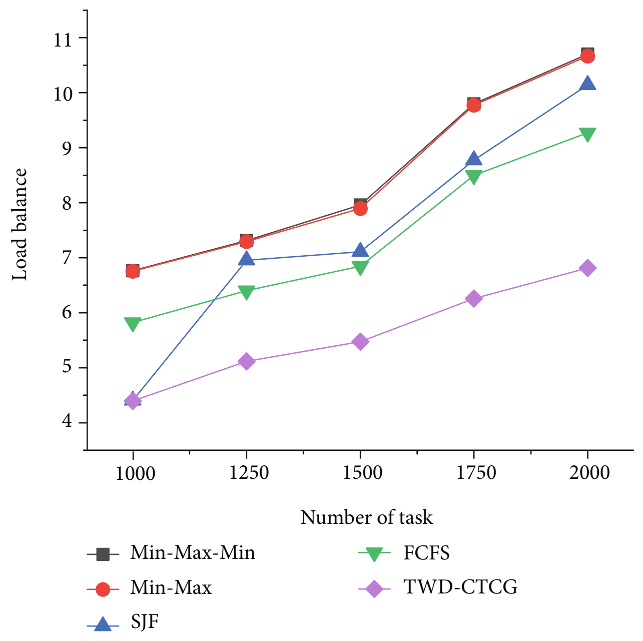

Figure 10: Load balancing comparison of more tasks.

Table 4: Load balancing of different hosts.

Algorithm	Hosts					Hosts				
	10	20	30	40	50	100	150	200	250	300
Min-max-min	0.1765	0.6612	0.8228	0.9036	0.9521	3.4848	9.3179	12.2344	13.9844	15.1510
Min-max	0.1732	0.6566	0.8178	0.8984	0.9467	3.4799	9.3116	12.2274	13.9769	15.1433
SJF	0.1159	0.4517	0.5637	0.6197	0.6533	3.0106	8.1553	10.7277	12.2711	13.3000
FCFS	0.1401	0.5450	0.6799	0.7474	0.7879	2.9456	7.9664	10.4768	11.9830	12.9872
TWD-CTCG	0.1174	0.3844	0.4747	0.5288	0.5472	2.2380	5.4686	7.0559	8.0396	8.6618

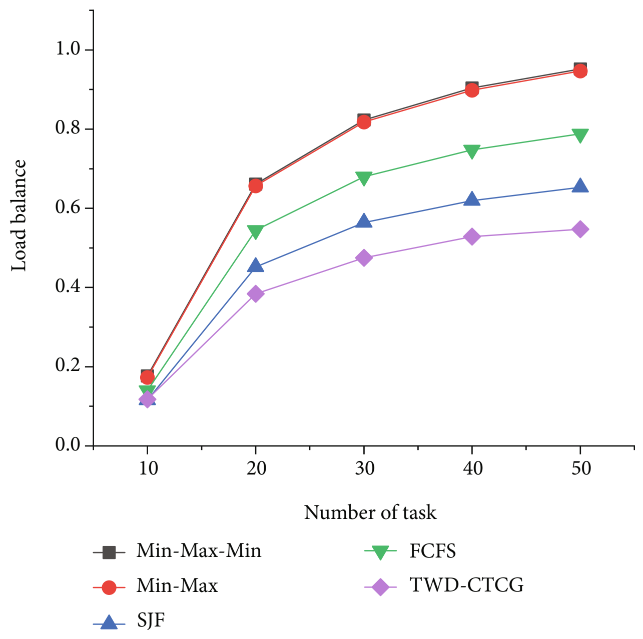

Figure 11: Load balancing comparison of less hosts.

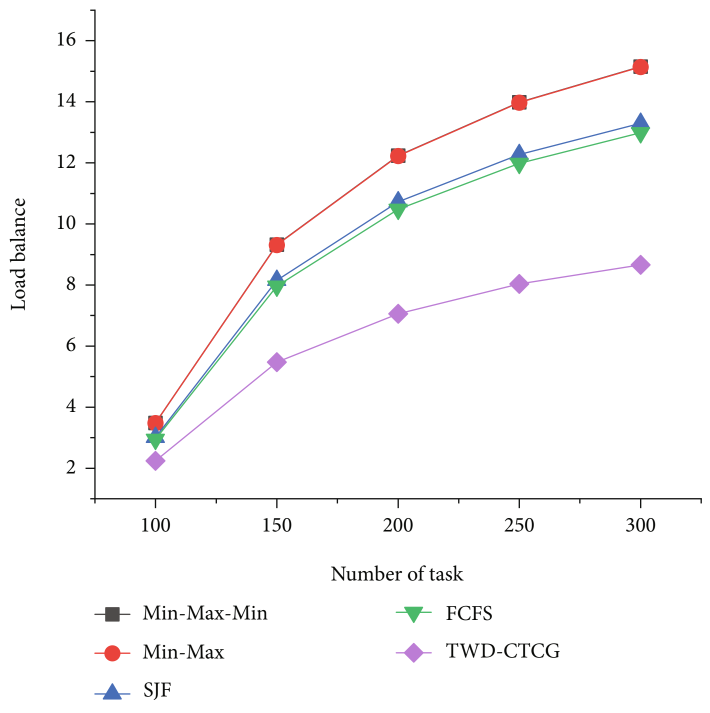

Figure 12: Load balancing comparison of more hosts.

max-min is similar to min-max and SJF is similar to FCFS. In this case, SJF is slightly better than FCFS, but TWD-CTCG has been keep a relatively low level.

In the case of less hosts and more hosts, TWD-CTCG maintains a good effect, indicating that TWD-CTCG has a good effect on load balancing in various situations of hosts.

5. Conclusions

In the development process of cloud computing, energy consumption optimization is still one of the important issues. This paper proposes TWD-CTCG algorithm, which cluster tasks by resource requests and schedule cloud tasks with greedy strategy to achieve reasonable resource allocation.

In the future work, it is necessary to increase the dataset and introduce big data processing methods based on the in-depth research on Hadoop, Spark big data platform, container cloud platform, etc. into our researches. The task clustering density will be more refined, the threshold function will be further improved, and the dynamic threshold will be used to further save energy consumption and improve resource utilization.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Shuaishuai Liu and Xinyu Ma contributed equally to this work.

Acknowledgments

This work is supported by the Postgraduate Innovation Project of the Harbin Normal University (HSDSSCX2021-30).

References

[1] Z. Zhang, M. Zhao, H. Wang, Z. Cui, and W. Zhang, "An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty," *Information Sciences*, vol. 583, pp. 56–72, 2022.

[2] H. Yuan, H. Liu, J. Bi, and M. Zhou, "Revenue and energy cost-optimized biobjective task scheduling for green cloud data centers," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 817–830, 2021.

[3] F. Mahan, S. M. Rozehkhani, and W. Pedrycz, "A novel resource productivity based on granular neural network in cloud computing," *Complexity*, vol. 2021, Article ID 5556378, 15 pages, 2021.

[4] L. Decker, D. Leite, F. Viola, and D. Bonacorsi, "Comparison of evolving granular classifiers applied to anomaly detection for prctive maintenance in computing centers," in *Proceedings of the 2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pp. 1–8, Bari, Italy, 2020.

[5] D. A. Kumar, S. K. Meher, and K. P. Kumari, "Fusion of progressive granular neural networks for pattern classification," *Soft Computing*, vol. 23, no. 12, pp. 4051–4064, 2019.

[6] L. Shi, J. Xu, L. Wang et al., "Multitask associated task scheduling for cloud computing based on task duplication and

insertion," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6631752, 13 pages, 2021.

[7] L. W. Jia, K. Li, and X. Shi, "Cloud computing task scheduling model based on improved whale optimization algorithm," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 4888154, 13 pages, 2021.

[8] Y. Yao, "Tri-level thinking: models of three-way decision," *Journal of Machine Learning and Cybernetics*, vol. 11, no. 5, pp. 947–959, 2020.

[9] D. Okumura, H. Kawabata, and S. A. Chester, "A general expression for linearized properties of swollen elastomers undergoing large deformations," *Journal of the Mechanics and Physics of Solids*, vol. 135, article 103805, 2020.

[10] S. Y. Alsadi and Y. F. Nassar, "A general expression for the shadow geometry for fixed mode horizontal, step-like structure and inclined solar fields," *Solar Energy*, vol. 181, pp. 53–69, 2019.

[11] T. Zhan, S. Ma, W. Li, and W. Pedrycz, "Exponential stability of fractional-order switched systems with mode-dependent impulses and its application," *Cybernetics*, 2021.

[12] D. Liu, T. R. Li, D. C. Liang, and X. Yang, "The temporality and spatiality of three-way decisions," *CAAI Transactions on Intelligent Systems*, vol. 14, no. 1, pp. 141–149, 2019.

[13] D. Liu, X. Yang, and T. Li, "Three-way decisions: beyond rough sets and granular computing," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 5, pp. 989–1002, 2020.

[14] W. Li, X. Xue, X. Weihua, T. Zhan, and B. Fan, "Double-quantitative variable consistency dominance-based rough set approach," *International Journal of Approximate Reasoning*, vol. 124, pp. 1–26, 2020.

[15] X. Weihua and W. Li, "Granular computing approach to two-way learning based on formal concept analysis in fuzzy datasets," *IEEE Transactions on Cybernetics*, vol. 46, no. 2, pp. 366–379, 2016.

[16] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and clustering of workload in google cluster trace based on resource usage," in *2016 IEEE Intl conference on computational science and engineering (CSE) and IEEE Intl conference on embedded and ubiquitous computing (EUC) and 15th Intl symposium on distributed computing and applications for business engineering (DCABES)*, pp. 740–747, IEEE, New York, USA, 2016.

[17] Q. Zhang, J. Hellerstein, and R. Boutaba, "Characterizing task usage shapes in Google compute clusters," 2011.

[18] G. Da Costa, L. Grange, and I. De Courchelle, "Modeling, classifying and generating large-scale Google-like workload," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 305–314, 2018.

[19] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, vol. 1, 2011.

[20] Y. Li, Z. Zhu, and Y. Wang, "Min-max-min:a heuristic scheduling algorithm for tasks across geo-distributed datacenters," *Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 1573-1574, Vienna, Austria, 2018.

[21] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Abraham, "Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing," *Neural Computing and Applications*, vol. 30, no. 6, pp. 1845–1863, 2018.

[22] T. Aladwani, "Types of task scheduling algorithms in cloud computing environment," in *Scheduling Problems-New Applications and Trends*, IntechOpen, London,UK, 2020.