information

MDPI

Article

Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing

Jian Li [1], Tinghuai Ma [1,2,*], Meili Tang [1], Wenhai Shen [3] and Yuanfeng Jin [4,*]

1 School of Computer Software, Nanjing University of Information Science & Technology, Nanjing 210044, China; keng1585291@163.com (J.L.); meilitg@126.com (M.T.)
2 CICAEET, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing 210044, China
3 National Meteorological Information Center, Beijing 100080, China; shenwh@cma.gov.cn
4 Department of Mathematics, YanBian University, Yanji 133002, China
* Correspondence: thma@nuist.edu.cn (T.M.); yfkim@ybu.edu.cn (Y.J.)

Academic Editor: Willy Susilo
Received: 31 December 2016; Accepted: 17 February 2017; Published: 24 February 2017

Abstract: In cloud computing, some large tasks may occupy too many resources and some small tasks may wait for a long time based on First-In-First-Out (FIFO) scheduling algorithm. To reduce tasks' waiting time, we propose a task scheduling algorithm based on fuzzy clustering algorithms. We construct a task model, resource model, and analyze tasks' preference, then classify resources with fuzzy clustering algorithms. Based on the parameters of cloud tasks, the algorithm will calculate resource expectation and assign tasks to different resource clusters, so the complexity of resource selection will be decreased. As a result, the algorithm will reduce tasks' waiting time and improve the resource utilization. The experiment results show that the proposed algorithm shortens the execution time of tasks and increases the resource utilization.

Keywords: cloud computing; fuzzy clustering algorithms; FIFO; scheduling algorithm; resource clustering

1. Introduction

With the explosive growth of information, the demand on computing is sharply increasing. For example, users always hope that they can use computing resources and finish computing rapidly—anytime, anywhere. Besides, there are many heterogeneous resources in the world. In order to use more available resources, many researchers have proposed some computing models, such as grid computing and parallel computing. Since 2006, cloud computing has gradually become the most popular commercial computing model [1]. The target of cloud computing is on-demand billing. Through virtualization technology, cloud computing makes heterogeneous physical resources into virtual resource pools and then users could pay to use these resources. Cloud computing fully considers reliability of resources and scalability of computing, and realizes sharing multiple resources comprehensively. Moreover, cloud computing could handle massive amounts of data and satisfy multiple computing needs. Due to a large number of computing tasks and heterogeneous resources, scheduling algorithm is an important part of cloud computing and has a great influence on quality of cloud service. When computing involves high complexity and mass computation, the resource allocation strategy may affect the efficiency of scheduling heavily. Currently, task scheduling in cloud environment gradually focuses on all computing resources. There are many resources and tasks in cloud environment, so a scheduler could classify and analyze computing resources reasonably. The clustering analysis of resources and the calculation of tasks' resource expectation are beneficial for reasonable choices about resources. In addition, they could reduce the execution time of tasks.

The survey on scheduling algorithms in cloud computing [2] and the survey on distributed job scheduling based on Swarm Intelligence [3] are useful for us to characterize the various problems in cloud computing. Besides, they give us some motivation. To reduce the decrease in execution time of tasks and increase resource utilization of the system, we proposed an improved First-In-First-Out (FIFO) scheduling algorithm based on kernel-based fuzzy c-means clustering algorithm. In the algorithm, a kernel-based fuzzy c-means clustering algorithm has been used to classify resources and FIFO algorithm has been improved by multi-queues. Inverse trigonometric function formula has been used to calculate the similarity of tasks' expectation and resources. Experimental results show that the proposed algorithm could reduce the execution time of tasks and improve the utilization of resources of the system. The main contribution of this paper is as follows:

(1) Combining kernel-based fuzzy c-means clustering algorithm and the improved FIFO algorithm to design a new scheduling algorithm. In the kernel-based fuzzy c-means clustering algorithm, we use radial basis function (RBF) as the kernel function.
(2) Using new methods to calculate the similarity of the task and resources to assign tasks.

The main contents of this paper are as follows. Section 2 introduces the related work. Section 3 describes the scheduling model. In Section 4, we introduce our task scheduling algorithm. Section 5 shows the experiment results and the results will be analyzed. Finally, the conclusion and future works are put forward.

2. Related Work

Task scheduling is an important component of cloud computing and has a great impact on quality of cloud service. The basic problem of task scheduling that needs to be solved is assigning tasks to resources. Unfortunately, it is an NP (Non-deterministic Polynomial) complete problem. According to different objectives, there are different scheduling systems. As a result, there will be different scheduling algorithms in different cloud systems. Different tasks may have different computing requirements and resources which are heterogeneous. Recently, to improve the efficiency of scheduling and reduce the tasks' execution time, the clustering methods have been introduced into task scheduling to classify resources. After classification, tasks can be assigned to resources.

The development of cloud computing is based on grid computing and many researchers have made a lot of contributions in grid computing. Klaus Krauter et al. [4] made a taxonomy and survey of grid resource management systems. In their survey, they grouped design objectives into three themes: improving application performance, data access, and enhanced services. As a result, the grid systems are grouped into three categories: computational grid, data grid and service grid. Furthermore, they divide the resource model into two categories: schema and object model. In addition, they classify the scheduler organization into three categories: centralized, hierarchical and decentralized. In the end, the paper introduces some grid resource management systems. Fatos Xhafa and Ajith Abraham [5] made a survey on computational models and heuristic methods for grid scheduling problems. In their article, they listed some scheduling problems in grid systems and introduced us to some computational models for grid scheduling. In addition, they introduced some heuristic and meta-heuristic methods for scheduling in grids and the integration of schedulers. They also listed some issues, such as security, in the article.

2.1. Fuzzy c-Means Clustering Algorithm (FCM)

In grid computing, some researchers have utilized fuzzy c-means clustering algorithm to improve scheduling. Tarek Helmy and Zeehasham Rasheed [6] proposed the independent job scheduling by fuzzy c-means clustering and an ant optimization algorithm. They proposed a framework which combines the fuzzy c-means clustering algorithm with an ant colony optimization (ACO) algorithm to improve scheduling decisions. They used the FCM to classify the jobs into appropriate classes and the ACO to map the jobs to the appropriate resources. Siriluck Lorpunmanee et al. [7] proposed

fuzzy c-means and genetic algorithms-based scheduling for independent jobs in computational grid. They presented the method of job classification based on fuzzy c-means algorithm and mapped the jobs to the appropriate resources based on genetic algorithm.

In cloud computing, FCM has been used to optimize scheduling. Mahesh S. Shinde and Anilkumar Kadam [8] used fuzzy c-means and a linear programming approach to optimize cloud task scheduling. They used FCM to classify the batch of images into three categories: small images, medium images and large images. When processing the medium images, they used the linear programming to find the value that satisfies objective function and constraints. Xiaojun Wang et al. [9] proposed a scheduling algorithm based on a fuzzy clustering algorithm. They used CBFCM (Cost based Fuzzy Clustering Algorithm) to classify the cloud computing resources and analyzed the jobs with different scheduling algorithms.

2.2. Scheduling Algorithms

"Cloud computing" was first mentioned by Compaq in their business plan in 1996. Since Google proposed their cloud computing in 2006, many famous technology companies have launched their own cloud computing platform, such as AWS (Amazon Web Service), Microsoft Azure, aliyun, and cloud computing has become the most popular business computing model. Hadoop is a project developed by the Apache Software Foundation that is inspired by GFS (Google File System) and MapReduce. It is a framework for the distributed processing of large data sets across clusters of computers using simple programming models. Many companies have adopted Hadoop as the basis architecture of their cloud computing platform, such as Yahoo! Facebook and Alibaba.

There are three common schedulers in cloud computing platforms, namely FIFO scheduler, Capacity scheduler (Yahoo!) and Fair scheduler (Facebook).

FIFO scheduler is the default scheduler in Hadoop and it chooses the job to execute according to job arrival time. By default, FIFO scheduler does not consider a job's priority. When taking priority into account, FIFO scheduler will first consider the priority but it does not support capture [10]. Based on FIFO scheduler, Shujun Pei et al. [11] designed TLI (Task Locality Improvement) scheduler in Hadoop. They put jobs into several queues according to the probability's threshold of the task locality. The jobs that have ideal data are put into queue 1 and others are put into queue 2. If queue 2 is not empty, the job in queue 2 will be preferentially executed in the next scheduling.

Capacity scheduler was developed by Yahoo! and it supports multi-queues. It will allocate a certain number of resources to each queue and adopt FIFO strategy in each queue. To prevent one user from occupying all resources in one queue, capacity scheduler will limit the resource number that one user occupies. During the scheduling, capacity scheduler first chooses a suitable queue based on the strategy which will calculate the ratio between the number of tasks in each queue and their deserved resources and choose the queue with the smallest ratio. In the selected queue, capacity scheduler will choose the job according to priority and the submit time of jobs and take into account resource and memory limitation that users could use. Shivani Thakur et al. [12] proposed a new dynamic capacity scheduler to decrease tasks' execution time. They add the properties in Hadoop to collect the running container and then add it to the scheduler. The scheduler will distribute jobs in the queue with less capacity than others and destroy the queue to make the queue come to capacity as resources area unit.

Fair scheduler always assigns jobs according to the resource pools and fairly distributes resources to these pools. By default, every user has an independent resource pool to ensure that every user will get a copy of the same cluster resources no matter how many jobs they have submitted. In each resource pool, the running jobs share capacity with fair strategy. Users will also give the corresponding weight to the resource pool to make jobs share the resource pool in a disproportionate manner. Besides, fair scheduler allows for distributing minimal resources to the resource pool. Shanjiang Tang et al. [13] proposed a new resource allocation strategy. They adopted different methods to solve memoryless source allocation fashion. For single-level resource allocation, they take the fairness of the total amount of allocated resources into account. The core idea lies in lending one user's available resources to

others. For hierarchical resource allocation, they relax the constraint on the total amount of allocated resources by giving starved users higher priority to get resource allocated. In addition, they also provide a LTYARN (LONG-TERM YARN) fair scheduler, which designs a long-term max–min fairness model and implements LTYARN, to implement the fair resource allocations.

To improve the efficiency of scheduling in shared MapReduce environment and decrease the average response time under different workloads, Yi Yao et al. [14] proposed a new scheduler named LsPS (Leverage the job size Patterns Scheduler). Their scheduler utilizes the workload patterns, in which the scheduler always adjusts the resource user's share and the scheduling algorithms for each user, to reduce average job response times. The main consists of LsPS includes workload information collection, scheduling among multiple users and scheduling for every user.

Based on Swarm Intelligence (SI) techniques, Elina Pacini et al. [15] designed two schedulers to realize multi-objective optimization for online scientific clouds.

3. Scheduling Models

The cloud system abstracts heterogeneous physical resources into resource pools through virtualization technology. When users submit tasks to the system by different clients, the scheduler in the system will assign tasks to the suitable virtual machine that holds sufficient resources. The main process of task scheduling is as shown in Figure 1.

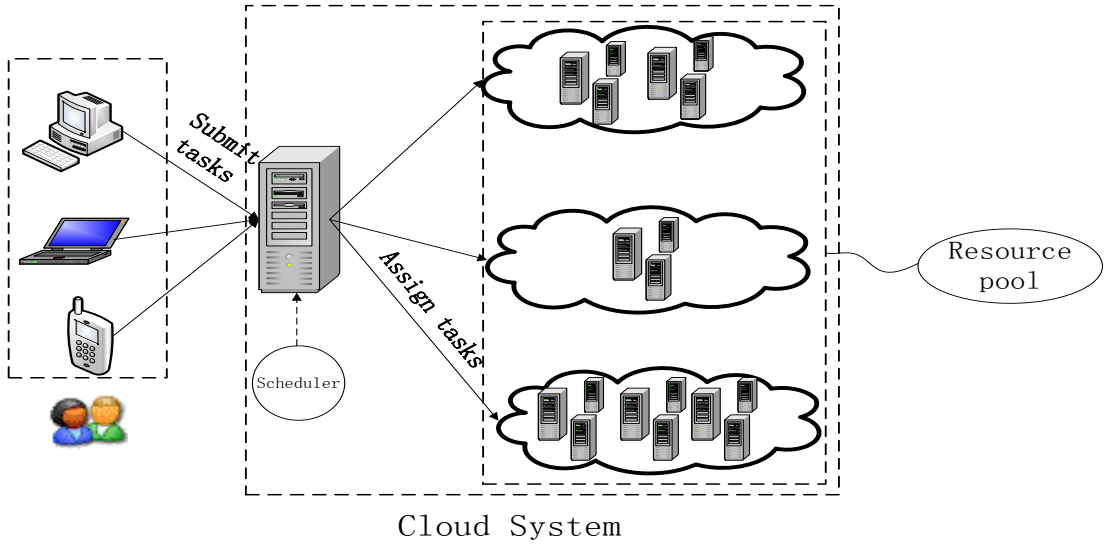

Figure 1. The main process of task scheduling.

In cloud computing, the essence of task scheduling is allocating n tasks to m heterogeneous physical resources, so that the total completion time of these tasks will be minimal and the resources could be fully utilized to the greatest extent possible. In some other cases, cloud system may also take other factors into consideration, such as economy, load balance, optimum makespan and QoS (Quality of Service). Cloud computing usually designs models to utilize virtualization technology to collect heterogeneous resources into resource pools and designs different algorithms for different targets. In this paper, we build two models: a task model and a resource model.

3.1. Task Model

As we know, tasks in a cloud system can be divided into independent tasks that are independent between each other and workflow tasks that have precedence constraints between each other. In this paper, we only take independent tasks into consideration. We assume that cloud systems have received m tasks within a period of time.

These tasks have different sizes and are independent of each other. We use $cloudlet_i$ to represent the ith task and a vector to describe $cloudlet_i$, which is shown as follows:

$$\begin{aligned} &cloudlet_i \leq cloudlet_{i_{id}}, cloudlet_{i_{userid}}, cloudlet_{i_{length}}, \\ &cloudlet_{i_{PEs}}, cloudlet_{i_{bandwidth}}, cloudlet_{i_{storage}}, cloudlet_{i_{ifile}}, \\ &cloudlet_{i_{ofile}} > i = 1, \ldots, m \end{aligned}$$

where $cloudlet_{i_{id}}$ is the ID of the task, $cloudlet_{i_{userid}}$ is the ID of the user who submits the task, $cloudlet_{i_{length}}$ is MIPS (million instructions per second) of the task, $cloudlet_{i_{PEs}}$ is the quantity of CPU (Central Processing Unit) users expect, $cloudlet_{i_{bandwidth}}$ is the bandwidth (unit is MB) of current network, $cloudlet_{i_{storage}}$ the needed space (unit is MB) to store the task, $cloudlet_{i_{ifile}}$ and $cloudlet_{i_{ofile}}$ are respectively the size of input and output files.

The computing demand of $cloudlet_i$ can be calculated by Formula (1)

$$cloudlet_{icompute} = cloudlet_{ilength} / cloudlet_{iPEs} \tag{1}$$

3.2. Resource Model

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. We assume that there are n virtual machines (VM) that have different performances in a resource pool called datacenter. We use the VM_j to represent the j^{th} resource and another vector to describe the characteristic of VM_j.

$$VM_j \leq vm_{j_{id}}, vm_{j_{userid}}, vm_{j_{pesNumber}}, vm_{j_{MIPS}}, vm_{j_{bandwidth}}, vm_{j_{size}} > j = 1, \ldots, n$$

In the vector, $vm_{j_{id}}$ is the ID of the resource, $vm_{j_{userid}}$ is the ID of the service provider, $vm_{j_{pesNumber}}$ is the quantities of CPU on the VM, $vm_{j_{MIPS}}$ is MIPS (million instructions per second) of a CPU on a VM, $vm_{j_{bandwidth}}$ is the bandwidth of VM_j, $vm_{j_{size}}$ is the storage capacity of VM_j.

The computing capacity of VM_j can be calculated by the following Formula (2)

$$VM_{j_{compute}} = vm_{j_{pesNumber}} * vm_{j_{MIPS}} \tag{2}$$

4. Proposed Algorithms

As mentioned above, resources are usually heterogeneous, so different resources tend to have different capabilities, such as storage, transmission, etc. Many algorithms may choose inefficient resources and it will have a certain influence on completion of the tasks. So, we have proposed a task scheduling algorithm based on kernel-based fuzzy c-means clustering. In cloud computing, some tasks may have a large amount of data, but the computation of these tasks is small, such as document storage. However, some other tasks may require massive calculation. Besides, some tasks may not have large data and computation but require high performance of real-time, such as real-time data display. Furthermore, to reduce the complexity of resource classification, the proposed algorithm only divides resources into three categories: computing resources, storage resources and transmission resources. In every cluster, we use FIFO algorithm to schedule tasks. In addition, we have also improved FIFO algorithm to shorten the completion time of tasks and increase the resource utilization of system. The proposed algorithm contains two parts. The first part is to classify the resources into three clusters, which means the type of tasks that the resource may do well in dealing with, with kernel-based fuzzy c-means clustering algorithm based on the models. The second part is assigning tasks to resources with improved FIFO algorithm. The procedure is as shown in Figure 2. The main process of the algorithm is shown in Algorithm 1.

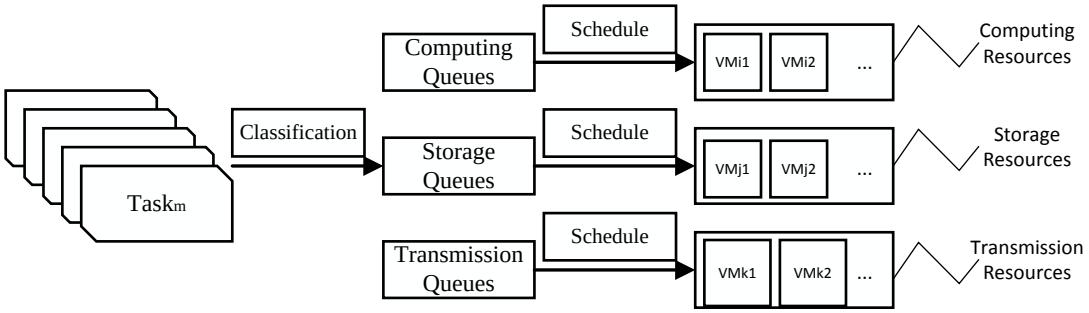

Figure 2. Procedure of Algorithm 1. VM = virtual machine.

Algorithm 1. The main process of the proposed algorithm

Input: a set of tasks
Output: the result of tasks
Process:

Step 1: Classify resources into three clusters through kernel-based fuzzy c-means clustering algorithm;
Step 2: Analyze the preference of the tasks and assign the tasks to three queues;
Step 3: Schedule the tasks in each queue by the improved FIFO algorithm;
Step 4: Output the result of the tasks.

4.1. Resource Classification Based on Kernel-Based Fuzzy c-Means Clustering

Fuzzy clustering algorithms analyze and model the important data based on the fuzzy theory. They establish the uncertainty descriptions of the sample and reflect the real world objectively. The most obvious difference between fuzzy clustering algorithm and some other clustering algorithms is that the value of membership degree is between 0 and 1 in fuzzy clustering algorithm. It gets the membership degree of every sample point to all cluster centers by optimization objective function. Yi Ding and Xian Fu [16] proposed a kernel-based fuzzy c-means clustering algorithm based on genetic algorithm and utilized the kernel-based fuzzy clustering algorithm (KFCM) to guide the categorization. The algorithm defines a nonlinear map as Ψ: $x \to \Psi(x) \in F$, where $x \in X$. X is the data space and F is the transformed feature space with higher or even infinite dimension. KFCM tries to minimize the following objective function:

$$J_{KFCM}(U,V) = \sum_{j=1}^{c}\sum_{i=1}^{n} \mu_{ji}^{m} \left|\Psi(x_j) - \Psi(v_i)\right|^2 \tag{3}$$

$$\left|\Psi(x_j) - \Psi(v_i)\right|^2 = K(x_j,x_j) + K(v_i,v_i) - 2K(x_j,v_i) \tag{4}$$

where $K(x,v) = \Psi(x)^T\Psi(v)$.

In Formula (3) and (4), the parameters are as follows: c is the number of data, n is the number of clustering centers, μ is the membership and m is a variable and set as 2 in our algorithm. The $K(x,v)$ is an inner product kernel function, If we adopt the Gaussian function as a kernel function, i.e., $K(x,v) = \exp(-|x-v|^2/\tau^2)$, then $k(x,x) = 1$. According to Equation (4), Equation (3) can be rewritten as

$$J_{KFCM}(U,V) = 2\sum_{j=1}^{c}\sum_{i=1}^{n} \mu_{ji}^{m}(1 - K(x_j,v_i)) \tag{5}$$

In the Formula (5), U is the membership matrix and V is the cluster centers set. Minimizing Equation (5) under the constraint of U, we have

$$\mu_{ki} = \frac{(\frac{1}{1-K(x_j,v_i)})^{1/(m-1)}}{\sum_{j=1}^{c}(\frac{1}{1-K(x_j,v_i)})^{1/(m-1)}} \tag{6}$$

$$v_i = \frac{\sum_{j=1}^{n}\mu_{ji}^{m}K(x_j,v_i)x_j}{\sum_{j=1}^{n}\mu_{ji}^{m}K(x_j,v_i)} \tag{7}$$

Although Equations (6) and (7) are derived by using the Gaussian kernel function, we can use other functions to satisfy $K(x,x) = 1$ in Equations (6) and (7). In real applications, such as the following RBF functions and hyper tangent functions:

(1) RBF function:

$$K(x,v) = \exp(\frac{-\sum_{i=1}^{c}\left|x_i^a - v_i^a\right|^b}{\tau^2}) \tag{8}$$

(2) Hyper tangent function:

$$K(x,v) = 1 - \tanh(\frac{-|x-v|^2}{\tau^2}) \tag{9}$$

In Formula (8), if set $a = 1$, $b = 2$, the RBF function will be reduced into Gaussian function. In fact, Equation (4) can be viewed as kernel-included new metric in the data space, which is defined as the following:

$$d(x,v) = \left|\Psi(x_j) - \Psi(v_i)\right| = \sqrt{2(1-K(x,v))} \tag{10}$$

The algorithm endows the data points with an additional weight $K(x_k, v_i)$, which measures the similarity between x_k and v_i.

The full descriptions of kernel-based fuzzy c-means clustering algorithm are as the Algorithm 2:

Algorithm 2. Kernel-based fuzzy c-means clustering algorithm

Input: m one-dimensional vectors
Output: three clusters and their cluster centers
Process:

Step 1: Set c, t_{max}, $m > 1$, $\varepsilon > 0$;
Step 2: Initialize the memberships μ_{ji}^{0};
Step 3: For $t = 1, 2, ..., t_{max}$ do:

Update all prototype v_{ji}^{t} with Equation (7);
Update all memberships μ_{ji}^{t} with Equation (6);
Compute $E^t = \max_{ji}\left|\mu_{ji}^{t} - \mu_{ji}^{t-1}\right|$;
If $E^t \leq \varepsilon$
stop;
Else
$T = t + 1$;

Step 4: End.

We use KFCM to get the better initial clustering centers because the performance of the algorithm depends on the initial clustering centers.

4.2. Analysis and Preprocessing of Tasks

In our algorithm, we first get each kind of available resources, such as computing or storage and so on. Then we will get the task's expectation of each kind of resources by the task model vector *VEC* and calculate exp_i (i = 1, . . . , n), which is the ratio of the expectation of the task named $cloudlet_i$ and the available resources. The formula is as shown in Formula (11) below

$$\exp_{i\ cata} = cloudlet_{icata} / available_{cata} \tag{11}$$

In the formula, "cata" is one of the resource types including computing, storage and bandwidth, available; "cata" can be achieved from the vectors.

According to the exp_i of $cloudlet_i$, we put the task into the queue tending to the resource corresponding to the task's maximum expectation. By using the basic classification, some insufficient scheduling will be avoided and the time to search appropriate resources will also be shortened because of fewer targets. After that, we will adopt the improved FIFO algorithm in the next chapter to assign tasks. The procedure of data preprocessing of tasks is as shown in Algorithm 3:

Algorithm 3. The procedure of data preprocessing of tasks

Input: n one-dimensional vector
Output: three task queues
Process:

Step 1: Get the total of available $VM_{compute}$, VM_{size} and $VM_{bandwidth}$;
Step 2: Get $cloudlet_{i_{\text{compute}}}$ $cloudlet_{i_{\text{storage}}}$ $cloudlet_{i_{\text{bandwidth}}}$;
calculate $\exp_{i_{compute}}$ $\exp_{i_{storage}}$ and $\exp_{i_{bandwidth}}$ based on Step 1;
Step 3: Compare $\exp_{i_{compute}}$ $\exp_{i_{storage}}$ and $\exp_{i_{bandwidth}}$;
put $cloudlet_i$ into the queue corresponding to the maximum;
Step 4: Call for the improved FIFO algorithm.

After this process, all of the tasks will be put into queues and wait for scheduling.

4.3. Improved FIFO Algorithm

Taking the utilization and the availability of resources into consideration, the proposed algorithm sets a threshold named *Th* to limit the quantity of tasks in each queue and adjust the assignment of tasks according to the current resources. This algorithm sets *Th* according to the quantities of VMs. During the FIFO scheduling, if the VMs are not available, the algorithm will calculate the tasks' waiting time in other queues. If the waiting time of the task will decrease after reassignment, the task will be reassigned to the queue having the least waiting time.

The waiting time *wt* can be calculated from the following Formula (12)

$$wt_i = t_{i-1_{exe}} + t_{i_{trans}}\ (i\ < \text{limit}) \tag{12}$$

$$t_{i_{exe}} = cloudlet_{compute} / VM_{compute} \tag{13}$$

$$t_{i_{trans}} = cloudlet_{length} / VM_{bandwidth} \tag{14}$$

In each queue, the algorithm chooses the resource based on the inverse trig function between the $cloudlet_i$ and the VM_j. The formula is as shown below:

$$\theta_{ij} = \arccos \frac{cloudlet_{i_{compute}} \times VM_{j_{compute}} + cloudlet_{i_{storage}} \times VM_{j_{size}} + cloudlet_{i_{bandwidth}} \times VM_{j_{bw}}}{\sqrt{cloudlet^2_{i_{compute}} + cloudlet^2_{i_{storage}} + cloudlet^2_{i_{bandwidth}}} + \sqrt{VM^2_{j_{compute}} + VM^2_{j_{size}} + VM^2_{j_{bw}}}} \tag{15}$$

The procedure of improved FIFO algorithm is as shown in Algorithm 4:

Algorithm 4. The procedure of improved FIFO algorithm

Input: three task queues
Output: the execution time of tasks and the utilization of VMs
Process: In each queue:

Step 1: Set the threshold of each queue according to the quantities of VMs;
Step 2: If there is no available VM_j

If the number of tasks in the other queue < Th

Calculate wt_i in the current queue and $wt_i^{'}$ in the other queue

If $wt_i^{'} < wt_i$

Reassign the task;

Else

Break;

Step 3: In each queue, we calculate θ_{ij} of $cloudlet_i$ in the queue and the available VM_j, then choose the VM_j corresponding to the minimal θ_{ij};
Step 4: Execute tasks in the queue and output the results.

Our algorithm uses clustering algorithm and combines with the pruning idea to shorten the finish time. The procedure of Algorithm 3 is as shown in Figure 3. The experiment is as shown in the following Section 5.

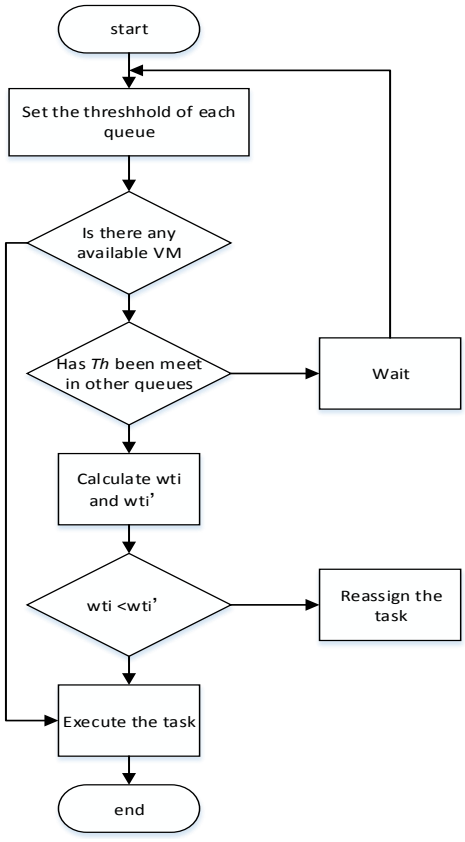

Figure 3. Procedure of Algorithm 3.

5. Experiment and Analysis

Our algorithm aims to decrease the execution time of tasks and increase resource utilization of the system. The experiment has realized original FIFO algorithm and improved FIFO algorithm after using clustering and not using clustering. Besides, based on the scheduling strategy [17], we implement the ACO algorithm as contrast.

5.1. Experiment Configuration

Our experiment uses CloudSim to simulate the real environment. CloudSim [18] is a cloud computing simulation software that was developed by Melbourne University and could model and simulate cloud computing. Our experiment realizes the FIFO algorithm and the improved FIFO algorithm based on classifying the resources with fuzzy c-means clustering algorithm. According to the results of the experiment, we evaluate these algorithms from two aspects: execution time and the average resource utilization.

In our experiment, the specific experimental configuration is as follows:

(1) The parameters of platform: there are 10 datacenters that have 4 hosts. The configuration of each host includes 2000 MIPS compute speed, 4 GB memory, 1 TB storage, 10 GB/s bandwidth and the quantity of PE (Processing Element) is 1, 2 or 4. The characteristic of each datacenter is that the system architecture is x86 and the operating system is Linux.

(2) The parameters of tasks: task length is in [500, 4000], the range of bandwidth is between [1000, 2000] and the range of storage is between [512, 2048]

(3) The parameters of VMs: the quantity of CPU is between {1, 2, 4}, the range of CPU speed is between [500, 1000], the range of bandwidth is between [500, 3000] and the range of storage is between [512, 4096]

Taking an example, the main parameters of a system with 10 VMs is as shown in the following Table 1:

Table 1. Parameters of system.

VM ID	Computing (MIPS)	Bandwidth (MB/S)	Storage (GB)
0	10,000	4000	8192
1	8000	2000	10,240
2	4000	10,000	4096
3	16,000	4000	6144
4	3000	5000	5120
5	5000	7000	3072
6	12,000	6000	10,240
7	8000	4000	12,288
8	10,000	8000	8192
9	9000	9000	16,384

MIPS = million instructions per second.

5.2. Experiment Results

5.2.1. The Results of Resource Clustering

We use fuzzy quantification and min–max standardization to process data and the formula of standardization is as shown in Formula (16) and (17) below:

$$vm'_{ij} = 1/2 + 1/2 * \sin(\pi/(vm_{j\max} - vm_{j\min}) * (vm_{ij} - (vm_{j\max} - vm_{j\min})/2)) \quad (16)$$

$$vm''_{ij} = (vm'_{ij} - vm'_{j_{\min}})/(vm'_{j_{\max}} - vm'_{j_{\min}}) \quad (17)$$

Original resources data and the standardized VMs data when the VM number is 10 are shown in Figure 4 below:

$$\begin{bmatrix} 10000 & 4000 & 8192 \\ 8000 & 2000 & 10240 \\ 4000 & 10000 & 4096 \\ 16000 & 4000 & 6144 \\ 3000 & 5000 & 5120 \\ 5000 & 7000 & 3072 \\ 12000 & 6000 & 10240 \\ 8000 & 4000 & 12288 \\ 10000 & 8000 & 8192 \\ 9000 & 9000 & 16384 \end{bmatrix} \rightarrow \begin{bmatrix} 0.874 & 0.500 & 0.677 \\ 0.677 & 0.146 & 0.874 \\ 0.216 & 0.854 & 0.216 \\ 0.874 & 0.500 & 0.440 \\ 0.126 & 0.691 & 0.323 \\ 0.323 & 0.962 & 0.126 \\ 0.985 & 0.854 & 0.874 \\ 0.677 & 0.500 & 0.985 \\ 0.874 & 1.000 & 0.677 \\ 0.784 & 0.962 & 0.874 \end{bmatrix} \rightarrow \begin{bmatrix} 0.871 & 0.415 & 0.641 \\ 0.641 & 0.000 & 0.871 \\ 0.105 & 0.829 & 0.105 \\ 0.871 & 0.415 & 0.366 \\ 0.000 & 0.642 & 0.229 \\ 0.229 & 0.956 & 0.000 \\ 1.000 & 0.829 & 0.871 \\ 0.641 & 0.415 & 1.000 \\ 0.871 & 1.000 & 0.641 \\ 0.766 & 0.956 & 0.871 \end{bmatrix}$$

Figure 4. Process of data.

After the fuzzy clustering, we will divide these resources into three clusters, which are *Cluster*0 = {vm_0, vm_3, vm_6}, *Cluster*1 = {vm_2, vm_4, vm_5, vm_8, vm_9} and *Cluster*2 = {vm_1, vm_7}. In these clusters, *Cluster*0 is marked as computing resource, *Cluster1* is marked as bandwidth resource and *Cluster*2 is marked as storage resource.

5.2.2. The Results of Scheduling and Analysis

In our experiment, we set task number n = {100, 200, 300, 400, 500, 600, 800, 1000} and VM number m = {10, 20, 30, 40, 50, 60, 80, 100}. We implement and compare FIFO, ACO and our algorithm by means of execution time and the average resource utilization. The result is as follows.

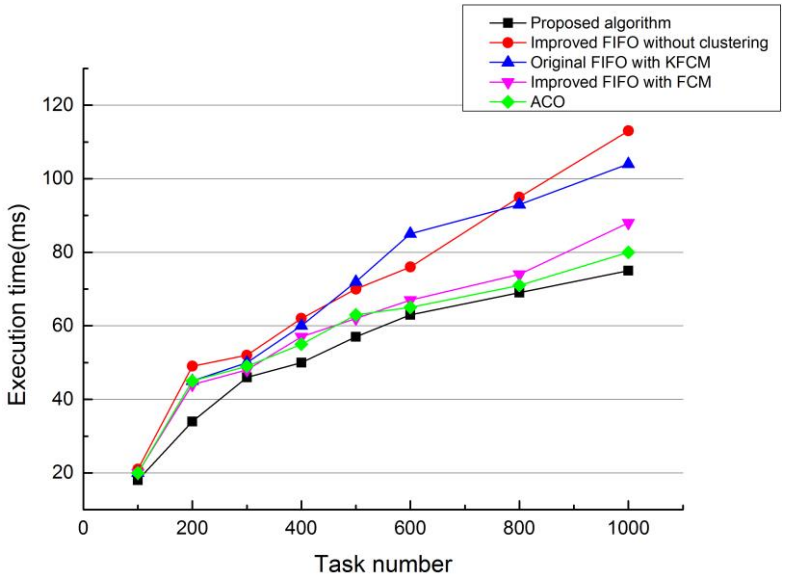

Figure 5. Execution time of the four algorithms.

From Figure 5, we can find that the execution time of proposed algorithm is the minimal among the four algorithms and the decrease is more obvious with the increase of task number, for example, when n = 1000, the execution time of our algorithm is reduced by approximately 33.6% of improved FIFO without clustering and approximately 27.9% of the original FIFO with KFCM. By comparing the execution time of the other three algorithms related to FIFO and the proposed algorithm, we

can find that the proposed algorithm will finish tasks earlier, so we can guess that the multi-queues could improve the FIFO algorithm in scheduling and clustering algorithms can make contributions to reduce the scheduling time. By comparing the execution time of ACO and the proposed algorithm, we can draw the conclusion that the proposed algorithm is effective in reducing execution time of tasks. Besides, we believe that the difference between results of the proposal algorithm and others will be more obvious if resource homogeneity is further increased.

From Figure 6, we can find that the resource utilization has ups and downs with an increase in the number of tasks. At the beginning, with the increase of task number, the resource utilization will increase, but when the task number is more than 600, the resource utilization will decrease because of the increase of data. However, the resource utilization of the proposed algorithm is always more than 40% and is always the maximal among the five algorithms. We can believe that the resource utilization will be improved by classifying the resources.

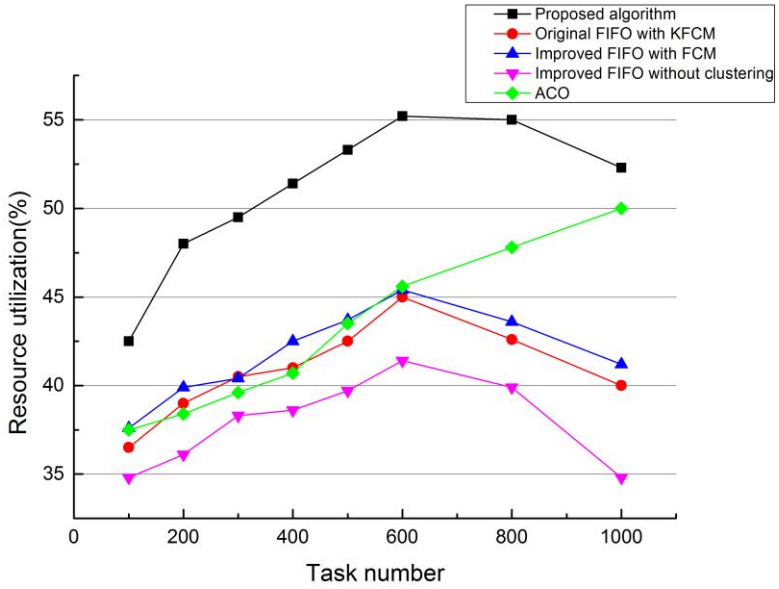

Figure 6. Resource utilization of the four algorithms.

Based on the above analysis, we can assume that our algorithm can apparently shorten the execution time and improve the resource utilization obviously.

6. Discussion

We study the scheduling problem of independent tasks in cloud computing, summarize other scheduling algorithms, introduce the clustering algorithms and propose a new scheduling algorithm. Our algorithm abstracts resources into a model and analyzes these characteristics of resources with the fuzzy c-means clustering algorithm to narrow the range of resources and shorten the execution time of tasks. From the experiment, we find that our algorithm could decrease the execution time of tasks and increase resource utilization of the system.

Next, we will take the dynamic characteristic of tasks and the link between them and take further research on scheduling algorithms. We will take the QoS into account, further optimize the algorithm and use some open source tools to do experiments with new data sets. Moreover, we will take some alternative approaches, such as ACO and PSO (Particle Swarm Optimization), into account.

Acknowledgments: This work was supported in part by the National Science Foundation of China (No. 61572259), Special Public Sector Research Program of China (No. GYHY201506080).

Author Contributions: Tinghuai Ma proposed the idea and provided the financial support; Jian Li designed the experiments; Wenhai Shen provided the experiment data; Meili Tang and Yuanfeng Jin revised the manuscript; Jian Li wrote the paper. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bosoteanu, M.C. Cloud Accounting In Romania. A Literature Review. *Risk Contemp. Econ.* **2016**, 400–405.
2. Vijindra, S.S. Survey on Scheduling Issues in Cloud Computing. *Procedia Eng.* **2012**, *38*, 2881–2888. [CrossRef]
3. Pacini, E.; Mateos, C.; Garcia, G.C. Software Survey: Distributed job scheduling based on Swarm Intelligence: A survey. *Comput. Electr. Eng.* **2014**, *40*, 252–269. [CrossRef]
4. Krauter, K.; Buyya, R.; Maheswaran, M. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Softw. Pract. Exp.* **2000**, *32*, 135–164. [CrossRef]
5. Xhafa, F.; Abraham, A. Computational models and heuristic methods for Grid scheduling problems. *Future Gener. Comput. Syst.* **2010**, *26*, 608–621. [CrossRef]
6. Helmy, T.; Rasheed, Z. Independent Job Scheduling by Fuzzy C-Mean Clustering and an Ant Optimization Algorithm in a Computation Grid. *IAENG Int. J. Comput. Sci.* **2010**, *37*, 136–145.
7. Siriluck, L.; Noor, M.S.M.; Hanan, A.A.; Surat, S. A static jobs scheduling for independent jobs in Grid Environment by using Fuzzy C-Mean and Genetic algorithms. In Proceedings of the Postgraduate Annual Research Seminar 2006, Johor Bahru, Malaysia, 24–25 May 2006; p. 20.
8. Mahesh, S.; Kadam, A. Cluster Oriented Optimized Cloud Task Scheduling Strategy using Linear Programming. *Int. J. Comput. Appl.* **2015**, *128*, 26–31.
9. Wang, X.; Wang, Y.; Hao, Z.; Du, J. The Research on Resource Scheduling Based on Fuzzy Clustering in Cloud Computing. In Proceedings of the 8th International Conference on Intelligent Computation Technology and Automation, Nanchang, China, 14–15 June 2015; pp. 1025–1028.
10. White, T. *Hadoop: The Definitive Guide*; O'Reilly Media Inc. Gravenstein Highway North: Sebastopol, CA, USA, 2010.
11. Pei, S.J.; Zheng, X.M.; Hu, D.M.; Lou, S.H.; Zhang, Y.X. Optimization and Research of Hadoop Platform Based on FIFO Scheduler. In Proceedings of the 7th International Conference on Measuring Technology & Mechatronics Automation, Nanchang, China, 13–14 June 2015; pp. 727–730.
12. Thakur, S.; Singh, R.; Sharma, S. Dynamic Capacity Scheduling in Hadoop. *Int. J. Comput. Appl.* **2015**, *125*. [CrossRef]
13. Tang, S.; Lee, B.S.; He, B. Fair Resource Allocation for Data-Intensive Computing in the Cloud. *IEEE Trans. Serv. Comput.* **2016**. [CrossRef]
14. Yao, Y.; Tai, J.; Sheng, B.; Mi, N. LsPS: A Job Size-Based Scheduler for Efficient Task Assignments in Hadoop. *IEEE Trans. Cloud Comput.* **2015**, *3*, 411–424. [CrossRef]
15. Pacini, E.; Mateos, C.; Garino, C.G. Multi-objective Swarm Intelligence schedulers for online scientific Clouds. *Computing* **2016**, *98*, 495–522. [CrossRef]
16. Ding, Y.; Fu, X. Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm. *Neurocomputing* **2015**, *188*, 233–238. [CrossRef]
17. Wen, W.T.; Wang, C.D.; Wu, D.S.; Xie, Y.Y. An ACO-based Scheduling Strategy on Load Balancing in Cloud Computing Environment. In Proceedings of the Ninth International Conference on Frontier of Computer Science and Technology, Dalian, China, 26–28 August 2015; pp. 364–369.
18. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience. *Softw. Pract. Exp.* **2010**, *41*, 23–50. [CrossRef]