International Journal of Grid and Distributed Computing
Vol. 9, No. 4 (2016), pp.41-52
http://dx.doi.org/10.14257/ijgdc.2016.9.4.04

A Load Balancing Task Scheduling Algorithm based on Feedback Mechanism for Cloud Computing

Zhang Qian, Ge Yufei, Liang Hong, Shi Jin

Computer and Communication Engineering,
China University of Petroleum Qingdao, 266580, China
azhangqianupc@163.com, bgeyfupc@163.com, cliangh@upc.edu.cn,
d474469076@qq.com

Abstract

Scheduling algorithm is always a hot topic in cloud computing environment. In order to eliminate system bottleneck and balance load dynamically. A load balancing task scheduling algorithm based on weighted random and feedback mechanisms was proposed in this paper. At first the chosen cloud scheduling host chose resources by needs and made static quantification, and then sorted them; secondly the algorithm chose resources from which sorted by weight randomly; then it acquired corresponding dynamic information to make load filter and sort the left. At last it achieved the self-adaptively to system load through feedback mechanisms. The experiment shows that the algorithm has avoided the system bottleneck effectively and has achieved balanced load as well as self-adaptability to it.

***Keywords**: Task Scheduling; Feedback Mechanism; Cloud Computing; Load Balancing*

Introduction

Cloud computing is the further development of the research on the traditional parallel computing, distributed processing, grid computing and grid storage based on a cluster, or the specific representative of the commercialization of computer science concept [1]. Cloud computing environment organization is mainly divided into centralized and peer-to-peer, the centralized computing environment is easy to be implemented and after knowing the running state of the global resources, the scheduling node implements the task scheduling and allocates tasks reasonably, but when the system is larger, centralized task scheduling process makes the performance and stability of the scheduling node become the system bottleneck. As a result, the centralized scheduling is suitable for small-scale cloud computing environment. Peer-to-peer cloud computing environment solves the bottleneck problems existing in centralized scheduling mode, supporting dynamic entry and exit of computing node and suitable for large-scale computing environment. However, peer-to-peer scheduling algorithm is difficult to be designed and implemented. In a unknown state of global resources, to schedule resources only based on local information is not likely to achieve the goal of scheduling optimization. Although the scheduling process of peer-to-peer scheduling algorithms such as ant colony and particle swarm tends to be global optimum, the algorithm is difficult to be implemented, the scheduling process is complex and the cost of computing and network transmission is large.

According to the results of researches in literature [2] [3], at present most of the cloud computing environment usually uses relatively simple scheduling algorithms and at the time of task scheduling and resource allocation cloud platform may appear uneven load. These scheduling algorithms dispatches tasks of minimal and earliest completion time to the corresponding resources, or dispatches tasks of maximal and earliest completion time to the corresponding resources, or choose dispatching nodes of minimal value of trust and benefit. They don't handle the relationship between the node load and task allocation. If some of unequal or overloaded nodes are selected, it will reduce the overall performance of the system.

Therefore, according to the ineffective system execution in cloud computing environment possibly caused by unbalanced load, a distributed scheduling algorithm based on weighted random [4] and feedback mechanism was proposed in this paper. On the premise of increasing algorithm implementation complexity indistinctively, it guarantees that excellent nodes won't be overloaded and there is a chance for general performance computing nodes to execute tasks by using the weighted random, overload assessment and feedback strategy.

Algorithm Description

Task scheduling algorithm proposed in this paper is based on peer-to-peer cloud computing environment. Each computing node has the complete function of receiving tasks, resource evaluation, task scheduling and execution. In the process of scheduling, the implementation of the scheduling is carried on after considering the running state of the global resource comprehensively. The description of task scheduling process is as shown in Figure 1:

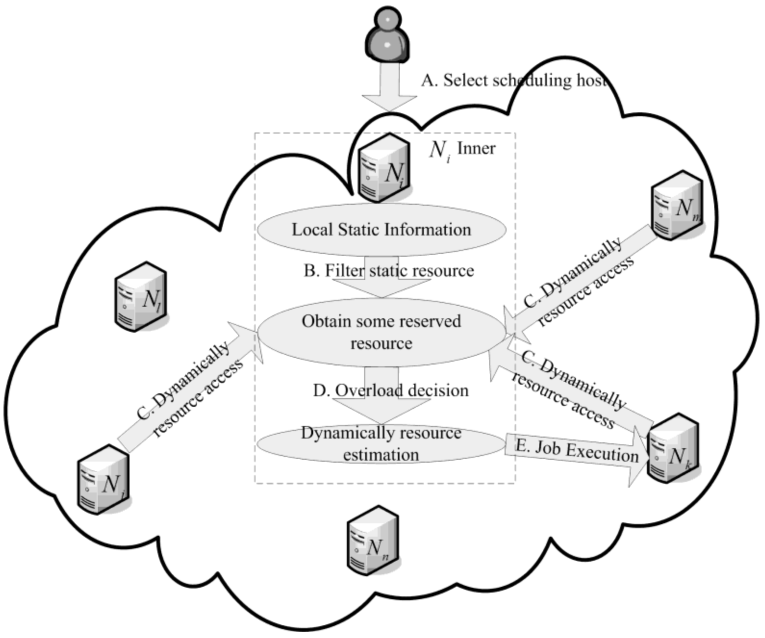
A. Select scheduling host
N_i Inner
N_i
Local Static Information
B. Filter static resource
Obtain some reserved resource
D. Overload decision
Dynamically resource estimation
C. Dynamically resource access
C. Dynamically resource access
C. Dynamically resource access
E. Job Execution
N_l
N_j
N_n
N_m
N_k

Figure 1. Description of Task Scheduling Process

The description of task scheduling process is as follows:

(1) Choose the scheduling host. Users choose a host in the computing environment to submit tasks according to a certain strategy, in Figure. 1 the user selects N_i to submit tasks, N_i is responsible for evaluating the running state of computing resources in the system, dispatching tasks to the execution host, monitoring the variation of the task status and feeding back the information of task execution.

(2) Filter static resource information. The scheduling host uses configuration requirements of task resources, eliminates resources which don't meet the minimum configuration of task execution, uses vector standardization method to evaluate multiple static indicators of resources comprehensively based on static resource information in the

local storage, sort s resources based on the merits of indicators and uses probabilistic algorithm to select several backup computing resources.

(3) Access dynamic information resource. Access the real-time information of backup computing resources through the network.

(4) Overload judgment. Eliminate the resources of an indicator above overload standard according to the real-time information. Evaluate remaining computing resources comprehensively using vector standardization method and select the computing resource of the highest dynamic indicator to execute tasks. If all of backup resources are eliminated because of overload judgment, go back to step (2), flitter the static resource information again. However, the filtering process eliminates the previous backup computing resources and adjusts the probability algorithm, letting more computing resources selected. After the scheduling, adjust the probability algorithm according to the ratio of the number of backup hosts to the number of remaining hosts after overload judgment.

(5) Scheduling host delivers tasks to the execution unit, monitors the execution status of the task and feeds back the execution results.

(6) Geometric weighted coefficient feedback regulation. Adjust the geometric weighted coefficient according to the ratio of the number of backup hosts to the number of remaining hosts after overload judgment.

Resource Requirements Description

The algorithm described in this paper is based on the task process which has been analyzed and the task which has been decomposed. The sub-tasks of each task have atomicity, that is no precedence constraints, not running across nodes and atomic. Each subtask can present minimum configuration requirement to resources which execute tasks. Task configuration requirement is defined as follows:

$$R = \{R_{proc}, R_{mem}, R_{disk}, R_{comm}, R_{os}\} \tag{1}$$

Different processing logic of task business influences that tasks have special requirements for some properties of computing resources, for example, compute-intensive tasks are likely to have higher demand for CPU performance and memory capacity, and storage intensive tasks are likely to have higher demand for hard disk or network bandwidth, therefore, the task defines the requirement weight of different resource attribute as follows:

$$w = \{w_{proc}, w_{mem}, w_{disk}, w_{comm}\} \tag{2}$$

Resource Attribute Description

Definition of Resource Attributes. The paper describes the computing unit which can implement the function of executing tasks as the resource. According to different functions, resources can be divided into computing resources, storage resources and network resources, *etc*. First, the definition of nodes is as follows:

$$NS = \{N_1, N_2, \ldots, N_n\} \tag{3}$$

The attribute set which influences service capability in the node $(i = 1, 2, \ldots, n)$ includes processor capacity, memory, storage capacity, hard disk storage capacity, network communication capacity, reliability and queue load. The definition is as follows:

$$RS = \{RS_{proc}, RS_{mem}, RS_{disk}, RS_{comm}, RS_{rely}, RS_{queue}\} \tag{4}$$

Each attribute of resources can be measured through specific value. The scheduling algorithm uses attribute value as the basis of evaluating resource performance. RS_{proc} includes three main attributes, the number of processors, dominant frequency and operation load parameter values, the definition is as follows:

$$RS_{proc} = \{p, k, p(t)\} \tag{5}$$

In (5), p stands for the frequency of the processor, k stands for the kernel number, $p(t)$ and stands for the processor load at t. If $k > 1$, $p(t) = \{p_1(t), p_2(t), ..., p_k(t)\}$ is a set, and is a time-related attribute parameter.

In the paper, RS_{mem}, RS_{disk} and RS_{comm} all contain two parameters, the size and load of the memory, the size and load of the disk, network bandwidth and load, the definition is as follows:

$$RS_{mem} = \{m, m(t)\},\ RS_{disk} = \{d, d(t)\}\ \ RS_{comm} = \{c, c(t)\} \tag{6}$$

In (6), the memory load, disk load and network load are time-related parameters.

The reliability RS_{rely} is measured through $r(t)$, the successfully completed probability of several tasks that resources execute recently, $r(t)$ is defined as follows:

$$r(t) = success\_task(t) / recent\_total\_task(\mathrm{t}) \tag{7}$$

The queue load is measured through $q(t)$, the ratio of the number of tasks accepted by task processing queue at present to the number of the tasks that task processing queue can hold. $q(t)$ is defined as follows:

$$q(t) = queue\_used(t) / queue\_size \tag{8}$$

Resource Attribute Value Standardization. Among the decision problems of multi-objective attribute, the way of quantizing attributes directly affects objective assessment. The paper uses multiple indicators in the process of evaluating resources; each indicator has different unit and dimension. In order to reflect the actual performance state of the resources better, each indicator needs to be compared under unified metrics; therefore, before evaluating each computing node resource comprehensively, we need dimension and standardize related indicators. The paper uses the geometric mean method in the vector standardization method [5] to standardize the resource attributes, the method is simple to calculate and influenced by the extreme value smaller. The method is as follows:

$$Z_{ij} = \frac{X_{ij}}{\sqrt{\sum (X_{ij})^2}} \tag{9}$$

Algorithm Implementation

Resources Filtering On-Demand and Static Attribute Quantization

Filtering Resources On-Demand. After the scheduling unit receives the task request, according to the formula (1)-(9), the algorithm filters the resources based on the static attributes and selects u available resource nodes which meet the minimum requirements for the task, the expression is $N_i \ (i = \{1, 2, \ldots, u\})$.

Available Resources Static Attribute Quantization. Static resource attribute is the unchanged part of resource attribute indicators during the running of the system. According to the definition (5) and (6), static attribute indicator of a single computing node is defined as follows:

$$S = \{p, k, m, d, c\} \tag{10}$$

In the definition, p stands for the frequency of the processor, k stands for the kernel number, m stands for the memory capacity, d stands for the peripheral storage capacity and c stands for the bandwidth.

Because different resource attribute has different dimension, we uses the formula (9) to dimension and standardize static resource attributes, as follows:

$$p_i = \frac{p_i \times k_i}{\sqrt{\sum_{j=1}^{u} (p_j \times k_j)^2}} \quad m_i = \frac{m_i}{\sqrt{\sum_{j=1}^{u} m_j^2}} \quad d_i = \frac{d_i}{\sqrt{\sum_{j=1}^{u} d_j^2}} \quad \tilde{c}_i = \frac{c_i}{\sqrt{\sum_{j=1}^{u} c_j^2}} \tag{11}$$

In the formula, $i = \{1, 2, \ldots, u\}$ stands for all the available computing resources. In the formula (11), all respectively represents the standardized result of the processor, memory, hard disk, network, the standardization of a single computing node. According to the definition (2), tasks have different requirement for different attribute indicator of resources. Therefore, the definition of static resource evaluation criteria is as follows:

$$\eta_i = w_{proc} \times p_i + w_{mem} \times m_i + w_{disk} \times d_i + w_{comm} \times \tilde{c}_i \quad i = \{1, 2, \ldots, u\} \tag{12}$$

We can obtain the evaluation of the available computing node $N_i \ (i = \{1, 2, \ldots, u\})$.

The Task Mapping based on Weighted Random and Feedback Mechanism

In the previous step we chose u available resources on-demand. If the available resources are more, we still need choose l backup resources with excellent performance from them. First we use bubble sort algorithm to sort the resources in descending order based on η that the formula (12) calculate. After the process produces v sequence nodes, the sorting algorithm is over. At this moment, the nodes from l to v are in descending order and the nodes from $v+1$ to u are out of order, u>v>l and

$$\forall i \in [v+1, u], \text{ and } \eta_i < \eta_v \tag{13}$$

We use the sorting algorithm to truncate computing nodes with poor performance and need use randomized algorithm to select l backup computing units from the remaining v resources with better performance. This paper calls v as performance intercepting window width and l as backup resources window width.

It is generally believed that the performance of the computing resources intercepted by the performance intercepting window v is excellent and the time is close when the tasks are submitted to any host among them. Therefore, using the weighted random algorithm to select l backup resources from v resource nodes not only can satisfy that tasks can be executed on nodes with better performance, but also avoids the imbalanced load caused by that all of scheduling nodes submit tasks to the most excellent computing node.

Weighted random uses geometric weighted method. The geometric weighted coefficient is γ, therefore, the probability of two adjacent ordered resources is satisfying:

$$a_{i+1} = \gamma a_i \text{, and } i = \{1, 2, \ldots, v-1\}\text{, } 0 < \gamma < 1 \tag{14}$$

In (14), a_i stands for the probability of the resource whose static performance is arranged in i to be selected. Because r is between 0 and 1, the probability of the resource with better static performance to be selected is higher and the probability decreases geometric to γ. In one choice the sum of the probability of all v computing nodes to be selected should be equal to 1. According to the geometric series summation formula, we can obtain the probability of the resources with optimal performance to be selected, the probability is

$$a_1 = (1-\gamma) / (1-\gamma^v) \tag{15}$$

We can find the probability a_k of the computing resource in the k place of the order to be selected using the formula (14) and (15).

We can generate a random number z between (0,1) using uniformly distributed random number generator and determine the selected host in the order using the following method:

$$\sum_1^k a_k \le z < \sum_1^{k+1} a_{k+1}\text{, } k = \{1, 2, \ldots, v-1\} \tag{16}$$

We can see from the formula above that Begin to cumulate the probability from the beginning of the order until the cumulative value is more than a random number z; the k computing unit of the order is selected.

We can find the value of k using the formula (14) and (16):

$$k = \left\lfloor \frac{\ln(1 - \frac{z \times (1-\gamma)}{a_1})}{\ln \gamma} \right\rfloor \tag{17}$$

Therefore, the process of using the weighted random algorithm to extract a backup resource is as follows:

(1) According to the known proportion coefficient γ and the performance intercepting window v, we use the formula (15) to find the probability a_1 of the optimal resource in the order to be selected.
(2) We use uniformly distributed random number generator to generate a random number z between 0 and 1.
(3) We calculate the value of k according to the formula (17), delete the computing host which ranks in the k place of the order and add it to the list of backup hosts.

In the algorithm above, we use the probability to select a backup computing resource from the sorted resource list. After the selected computing host k is deleted, the number of arranged host sample space will become v – 1. At this moment, we select the computing host with the highest value of η from the computing resource available (namely from $v+1$ to u) cut out by the performance intercepting window to join in the end of the order. We can use the 2 and 3 step of the algorithm above to find the second randomly selected backup computing node. Repeat the above process until l backup computing resources are selected.

Geometric weighting is just a simple weighted kind in weighted random methods, suits to choose a set of resources with obvious different individual performance. If the performance of some resources in a set of resources is similar, we can group the resources according to the performance (fuzzy clustering [6]), confirm the selection probability between groups using geometric weighted method, the level probability is shared between the resources with similar performance within the group. The later experiment is done in this way. After getting the alternative set using weighted random, carry on equal probability random to get the final resources within the group.

Filter Overloaded Resources

After filtering on demand, static quantization and weighted randomly selection, the selected l backup computing resources are the ones with better static attributes. The scheduling host need evaluate the real-time information of the backup computing resources and select the computing host with the best current running status to submit tasks. Through information service component, the scheduling served host gets the dynamic information of the backup resources. According to the definition of (5), (6), (7), (8), the description of the dynamic information of computing resources is as follows:

$$D=\{p(t),m(t),d(t),c(t),r(t),q(t)\} \tag{18}$$

The dynamic information is respectively the processor load, memory load, hard disk load, network load, credibility and tasks queue load. According to the definition, all of the loads are within the interval $(0,1)$.

The definition of the standard for the load overweight is as follows:

$$L=\{L_p,L_m,L_d,L_c,L_r,L_q\} \tag{19}$$

The standardized indicators for the load overweight are respectively the processor, memory, hard disk, network, credibility and the top limitation of task queue load. The paper argues that if an attribute indicator of the computing resources has exceeded the top limitation of the corresponding indicator, task resource has run overloaded and is not suitable for continuing to submit tasks to it. It is obvious that the indicator of overweight ranges from 0 to 1.

Before the overload assessment, we need preprocess some data of the dynamic resources, for example, computing host may have multiple kernels. As a result, $p(t)$ is a set, the average of the kernel processor load is obtained using the average method, as shown below:

$$\overline{p}(t)=\frac{\sum_{i=1}^{k} p_k(t)}{k} \tag{20}$$

Use $m(t)$、 $d(t)$、 $c(t)$、 $r(t)$、 $q(t)$ in the formula (18), $\overline{p}(t)$ in the formula (20) and the standardization for the overload judgment in the formula (19). If an indicator of the resources has exceeded the standardization for the overload judgment, the host is called as the overloaded host and not suitable for continuing to submit tasks to it.

Quantitative Ordering of Dynamic Attributes

Overload assessment eliminates the computing resources with heavy load and remains a number of hosts which have better performance in the system and are the computing nodes with relatively light load. The algorithm finally chooses the optimal computing nodes to submit tasks from them using dynamic assessment.

Before the dynamic assessment, we need standardize the dynamic information of the computing nodes. Because each computing processor may have multiple kernels, the process of the indicators is as follows:

$$p_i'(t) = \sum_{i=1}^{k_i} [(1 - p_i(t)) \times p_i] \tag{21}$$

In the formula, k_i stands for the core number of the host N_i.

We can find the standardized value of the dynamic attribute indicator of a single computing resource using the formula (9), (11) and (21), as follows:

$$p_i(t) = \frac{p_i'(t)}{\sqrt{\sum_{j=1}^{g} (p_j'(t))^2}} 、 \quad m_i(t) = \frac{(1 - m_i(t)) \times m_i}{\sqrt{\sum_{j=1}^{g} [(1 - m_j(t)) \times m_j]^2}}$$

$$d_i(t) = \frac{(1 - d_i(t)) \times d_i}{\sqrt{\sum_{j=1}^{g} [(1 - d_j(t)) \times d_j]^2}} 、 \quad \tilde{c}_i(t) = \frac{(1 - c_i(t)) \times c_i}{\sqrt{\sum_{j=1}^{g} [(1 - c_j(t)) \times c_j]^2}} \tag{22}$$

In the formula, g stands for the remaining number of the backup computing nodes after the overload filter in 4.3.

According to the definition (2) of the degree of tasks' thirst for resources and the formula (22), the definition of the dynamic evaluation value of a single resource is as follows:

$$\eta_i(t) = w_{proc} \times p_i(t) + w_{mem} \times m_i(t) + w_{disk} \times d_i(t) + w_{comm} \times \tilde{c}_i(t) \tag{23}$$

Select the computing resource with the biggest $\eta_i(t)$ ($i = \{1, 2, \dots, g\}$) from the remaining backup calculation node and submit task scheduling to the node.

Load Information Feedback Regulation

After filtering resources on demand and static attributes quantization, we use the performance intercepting window v to exclude computing nodes with poor performance and select l backup computing resources from v resources with better performance using stochastic weighted. In the process of selecting backup resources, the proportion coefficient γ directly affects the probability distribution of resource extraction. When γ tends to 1, the probability of v sorted backup computing resources to be selected is close; when λ is away from 1, the probability of the excellent computing node in the front to be selected will increase. For example, when v is taken 4, λ is taken 0.5, we can get the probability of four sorted computing resources to be selected are respectively:

$$a_1 = \frac{8}{15} 、 \quad a_2 = \frac{4}{15} 、 \quad a_3 = \frac{2}{15} 、 \quad a_2 = \frac{1}{15}$$

In the example above, the probability of the resource with the most excellent static attributes is more than 0.5.

After the selection of backup calculation nodes, the algorithm evaluates the dynamic attributes of the resource and uses overload judgment to rule out busy computing nodes. At last g resources which can continue to deliver tasks remains. Eventually the ratio of the number of idle resources g to the width of system backup resource window l can reflect the busy degree of a system to some extent. The definition of the system load on the basis of this is as follows:

$$e = g / l \tag{24}$$

When the whole system is idle, all of the selected l computing nodes with excellent performance will pass the overload evaluation and become the optional delivery host. at this moment, $e = 1$, it shows that the whole virtual computing environment is idle, computing tasks should be submitted to the resources with particularly excellent performance, therefore, we can reduce the width v of the performance intercepting window and reduce the value of the proportion coefficient γ. The scheduling system will consider more to submit tasks to several computing nodes with the most excellent performance in the system. When $0.5 < e < 1$, it shows that the computing system is relatively idle, therefore, we cannot change the value of v and γ; When $e \le 0.5$, it shows that most of the computing resources with excellent performance in the system are busy, we should broaden the width v of the intercepting window and increase the value of proportion coefficient γ, therefore, the scheduling system will consider fully to submit tasks to the computing nodes with general performance. Thus, it reduces the calculation pressure of the excellent host and reaches the purpose of selecting better host task to execute.

We can set small width $v_{\min}$ of the performance intercepting window and proportion coefficient γ according to the actual situation in the early time of the system operation ($\gamma_{\min}$ should not be too small and should be above 0.5, otherwise we cannot reach the purpose of load balancing). After scheduling, the system uses the following way to adjust the value of v and γ:

When $e = 1$, the system is idle. We should submit tasks more to the better computing resources; therefore, we need narrow the value of v and reduce the value of γ. The way to get new value is as follows:

$$v' = v / 2 , \gamma' = (3\gamma - 1) / 2 \tag{25}$$

If $v' < v_{\min}$, the width v of the performance intercepting window is set to $v_{\min}$, otherwise, set to v'.

If $\gamma' < \gamma_{\min}$, the proportion coefficient γ is set to $\gamma_{\min}$, otherwise, set to γ'.

When $0.5 < e < 1$, it indicates that the system is free, we don't need adjust the window width and the proportion coefficient.

When $e \le 0.5$, the system is very busy and we need increase the width of the performance intercepting window and improve the value of γ. Tasks have the opportunity to be submitted to the hosts with general performance and the calculation pressure of the resources with excellent performance can be relieved. The way to get the new value is as follows:

$$v' = 2v , \gamma' = (1 + \gamma) / 2 \tag{26}$$

The width v of the performance window is set to v' and the proportion coefficient γ is set to γ'.

The feedback mechanism regulates geometric weighted coefficient according to the load of the computing environment. When the system is idle, tasks are submitted to the computing nodes with excellent performance. When the computing nodes with excellent performance are busy, tasks are assigned to the other computing resources with better performance according to the feedback mechanism. The paper uses the load feedback mechanism to improve the resource utilization of the system and balance the load of the system dynamically.

The Experimental Results and Performance Analysis

Algorithm Testing. Cloud computing environment uses four Dell server nodes, four computing nodes with high performance, four common testing nodes to build the peer-to-peer structure. The cloud environment has no central node; each node has complete function and can carry on independent scheduling. Each Dell server node is composed of two CPUs, each CPU has four cores and uses Hyper-Threading technology, each core provides two threads, as a result, each server node is composed of 2 x 4 x 2 logical CPU, the configuration of each node is as shown in Table 1.

Table 1. Host Server Configuration

Host Name	CPU	Mem	OS
Server1	Xeon E5520 2.26GHz（2×4×2）	8.169GB	Linux RHEL 5.1
Server2	Xeon E5520 2.26GHz（2×4×2）	8.375GB	Linux RHEL 5.1
Server3	Xeon E5520 2.26GHz（2×4×2）	8.375GB	Linux RHEL 5.1
Server4	Xeon E5520 2.26GHz（2×4×2）	5.969GB	Linux RHEL 5.1
HP1	Pentium E5300 2.6GHz（1×2）	1.918GB	Linux RHEL 5.1
HP2	Pentium E5300 2.6GHz（1×2）	2.086GB	Linux RHEL5.1
HP3	Pentium E5300 2.6GHz（1×2）	1.918GB	Linux RHEL5.1
HP4	Pentium E5300 2.6GHz（1×2）	2.086GB	WinXP SP 3
Norm1	Pentium E2140 2.6GHz（1×2）	1.027GB	Linux FC6
Norm2	Pentium4 2.6GHz（1×1）	1.034GB	Linux FC6
Com1	Pentium4 2.6GHz（1×1）	0.545GB	Linux FC6
Com2	Pentium4 2.6GHz（1×1）	0.478GB	Linux FC6

Table 2. Host Server Configuration

	Server(4)	HP(4)	Norm(2)	Com(2)	Rate
MET	16	4	0	0	8/12
Random Weight	9	6	3	2	12/12

Ideal Weight	9.2	5.5	3.3	2.0

Experiment 1 uses MET algorithm and weighted random algorithm respectively to schedule 20 lightweight subtasks, in the latter process of the implementation resources can be divided into four groups according to the performance of nodes, groups are carried on through weighted random, namely, v is 4, and take γ the initial value of 0.6, and all the nodes in the environment meet the mission requirements, the final resource utilization is as shown in Table 2.

As we can see from Table 2, the weighted random has better resource utilization than MET and system load is more balanced.

Experiment 2 uses the algorithm proposed in this article to schedule 20 high-load subtasks, the execution time of tasks are more than 20 minutes, we submit a task every 5 minutes, v is still 4, the total number of resources is 12, the backup resource window size l is 4; because the scale of the environment is limited and the task load is not big, we don't take the assigned resources away from the sample space, namely v is constant. The initial value of geometric weighted coefficient γ is 0.7. The variation with time of idle ratio e and geometric weighted coefficient γ is as shown in Table 3:

Table 3. γ Feedback Mechanism

Time t(min)	Load e	Before scheduling γ	After scheduling γ
5	1	0.7	0.7
10	1	0.5	0.5
15	1	0.5	0.5
20	0.75	0.5	0.5
25	1	0.5	0.5
30	0.75	0.5	0.5
35	0.75	0.5	0.5
40	0.5	0.5	0.5
45	0.25	0.5	0.75
50	0.5	0.75	0.75
55	0.25	0.75	0.875
60	0.5	0.875	0.875

As we can see from Table 3, the system load has exceeded the threshold of feedback adjustment at the time of 45 minutes, γ expands according to the constraint of the formula (26). The mechanism carries on the feedback adjustment of the load by increasing the selected weight of the resources which have a bit poor performance but are empty. As we can see from Table 3, the system has reached the dynamic balance of the load at the time of 60 minutes.

Conclusions

This paper proposed a weighted random scheduling algorithm based on the peer-to-peer cloud computing environment, the resource attributes are divided into two parts, the static and dynamic, to be evaluated respectively. The algorithm uses the weighted random strategy, overload assessment and feedback and so on to ensure that the nodes with excellent performance will not be overburdened based on submitting tasks first to the resources with the best performance. At the same time the resources with general performance have the potential to perform a task. The algorithm can effectively balance the load, ensures the basic load balance of the nodes in the network and puts forward a solution for load balancing strategy in cloud computing environment.

Acknowledgements

This work was financially supported by National Natural Science Foundation of China (Grant No. 61309024), Shandong Provincial Natural Science Foundation, China (ZR2014FM017) and the Fundamental Research Funds for the Central Universities (14CX02030A).

References

[1] Z Hai-bin, T Lin-sha, L Li-xiang, "Survey of grid scheduling [J].Computer Engineering and Design", , vol. 30, no. 9, **(2009)**, pp. 2151-2153, 2190.

[2] L Chang, T Da, "An improved weighted random sampling algorithm [J]".Software,**(2011)**, vol. 1, pp. 14-17

[3] W Xing-zhu, Z Qing-huai, "Research of scheduling algorithms for virtual grid service flow [J]", Computer Engineering and Design, vol. 10, **(2009)**, pp. 2371-2374.

[4] O Ibarra, C Kim, "Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors [J]", Journal of the ACM, vol. 77, **(1997)**, pp. 280-289.

[5] A.E., Eiben, P-E Raué., Zs Ruttky, "Genetic algorithms with multiparent recombination: Proceedings of the 3rd Conference on Parallel Problem Solving from Nature", [s.l.]: Springer-Verlag, **(1994)**, pp.78-87.

[6] Z Min, Y Jian, "Fuzzy Partitional Clustering Algorithms [J]". Journal of Software, **(2004)**, vol. 15, no. 06, pp. 858-869.