International Journal of Grid and Distributed Computing
Vol.6, No.5 (2013), pp.1-16
http://dx.doi.org/10.14257/ijgdc.2013.6.5.01

A Load Balancing Algorithm with Key Resource Relevance for Virtual Cluster

Xu Chaoqun, Zhuang Yi and Zhu Wei

College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

xcq_nuaa@163.com, zhuangyi@nuaa.edu.cn

Abstract

Load balancing is one of key techniques in the virtual cluster system. In view of the fact that resource relevance has not been considered in the load balancing algorithm under current virtual cluster application environment, this paper proposes a load balancing algorithm with key resource relevance (RRK). Firstly, virtual resources are divided into groups by category. Then, considering the relevance between user tasks and each virtual resource group as well as the integrated load of each virtual resource group, the priorities of the tasks assigned to each virtual resource group are dynamically calculated, and thus the tasks can be assigned to the corresponding virtual resource group based on the priorities; while those tasks will be distributed depending on load values of the virtual resources and the weight values of resources needed to be consumed. The experimental results show that: RRK algorithm, compared with MTN algorithm and Weight algorithm, cannot only assure that the tasks can be preferentially assigned to those virtual resources of high relevance, but also enable the tasks with less average waiting time for assignments of virtual resource groups.

Keywords: *Load Balancing, Resource Relevance, Virtual Cluster, Weight Value*

1. Introduction

The virtualization enables the construction of virtual cluster [1]. The virtual cluster means: connecting multiple homogeneous or heterogeneous computing resources together and then virtualizing multi-computing resources out of one computing resource via virtualization, thus constructing a huge cluster system that is similar to the physical cluster [2, 3]. The core part of virtual cluster is to consider the virtual machine as the unit of assigning computing resources.

As the computer techniques are increasingly developing nowadays, the contradiction between continuous improvement of computer hardware performance and the low resource utilization rate of cluster system becomes increasingly prominent, which is mainly caused by the load imbalance [4]. Besides, the load balancing is also particularly important in the virtual cluster.

In the existing load balancing algorithms, little consideration is taken on the resource relevance. However, the resource relevance is generally the important factor that needs to be considered in actual task assignment. For example, in some specific cluster application environments, users wish to preferentially assign different tasks to resources of different categories as required; also, the multi-virtual machines virtualized by physically homogeneous computers may adopt different operating systems, but user tasks generally need to run on designated operating systems.

In this paper, a kind of load balancing algorithm with key resource relevance for virtual cluster (RRK) is proposed based on the relevance between user task and virtual cluster, task weight value and load value of each virtual resource. There are two steps in this algorithm: Firstly, assign the task to one certain virtual resource group in line with the task relevance to each virtual resource group and the integrated load of each virtual resource group; Then, assign the task to one certain virtual resource in line with the load value about all virtual resources in the virtual resource group and the task weight value. As to the acquisition of load information, a method combining "push" and "pull" together is hereby adopted, which means: all virtual resources periodically "push" their own load values; when a new task is to be assigned, the system will "pull" back the latest load values to all virtual resource groups. The experimental results show that: compared with MTN algorithm and Weight algorithm, RRK algorithm cannot only assure that the tasks can be preferentially assigned to those virtual resources of high relevance to themselves, but also enable the tasks with less average waiting time for assignment with in virtual resource groups.

2. Related work

Via virtualization, all physical resources are virtualized and transparently provided to users [5]. The virtualization technology is mainly classified into three types [6]:

(1) Resource virtualization: Virtualize specific resources such as CPU, disk, memory and network resources, and transparently provide them to users for use.

(2) Platform virtualization: Namely the operating system of virtualized computer, which can be classified into full virtualization and para-virtualization. The full virtualization does not need to modify the source codes of operating system and it can be well compatible with operating systems in various versions. The para-virtualization can obtain better performance by modifying the source codes of operating system, such as typical Xen virtual machine, *etc.*

(3) Application program virtualization: This includes simulation, emulation and explanation techniques, typically including the explanation execution mechanism of JAVA program.

The load is an abstract concept describing the busyness of the system, and the distribution balancing of load on all resources of parallel system is called load balancing. The load balancing contributes to assure the high efficiency of task assignment algorithm, and it can adjust the load assignment at all times to keep all resources of the system in load balance. There are two kinds of methods for the load balancing of all resources in the system: load assignment and load migration. The load assignment is to properly assign the user tasks to all resources in order to make the system load on all resources roughly equal. The load migration is to migrate the tasks from heavy-loaded resources to light-loaded resources, so that the system load gets balanced and the overall performance is improved. The load balancing algorithm proposed therein is used in load assignment.

There are three kinds of load balancing algorithm: static, dynamic and self-adaptive.

Based on the static information about related tasks, the static assignment algorithm uses one certain strategy to complete the task assignment. It determines the load distribution without any system status information. This algorithm is simple and convenient for use, but it has poor parallelism and only quite effective in some specific applications [8]. The commonly-used static assignment algorithms include round assignment and priority assignment [9], *etc.*

By analyzing the real-time load information of cluster system, the dynamic load assignment algorithm dynamically assigns and adjusts the tasks among all processors in order to eliminate the imbalance of load distribution in the system. Compared with the static algorithm, the dynamic algorithm is more flexible and efficient. But the dynamic algorithm has to collect, store and analyze the status information, it needs more system overhead than the static algorithm. Nevertheless, this kind of overhead can be counteracted by the increase of computer efficiency [10]. For this reason, the dynamic load assignment algorithm has become the research focus at present. Typical dynamic load balancing algorithms includes ADAPTLOAD algorithm [11] proposed by Q. Zhang et al., address hash algorithm [12] proposed by Beinkowski M et al., and game theory-based algorithm [13] proposed by Daniel Grosu *et al.*, *etc.*

As a kind of special dynamic algorithm, the self-adaptive load balancing algorithm can adjust its own behaviors via dynamically changing parameters to adapt to the changing system status. For example, one load balancing algorithm is quite effective under condition A whereas another algorithm is effective under condition B. Integrated with more than two kinds of strategies, the self-adaptive algorithm can select suitable strategies according to the changes of system status. At present, the popular self-adaptive scheduling algorithms include minimum load algorithm and average load algorithm [14], *etc.*

Nowadays, numerous researches have been made on the load balancing algorithms, such as dynamic feedback-based load balancing algorithm (MTN) [15] proposed by Liu Jian et al, which aims at timely modifying the load information about each node via dynamic feedback mechanism. The weight-based load balancing algorithm (Weight) [16] proposed by Zhang Yufang considers the processing capability of each node, and can fully play the advantage of cluster system. Sachin Kumar et al propose a kind of dynamic load-based priority algorithm used in distributed computer network [17]. It takes the priorities of resources into first consideration, which means the tasks will be assigned to the corresponding resources when the resources of high priorities are free. A content-based load balancing algorithm with admission control for cluster web servers [18], proposed by Saeed Sharifian *et al.*, classifies the users' requests based on their impact on the resources, which means each kind of resources respond to one kind of requests. However, this can classify the tasks to some extent, this classification pattern is not controllable to users. Although these algorithms mentioned above can assure the system load balancing to certain extent, the assignment of tasks to which resources has no laws to follow, which may be against the original intention of users. This is because the relevance between tasks and resources has not been sufficiently considered in existing load balancing algorithms. In view of this, the RRK algorithm proposed therein can assign the tasks to those resources of high relevance on the premise of assuring certain load balance in the system.

3. Load Balancing Model with Key Resource Relevance

The structure of virtual cluster adopted in this paper is shown in Figure 1. Multiple virtual machines run on every physical resource in the cluster system, each of which is here considers as one virtual resource. However, all virtual resources in every virtual resource group may not all run on one same physical resource, and they may come from a number of different physical resources. As they are of the same type, the task can be assigned to and then runs on any one of them. As shown in Figure 1, the virtual resources in italics belong to one same virtual resource group, but apparently they come from different physical resources.

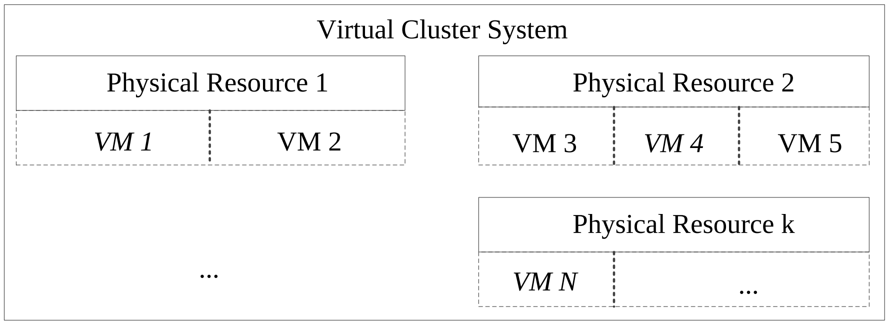

Figure 1. Structure diagram of virtual cluster

The load balancing model with key resource relevance considers the relevance between tasks and virtual resources as the key factor in assignment strategies. Assume Vector T=$(t_1 \quad t_2 \quad \dots \quad t_i \quad \dots \quad t_m)^T$ as the task set, where: t_i represents task i, and m is the number of tasks. Let R=$(r_1 \quad r_2 \quad \dots \quad r_r \quad \dots \quad r_n)$ denote the set of virtual resource groups, where: every r_r represents a set of virtual resources in same category, and n is the number of virtual resource groups. The classification of virtual resource group here is based on the demands of specific tasks. Different virtual machines on one same physical computer can be either of same category or different categories. Use Matrix C=TR= $(t_1 \quad t_2 \quad \dots \quad t_m)^T (r_a \quad r_b \quad \dots \quad r_n)$ = $(f_{(t_1,r_r)} \quad f_{(t_2,r_r)} \quad \dots \quad f_{(t_i,r_r)} \quad \dots \quad f_{(t_m,r_r)})^T$ to represent the matrix of relevance between tasks and virtual resource groups, where: $f_{(t_i,r_r)}$=$(t_ir_1 \quad t_ir_2 \quad \dots \quad t_ir_n)$, indicating the vector of relevance between one certain task t_i between each virtual resource group, with $\sum_{r=1}^{n} f_{(t_i,r_r)}$=1.

Assume L=$(L_1 \quad L_2 \quad \dots \quad L_r \quad \dots \quad L_n)$ is the load vector of virtual resource groups, where L_r is the integrated load value of virtual resource group r_r. So we can define the priority of task t_i to be assigned to virtual resource group r_r is $P(t_i, r_r)$= $\frac{f_{(t_i,r_r)}}{L_r}$. In this way, the priority matrix for the tasks assigned to the virtual resource groups is

$$P=\begin{pmatrix} P(1,1) & P(1,2) & \dots & P(1,n) \\ P(2,1) & P(2,2) & \dots & P(2,n) \\ \dots & \dots & \dots & \dots \\ P(m,1) & P(m,2) & \dots & P(m,n) \end{pmatrix}=\begin{pmatrix} \frac{f_{(1,1)}}{L_1} & \frac{f_{(1,2)}}{L_2} & \dots & \frac{f_{(1,n)}}{L_n} \\ \frac{f_{(2,1)}}{L_1} & \frac{f_{(2,2)}}{L_2} & \dots & \frac{f_{(2,n)}}{L_n} \\ \dots & \dots & \dots & \dots \\ \frac{f_{(m,1)}}{L_1} & \frac{f_{(m,2)}}{L_2} & \dots & \frac{f_{(m,n)}}{L_n} \end{pmatrix} \tag{1}$$

The load balancing model with key resource relevance is shown as Figure 2:

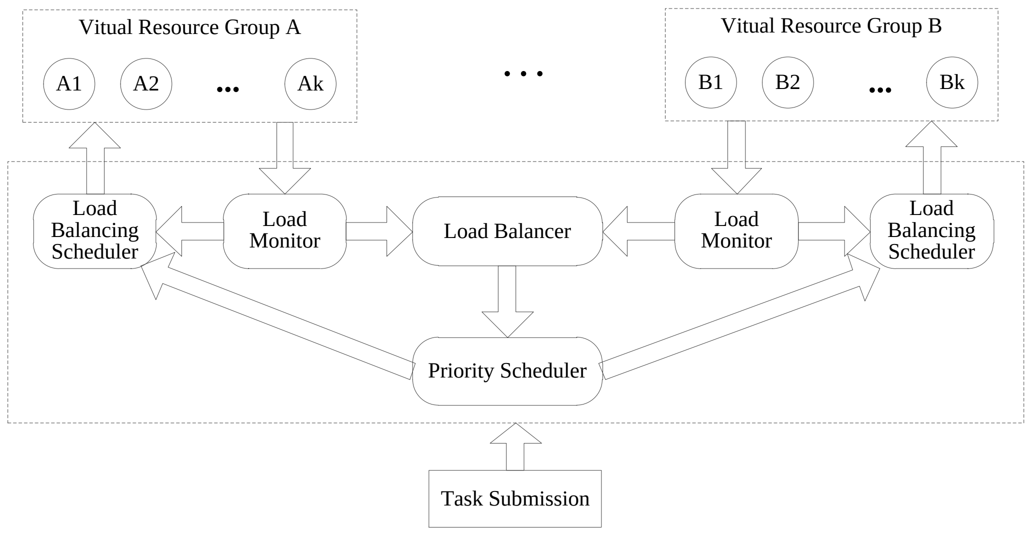

Figure 2. RRK-based System Model

The system model of RRK algorithm has several main elements as follows:

(1) Load monitor: Receive various kinds of load information periodically sent by each virtual resource, and then calculate its integrated load value. After observing that whether the load change magnitude exceeds the threshold value for a period of time, decide to upload this information to load balancing scheduler and the load balancer.

(2) Load balancing scheduler: Based on the integrated load value of each virtual resource calculated by the load monitor and on the weight value of each task, assign the task scheduled to this kind of resource group by priority scheduler to one certain virtual resource.

(3) Load balancer: According to the load value of each virtual resource in each kind of virtual resource group uploaded by load monitor, calculate the integrated load of this kind of virtual resource group.

(4) Priority scheduler: According to the relevance between task and virtual resource as well as the integrated load value of each virtual resource group, calculate the priority of the task to be assigned to each virtual resource group, and then assign the task to one certain virtual resource group.

4. Design of Load Balancing Algorithm with Key Resource Relevance

The load balancing algorithm with key resource relevance consists of following parts: collection strategies for load information, calculation of load value, inter-resource group priority assignment strategies and load balancing assignment strategies in resource groups.

4.1. Collection strategies for load information

There are two kinds of collection strategies for load information: “Push” and “Pull”. “Push” mode means that the monitored resource actively reports its own resource performance and load information to the system, while “Pull” mode means the system actively inquires the resource performance and load information from the monitored resource. Reference [19] proposes a kind of collection strategy that has combined these two modes together, namely the system periodically "pulls out" the resource performance and load information of each monitored resource. When the change of resource performance and load exceeds certain threshold values, it will actively "push" its performance and load information to the system.

In this paper, this kind of collection strategy combining "Push" & "Pull" modes together is also adopted, but the time to use the "Push" and "Pull" modes is different from that in Reference [19], as shown below in details:

(1) Each virtual resource sends its own load information to corresponding load monitor periodically.

(2) The load monitor compares this load information with the last load information. If the change exceeds the threshold value, it will upload this load information to the load balancing scheduler and load balancer.

(3) When a task is submitted, the priority scheduler inquires the load balancer about the integrated load value of each virtual resource group.

4.2. Feedback indexes

The feedback indexes are the core part of the dynamic load balancing strategy, which directly determines the effects of load balancing [20]. The task is generally compute-intensive and needs to be stored on the disk; besides, all the virtual resources are communicated with each other via the network, and hence the network branch width occupancy is also one factor affecting the load value. Consequently, the feedback indexes finally defined by the algorithm proposed therein include CPU utilization rate, memory utilization rate, occupancy rate of network band width and disk utilization rate.

4.3. Calculation of load value

As the resources in the virtual cluster system are normally heterogeneous [21], $\mathrm{r_r}=(\mathrm{r_{r1}} \quad \mathrm{r_{r2}} \quad \cdots \quad \mathrm{r_{rj}} \quad \cdots \quad \mathrm{r_{rn}})$ is adopted to represent the virtual resource vector in virtual resource group $\mathrm{r_r}$. $1 \leq \mathrm{j} \leq \mathrm{n_r}$ and $\mathrm{n_r}$ is the number of virtual resource in the virtual resource group $\mathrm{r_r}$.

To calculate the integrated load $L(r_{rj})$ of virtual resource r_{rj} in the virtual resource group r_r, the major concern lies in CPU utilization rate $R_{cpu}(r_{rj})$, memory utilization rate $R_{mem}(r_{rj})$, occupancy rate of network band width $R_{net}(r_{rj})$ and disk utilization rate $R_{disk}(r_{rj})$ of each virtual resource. The calculation formula (2) to calculate the integrated load of virtual resource r_{rj} in the virtual resource group r_r is shown as follows:

$$L(r_{rj}) = k_1 * R_{cpu}(r_{rj}) + k_2 * R_{mem}(r_{rj}) + k_3 * R_{net}(r_{rj}) + k_4 * R_{disk}(r_{rj})$$
$$j=1, 2, \ldots, n_r, \ \sum k=1 \qquad (2)$$

Where: $k_d, d = 1,2,3,4$ is the weight parameter of each index, reflecting the influence of different kinds of indexes on the load value, configured by users.

The integrated load L_r of one certain kind of virtual resource group refers to the load average value of all virtual resources in this virtual resource group, which is calculated via following formula (3):

$$L_r=\frac{\sum_{j=1}^{n_r} L(r_{rj})}{n_r}, \quad j=1, 2, \ldots, n_r \tag{3}$$

4.4. Inter-resource group priority scheduling strategies

After the user submits tasks, the priority scheduler will first assign them to a certain kind of virtual resource groups. This step fully considers the relevance between the tasks and virtual resource groups, and when the load of this certain virtual resource group exceeds the preset threshold value, this virtual resource group will not be considered, nor participate in the further assignment process. The algorithm is described as follows:

```
Input: A series of random tasks {$t_1$, $t_2$,…, $t_m$}, a set of virtual resource groups {$r_1$, $r_2$,…, $r_n$};
Output: Each task has been assigned to a virtual resource group;
For $i$=1 To $m$  // $m$ is the number of tasks;
{
  Update the load value of every virtual resource group;
  Delete the virtual resource groups of which load values exceed the threshold value from candidate queue;
  For $j$=1 To $n$  // $n$ is the number of virtual resource groups;
    Calculate the priority of task $t_i$ to be assigned to virtual resource group $r_j$;
  If the max priority value > 0
    Assign task $t_i$ to virtual resource group which has max priority;
  Else
    Prompt user the task $t_i$ has failed to be assigned;
}
```

Figure 3. Pseudo-code of inter-group priority scheduling

The priority scheduling process of single task is shown in following Figure 4.

(1) "Inquire" the load balancer about the load L_r of each virtual resource group, where: r=1, 2, …, n.

(2) Delete the virtual resource group with its load value exceeding the threshold value from the candidate queue.

(3) Use formula (1) to calculate the priority of the task to be assigned to each virtual resource group.

(4) Place the priorities in order. If the maximum is greater than 0, then the task will be assigned to the virtual resource group corresponding to the maximum priority.

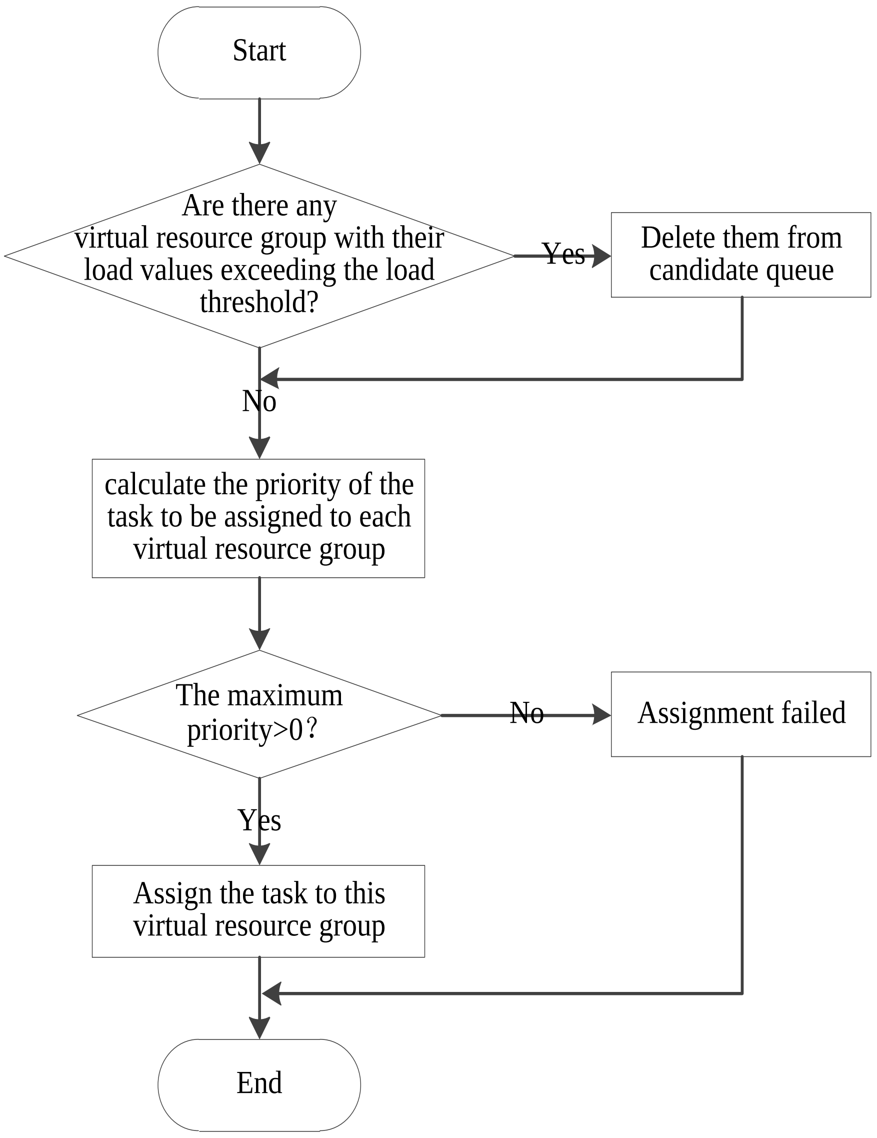
Start
Are there any virtual resource group with their load values exceeding the load threshold?
Yes
Delete them from candidate queue
No
calculate the priority of the task to be assigned to each virtual resource group
The maximum priority>0?
No
Assignment failed
Yes
Assign the task to this virtual resource group
End

Figure 4. Priority scheduling process

4.5. Load balancing scheduling strategies within a resource groups

After the priority scheduler assigns the task to one certain kind of virtual resource group, the load balancing scheduler of this virtual resource group will then assign this task to one of its virtual resources. Because the relevance of this task to each virtual resource in this one same virtual resource group is mutually identical, the difference only lies in the hardware performance. Thus, this step only considers the load value of each virtual resource and the weight value of the task in the assignment, instead of the relevance between the task and the virtual resource. The weight value of task t_i is already given when the user submits the task, which is to judge the complexity of task running. It may be an integral number within 1 to 5, and the smaller it is, the lower the weight value is. Due to the randomness of task arrival and the difference among task weight values, if the previous task with a lower weight value is assigned to the light-loaded virtual resource, and the weight value of the next task may be quite high with a need of a much light-loaded virtual resource. However, this much light-loaded virtual resource may have been already assigned, and the performance of other virtual resources is insufficient to run this task. At this point, this task can only wait for the proper virtual resources to be released [22]. Although there may be some virtual resources that can run this task, the extreme load imbalance of this virtual resource group will occur in case this heavy task has been assigned. For this reason, we will consider the task weight value during the load balancing scheduling carried out within the virtual resource groups.

Firstly, we classify the virtual resources into 5 grades depending on their load values, i.e. $\{g_1, g_2, g_3, g_4, g_5\}$. The calculation formula (4) for the load value difference between every two adjacent grades Δ_l is shown as follows, where N_r is the number of virtual resource in virtual resource group L_r.

$$\Delta_l = \frac{max\{L(r_{rj})\} - min\{L(r_{rj})\}}{5} \qquad (4)$$

$$1 \leq j \leq n_r$$

Then, the calculation formula (5) for the minimum load $L(g_i)$ on graded g_i is shown as follows:

$$L(g_i) \in [\Delta_l * (i-1) + min\{L(r_{rj})\}, \Delta_l * i + min\{L(r_{rj})\}) \qquad (5)$$

$$1 \leq i \leq 5$$

$$1 \leq j \leq n_r$$

When a task is assigned to the virtual resource group r_r, all virtual resources will be firstly classified into 5 grades according to their load values. Then, follow the task weight value to assign the task to well-matched virtual resource. If no virtual resource exists on the corresponding grade, then system will search for the virtual resource in a higher grade until it finds out the virtual resource that can satisfy the requirements. If not, the task will be added into the waiting queue; if there are multiple virtual resources in the corresponding grade, then the task will be assigned to the virtual resource with lowest load in this grade. The algorithm is described as follows:

```
Input: A series of random tasks {$t_1$, $t_2$,…, $t_k$}, a group of virtual resources;
Output: Each task has been assigned to a appropriate virtual resource;
Var: Succeed_Flag  : boolean   init : FALSE   // mark whether the task has been assigned successfully or not;
     Wait_Flag     : boolean   init : FALSE   // mark if the task is needed to be added to waiting queue or not;
For $i$=1 To $k$  // $k$ is the number of tasks;
{
    Update the load value of each virtual resource;
    Divide the virtual resources into five grades {$g_1$, $g_2$, $g_3$, $g_4$, $g_5$} from the minimum to the maximum load value;
    Succeed_Flag=Wait_Flag=FALSE;
    For $j$=1 To 5
    {
        If the weight value of $t_i$ corresponds to grade $g_j${
            If there are some virtual resources in grade $g_j${
                Assign $t_i$ to the virtual resource which has minimum load value on grade $g_j$;
                Succeed_Flag=TRUE;
                Break;
            }
            Else{
                If there is an appropriate virtual resource on grade $g_{i+1}$ to $g_5${
                    Assign $t_i$ to this virtual resource;
                    Succeed_Flag=TRUE;
                    Break;
                }
                Else{
                    Wait_Flag=TRUE;
                    Break;
                }
            }
        }// end If
    }// end For
    If !Succeed_Flag && Wait_Flag
        Add $t_i$ to the waiting queue;
    Else If !Succeed_Flag && !Wait_Flag
        Prompt user the weight value of $t_i$ is wrong, assignment failed;
}// end For
```

Figure 1. Pseudo-code of intra-group assignment

From the above, RRK algorithm is summarized below:

Step 1: Acquire integrated load of each virtual resource group, and delete the virtual resource group exceeding threshold value out of candidate queue;

Step 2: Considering the vector of relevance between task and virtual resource group, use formula (1) to calculate the priorities of the task to be assigned to each virtual resource group;

Step 3: Assign the task to the virtual resource group r_r with maximum non-zero priority;

Step 4: Update the load value of each virtual resource in virtual resource group r_r;

Step 5: Classify the virtual resources in virtual resource group r_r into 5 grades by their load values;

Step 6: Search for the virtual resource with minimum load on the grade corresponding to the weight value of task to run this task. If failed, turn to Step 7;

Step 7: Search for the proper virtual resource from current grade to Grade 5 to run this task. If failed, turn to Step 8;

Step 8: Add the task to waiting queue until there is a proper virtual resource to run this task.

5. Simulation Experiment and Performance Analysis

The purpose of the virtual cluster load balancing algorithm is to take the full advantage of the load and performance information of each virtual resource, so that the idleness possibility of some virtual resources can be reduced when there are a number of tasks in the system [23]. The key to the load balancing algorithm with key resource relevance (RRK) is considering assigning the task to the virtual resource group that has high relevance to this task, therefore the relevance-balance ratio $H(S)$ is introduce to measure the algorithm performance. The calculation formula of $H(S)$ is shown as follows:

$$H(S)=\frac{G(S)}{D(S)} \tag{6}$$

$G(S)$ refers to the sum of ratios between relevance of each task to its virtual resource group and relevance numbering position after m tasks have been assigned to n virtual resources groups. It calculation formula (7) is shown as follows:

$$G(S)=\sum_{i=1}^{m}\frac{C_i}{N_i} \tag{7}$$

Where: C_i is the relevance of task t_i to its assigned virtual resource group, N_i is the relevance numbering position in the vector of relevance between task and virtual resource groups. For example: The vector of relevance between task t_i and virtual resource group is $(0.1 \quad 0 \quad 0.6 \quad 0.3)$, and the algorithm assigns the task t_i to virtual resource group r_4. Thus, C_i=0.3, N_i=2.

$D(S)$ refers to the deviation among virtual resource groups after the task assignment is complete. Its calculation formula (8) is shown as follows:

$$D(S)=\sum_{i=1}^{n}\frac{(L_i-\bar{L})^2}{n} \tag{8}$$

Where: L_i is the integrated load value of virtual resource group r_i after the task assignment is complete; $\bar{L}$ is the average load value of all virtual resource groups. n is the number of virtual resource groups.

As from the definition of $H(S)$, in case the tasks and resources remain unchanged for each experiment, the greater $G(S)$ is, the higher the integrated relevance for the task to be assigned to virtual resource group is; the smaller $D(S)$ is, the more balanced the system load is. Hence, the greater $H(S)$ is, the better the algorithm performance becomes

The experiment makes a comparison among MTK algorithm, Weight algorithm and RRK algorithm proposed therein. The number of tasks is 100, and the experiment is conducted 30 times, respectively comparing $H(S)$ values of these three algorithms under the conditions of 10, 20 and 30 virtual resource groups.

Figure 6, Figure 7 and Figure 8 are the respective comparison diagrams for the relevance-balance ratios of MTK algorithm, Weight algorithm and RRK algorithm under the conditions of 10, 20 and 30 virtual resource groups. From these diagrams, when the number of virtual resource groups is 10, $H(S)$ value of RRK algorithm reaches its maximum for 20 times out of those 30 times of experiments. When the number of virtual resource groups is 20,

$H(S)$ value of RRK algorithm reaches its maximum for all 30 times. When the number of virtual resource groups is 30, the same thing happens again, and at this point the performance advantage of RRK algorithm seems more obvious. This means, the larger the number of virtual resource groups is, the more advantage RRK algorithm can present.

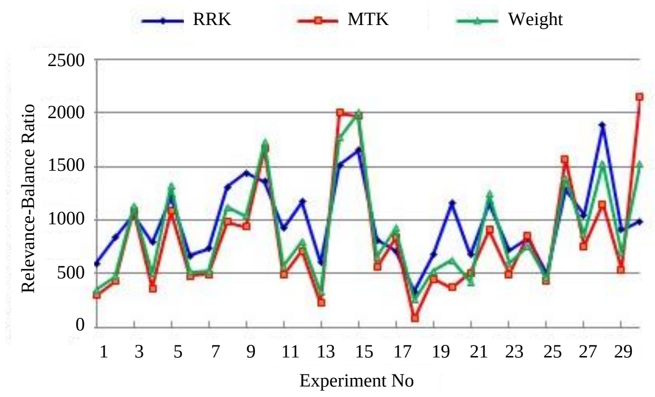

Figure 6. 10 Virtual resource groups

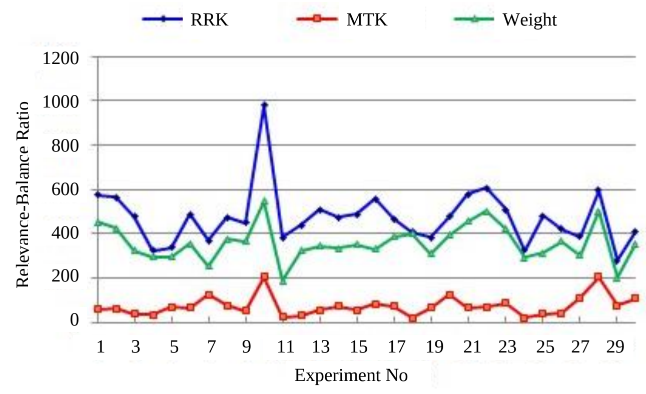

Figure 7. 20 Virtual resource groups

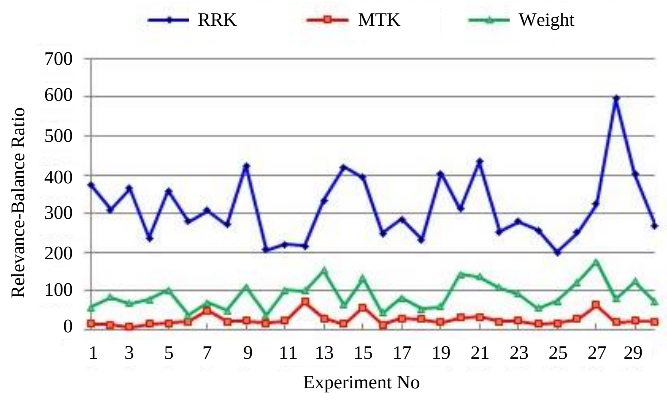

Figure 8. 30 Virtual resource groups

Within the virtual resource groups, the load balancing assignment strategies adopted by RRK algorithm take the weight value of task into consideration so as to predict the resource demand in task running and then assign proper virtual resource as required. This can reduce the average time for the tasks to wait the assignment. We use formula (9) to define the average waiting time of tasks:

$$t_{AvgW}=\frac{1}{m}\sum_{i=1}^{m}(t_{iE}-t_{iA}) \tag{9}$$

Where: m is the number of tasks; t_{iA} is the time for task t_i to arrive at virtual resource group; and t_{iE} is the time for task t_i to be assignd to one certain virtual resource.

In this experiment, we select the virtual resource group including 20 virtual resources. The distribution of initial load values is shown in Table1 below, and the numbers of virtual resources on each grade are shown in Table 2 below.

Table 1. Initial load values of virtual resources

Series No of Virtual Resource	Initial Load Value	Grade
1	0.13	2
2	0.21	3
3	0.45	5
4	0.06	1
5	0.29	3
6	0.51	5
7	0.32	4
8	0.16	2
9	0.23	3
10	0.09	1
11	0.45	5
12	0.20	2
13	0.36	4
14	0.15	2
15	0.03	1
16	0.25	3
17	0.12	2
18	0.34	4
19	0.01	1
20	0.28	3

Table 2. Numbers of virtual resources on each grade

Grade	Number of Virtual Resource
1	4
2	5
3	5
4	3
5	3

There are 1000 tasks to be assigned in this experiment. Once every 100 tasks have been assigned, we calculate the average time t_{AvgW} for all previously assigned tasks to wait. The experiment results of RRK algorithm proposed therein, MTK algorithm and Weight algorithm are shown as in Figure 9, from which it can be observed that: when the number of tasks is less than that of virtual resource groups, the average waiting time of RRK algorithm for the intra-group assignment is a little greater than that of other two algorithms. Because the tasks are so few at this point, the advantage of RRK algorithm on the virtual resource grading cannot be seen. However, as the tasks continuously increase, its average waiting time will be obviously less than that of MTK and Weight algorithms. This is because, as the tasks increase, a lot of tasks are assigned to the virtual resource groups corresponding to the weight values of these tasks, which then reduces the waiting possibility of tasks with high weight value.

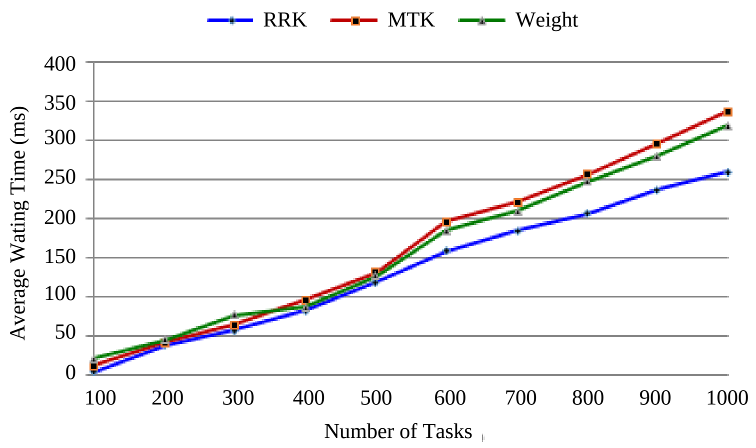

Figure 9. Average waiting time for intra-group task assignment

6. Conclusions

For the purpose of key resource relevance for load balancing algorithm in the virtual cluster system, this paper proposes a kind of load balancing algorithm with key resource relevance for virtual cluster, and have conducted a number of comparison experiments among RRK, MTK and Weight algorithms on the basis of different numbers of virtual resource groups. The experimental results show that: Compared with MTK and Weight algorithms, RRK algorithm proposed therein can assign the tasks to highly-relevant resources group whilst assuring better system load balance, and it performs better in case of more resource groups and more tasks.

In the future, we will consider extending the centralized scheduler herein into a distributed scheduler. And we will introduce multi-attribute decision-making theories to determine the weight coefficients of all load indexes, so that the artificial interference with the weight coefficients of all load indexes may be reduced.

Acknowledgements

The research is supported by the National High-Tech Research & Development Program of China (2008AA706103), the Aviation Science Funds of China (2010ZC13012), and Funding of Jiangsu Innovation Program for Graduate Education (CXZZ13_0171).

References

[1] J. Xin, K. Chen and W. Zheng, “Studies on virtualization of cluster resource management technology”, Journal of Frontiers of Computer Science and Technology, vol. 4, no. 4, **(2010)**, pp. 324-329.
[2] M. Amoon and H. M. Faheem, “Scheduling Jobs in Face of Status Update Timing of Resources in Computational Grids”, International Journal of Grid and Distributed Computing (IJGDC), vol. 5, no. 2, **(2012)**, pp. 33-42.

[3] J. Hai, D. Li, W. Song, S. Xuanhua and Z. Like, "Automatic Power-Aware Reconfiguration of Processor Resource in Virtualized Clusters", Journal of Computer Research and Development, vol. 48, no. 7, **(2011)**, pp. 1123-1133.
[4] X. J. Yang, Y. Dou and Q. F. Hu, "Progress and challenges in high performance computer technology", Journal of Computer Science and Technology, vol. 21, no. 3, **(2006)**, pp. 674-681.
[5] F. Yiqiu, F. Wang and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing", In Proceedings of International Conference on Web Information Systems and Mining, Sanya, China, **(2010)**, October, pp. 271-277.
[6] L. Yang, "Research for Key Technology on Virtual Server Consolidation", National University of Defense Technology, Changsha, China, **(2010)**.
[7] D. Yuxia, "Research on Scheduling Algorithm based on Grid", Shandong Normal University, Jinan, China, **(2011)**.
[8] S. Sharifian, S. A. Motamedi and M. K. Akbari, "A content-based load balancing algorithm with admission control for cluster web servers", Future Generation Computer Systems, vol. 24, no. 8, **(2008)**, pp. 775-787.
[9] X. Jiansheng, W. Yongji, L. Junxiang and Z. Haitao, "A Static Priority Assignment Algorithm with the Least Number of Priority Levels", Journal of Software, vol. 18, no. 7, **(2007)**, pp. 1844-1854.
[10] A. M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computer Systems", IJCSNS International Journal of Computer Science and Network Security, vol. 10, no. 6, **(2010)**, pp. 153-160.
[11] Q. Zhang, A. Riska, W. Sun, E. Smirni and G. Ciardo, "Workload-aware load balancing for clustered web servers", IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 3, **(2005)**, pp. 219-233.
[12] M. Beinkowski, M. Korzeniowski and F. der Heide, "Dynamic load balancing in distributed hash tables", Peer-to-Peer System IV, **(2005)**, pp. 217-225.
[13] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems", Journal of Parallel and Distributed Computing, vol. 65, no. 9, **(2005)**, pp. 1022-1034.
[14] S. Sharma, S. Singh and M. Sharma, "Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology, vol. 38, **(2008)**.
[15] L. Jian, X. Lei and Z. Wei-ming, "A Load Balancing Algorithm Based on Dynamic Feed-Back", Computer Engineering & Science, vol. 25, no. 5, **(2003)**, pp. 65-68.
[16] Z. Yu-fang, W. Qin-lei and Z. Ying, "Load Balancing Algorithm based on Load Weights", Application Research of Computers, vol. 29, no. 12, **(2012)**, pp. 4711-4713.
[17] S. Kumar and N. Singhal, "A Priority based Dynamic Load Balancing Approach in a Grid based Distributed Computing Network", International Journal of Computer Applications, vol. 49, no. 5, **(2012)**.
[18] S. Saeed, M. Seyed and A. Mohammad, "A content-based load balancing algorithm with admission control for cluster web servers", Future Generation Computer Systems, vol 24, no 8, **(2008)**, pp. 775-787.
[19] Y. Gang, Z. Xing-She and Y. Zhi-Yi, "An Adaptive Resource Monitor Approach for Distribute Computing Environment", Chinese Computer Conference Proceedings, Xian, China, **(2008)**, September, pp. 416-422.
[20] Z. Hui and H. Lianyue, "Dynamic Feedback Load Balancing Technology of Heterogeneous Distributed Storage System", Journal of Computer Research and Development, vol. 46, no. z2, **(2009)**, pp. 322-327.
[21] W. Jun, "Research on Key Technologies of Middleware based Load Balancing for Heterogeneous Distributed System", National University of Defense Technology, Changsha, **(2007)**.
[22] J. Naik, K. V. Kumar and N. Satyanarayana, "Scheduling Tasks on Most Suitable Fault tolerant Resource for Execution in Computational Grid", International Journal of Grid and Distributed Computing(IJGDC), vol. 5, no. 3, **(2012)**, pp. 121-132.
[23] E. Saravanakumar and P. Gomathy, "A Novel Load Balancing Algorithm for Computational Grid", International Conference on Innovative Computing Technologies, Karur-Tamilnadu, India, **(2010)**, February.

Authors

Xu Chaoqun received the B.S. degree from Hubei University of Technology in 2007. Now he is studying for the master degree at Nanjing University of Aeronautics & Astronautics. His research interest is Grid and Distributed Computing.

Zhuang Yi was born in 1957. She is a professor and Ph. D. supervisor of Nanjing University of Aeronautics & Astronautics. Her research interests include Distributed Computing and Information Security.

Zhu Wei received the B.S. degree from Northeastern University in 1996 and his M.S. degree in Computer Applications Technology from Huazhong University of Science & Technology in 2004. Currently, He is the research professor of Jiangsu Automation Research Institute, Lianyungang, China. His research interests include distribute systems, embedded systems, cloud computing and real-time control systems.