A Novel QoS-Aware Load Balancing Mechanism in Cloud Environment

Feng Ye[1], Shengyan WU[1], Qian Huang[1], Xu An Wang[2]
1 College of Computer and Information, Hohai University, Nanjing, China
2 Engineering University of CAPF, Xi'an, China
yefeng1022@hhu.edu.cn

Abstract— **Efficient low-level resource provisioning and QoS guaranteed are key challenges for cloud computing. When using virtual machines cluster to tackle various tasks scheduling, the target is to assign the tasks to each of the available nodes evenly in in premise of ensuring QoS of services, and it also means that the cloud providers should consider to reduce the load of overload nodes, and improve resource utilization of under-loading nodes. There are some limitations when applying these classic scheduling algorithms to the cloud computing environment. In order to solve this problem, we propose a novel QoS-aware load balancing mechanism in cloud environment. The key of this mechanism includes QoS model, resource model, and task model. We conduct a CloudSim based experiment to evaluate our method using a realistic dataset, and the results show that the algorithm proposed effectively shortens the waiting time in comparison to RR algorithm and Max-Min algorithm.**

Keywords—task; scheduling; Cloudsim; load balancing

I. INTRODUCTION

Due to the critical characteristics of elasticity, quality of service (QoS) guaranteed and on-demand resource provisioning model, cloud computing [1-3], has received more and more attention and been adopted in both the academia and the industry. Efficient low-level resource provisioning and QoS guaranteed are key characteristics for cloud computing. For cloud providers, the suitable load balancing [4] mechanisms are important to them, which can reduce the load of overload nodes and improve resource utilization of under-loading nodes. At present, many open-source or commercial cloud solutions have integrated one or more load balancing mechanism, for example: the round robin algorithm and green load balancing algorithm are provided in Resin 4.0; Microsoft Azure implements round-robin load balancer; GoGrid use redundant f5 hardware load balancers, and the load balancer can be configured in terms of round robin or least connect to use [5]. And there are many other classic scheduling algorithms [6], but not each algorithm can apply to all of the specific cloud service scenarios. Therefore, we propose a novel QoS-aware load balancing mechanism in cloud environment, and the target is to shorten the total waiting time of the tasks in premise of QoS guarantee.

The rest of this paper is organized as follows. Firstly, in section II, the related works are introduced. In section III, the QoS model, the task model and resource model are introduced. Nextly, the load balancing mechanism is discussed in details in Section IV. Then, we conduct an experiment to evaluate our method using a realistic dataset in section V. At last, the summary is given.

II. RELATED WORKS

Classic scheduling algorithms include: Round Robin (RR) algorithm, Opportunistic Load Balancing (OLB) algorithm, Minimum Execution Time (MET) algorithm, Minimum Completion Time (MCT) algorithm, Min-Min algorithm and Max-Min algorithm. RR algorithm assigns the tasks to each node in turn without considering the status of resource of the nodes and the execution time of the task; OLB algorithm is similar to RR algorithm, and it assigns the tasks to each node randomly without considering the status of resource of the nodes and the execution time of the task. MET algorithm assigns the tasks to the corresponding node according to the expected execution time of the task, and the nodes provide all of the resources for the task, so it may accelerate the overall completion time of all tasks. However, obviously, it also leads to serious unbalanced load of the clusters. In the case of the current load of the nodes, MCT algorithm assigns tasks according to minimum completion time, but this scheduling method can assign only one specific task at a time. Min-Min algorithm is suitable for the task as a group: the load balancer calculates the completion time of each task on every node at first, and selects the task and node with minimum completion time for task scheduling. Then, the load balancer repeats this process until all the tasks completed. But we can see this algorithm always assign the "small" tasks preferentially. Max-Min algorithm is similar to Min-Min algorithm, but the policy of Max-Min algorithm is to assign the "large" tasks preferentially. In short, the load balancer calculates the completion time of each task on every node at first, and selects the task and node with maximal minimum completion time for task scheduling. Then, the load balancer repeats this process until all the tasks completed. In Max-Min algorithm, "small" tasks often need to wait for "large" tasks are completed to perform, thus the latency will increase apparently.

There are some limitations when applying these algorithms to the cloud computing environment. For example, though RR algorithm and OLB algorithm are easy to implement, because they cannot consider the status of resource of the nodes and the task's characteristics, the effect of load balancing is difficult to guarantee. For cloud computing environment, RR algorithm and OLB algorithm are not the optimal choices. Though MET algorithm and MCT algorithm consider the diversity of tasks, both of them are suitable for assigning individual task. Currently, Max-Min algorithm and Min-Min algorithm are the most widely studied. According to various application scenarios and targets, some research works [7-8] improve Max-Min algorithm and Min-Min algorithm for task scheduling.

In sum, in cloud computing environment, it is necessary to study appropriate QoS-Aware load balance mechanism.

III. The Related Model

The QoS-Aware load balancing mechanism proposed should consider the status of resource of the nodes, the characteristics and QoS requirement of the tasks. Therefore, we should construct QoS model, task model and resource model at first.

A. *QoS Model*

The response time, availability, pay-per-use, cost and reputation are critical elements in cloud computing, therefore the QoS model could be defined as follows [9]:

1) *Response Time*: The guaranteed average time required to complete a cloud service request.

2) *Process Time*: It is a measure of the time that a cloud service takes between the time it gets a request and the moment it sends back the corresponding response.

3) *Availability*: It represents the probability of cloud services can be accessed, and it is estimating using mean time between failure (MTBF) and mean time to repair (MTTR).

4) *Reputation*: The property is obtained from feedback of service users.

5) *Cost*: It is a measure of the cost involved in requesting the cloud services.

According to the QoS model, we can define the priority based on one or more attributes for the services or tasks.

B. *Resoure Model*

Resource model is used to describe the processing ability of the virtual machine node in cloud. Different nodes have different virtual configuration, so their processing abilities are not the same. For the same task, there will be differences between the execution time at different nodes. Moreover, different tasks have different QoS requirements, so the resource of the nodes should match the demand.

Generally speaking, the most representative resources are computing resource, memory resource, network resource, storage resource and so on. Therefore, here we define the resource of node $\boldsymbol{i}$ using vector $\boldsymbol{K_i}$, and $\boldsymbol{K}_i = [\,Kcpu_i, Kmem_i, Kdisk_i, Knet_i]$, where: $\boldsymbol{K_{cpu}}$, $\boldsymbol{K_{men}}$, $\boldsymbol{K_{disk}}$ and $\boldsymbol{K_{net}}$ represent respectively processing capacity of CPU, memory capacity, hard drive capacity and network bandwidth.

C. *Task Model*

Because the tasks are complex, to simplify the task model, we make some assumptions as follows.

1) The tasks are independent of each other;

2) The priority of the task only depends on one attribute, such as the response time;

3) We can estimate resource consumption by task type;

4) Once the task is assigned to the virtual node, it can be processed immediately, and the tasks which are assigned to the same node can be processed in parallel.

Next, we give some definitions. Vector $\boldsymbol{J_i}$ is used to represent the resource that is consumed by the task $\boldsymbol{i}$, and $\boldsymbol{J}_i = [\,Jcpu_i, Jmem_i, Jdisk_i, Jnet_i]$, where: $\boldsymbol{J_{cpu}}$, $\boldsymbol{J_{mem}}$, $\boldsymbol{J_{disk}}$ and $\boldsymbol{J_{net}}$ represent respectively the numbers of CPU instruction, use of memory, use of hard drive capacity and the amount of data transferred. We define weight of the resource using vector $\boldsymbol{W}$, and in different applications, $\boldsymbol{W}$ can be assigned to different values.

Therefore, we can further define $\boldsymbol{H}$ as the execution time of task $\boldsymbol{i}$ on the virtual node $\boldsymbol{j}$, and $\boldsymbol{H}=(\boldsymbol{J_i} \times \boldsymbol{W^T}) / (\boldsymbol{K_i} \times \boldsymbol{W^T})$.

IV. The Load Balancing Algorithm

There are two tables in load balancer, which are used for task scheduling: one is task status table and the other is virtual node state table. For task status table, it can be defined as a six-tuple $\boldsymbol{S}$, and it includes six elements: task id (task_id), node id (node_id), arriving time of the task(arriving_time), execution time of the task(execution_time), the completed execution time of the task (completed_time), and the last updating time (updating_time).

$\boldsymbol{S}$ = < task_id, node_id, arriving_time, execution_time, completed_time, updating_time >

For virtual node state table, it can be defined as a quintuple $\boldsymbol{V}$, and there are five elements in $\boldsymbol{V}$, namely: node id (node_id), the number of tasks which are processing (numbers), the total time duration of all tasks which have not been processed (time_duration), the status of node (status) and the last updating time (updating_time).

$\boldsymbol{V}$ = <node_id, numbers, time_duration, status, updating_time >

The main idea of the algorithm proposed is that: when a group of the tasks arriving, the load balancing firstly check the priority of the tasks and assigns the tasks with highest priority to the node with the largest $\boldsymbol{K}$, and other tasks with some priority will be tackled in a similar fashion, in order to meet specific QoS requirements; then, the load balancing updates the virtual node state table. If the priorities of all tasks are the same, the load balancing firstly choose the task with maximum execution time, and calculate the expected completion time in each node according to virtual node state table. Then, the load balancer assigns the tasks to the node which can meet the minimum completion time. Similarly, at the same time, the load balancing updates the related attributes in the virtual node state table. The load balancer repeats this execution until all tasks to be assigned.

It's not hard to see that the critical steps are calculating the status of the tasks and updating task status table and virtual node state table. On this basis, we can implement task scheduling. In addition, Agent [10] technology can help us to acquire or collect the related data.

A. The Algorithm for Updating Status Tables

Updating the task status table means removing the completed tasks from the task status table and modifying the completed_time attribute in the task status table. Specially, if the task is still in the processing, the value of completed_time attribute should be increased. If the value of completed_time are larger than the value of execution_time attribute, that means the execution time of the task is underestimate.

The virtual node state table depends on the task status table. When the task status table change, the virtual node state table will recalculate the numbers and time_duration attribute and update them.

Assuming the current point in time is ***t***. The algorithm for updating status tables can be described as follows according to the analysis above.

1) Remove the information of the completed tasks from the task status table;
2) For any task ***i*** in the processing, if (completed_time> execution_time), set execution_time = 2* execution_time; and set completed_time = completed_time+ ((t- updating_time)/ numbers).
3) Set updating_time = t;
4) For any task ***i*** in the processing, repeat 2) and 3).
5) Set number=number +1 and time_duration = time_duration + completed_time.
6) For any task ***i*** in the processing, repeat 6).

B. The Task Scheduling Algorithm

Assuming ***T*** is the set of tasks when arriving at t. We define ***F*** as expected completion time when task ***i*** in node ***j***. In order to simplify the calculation, ***F*** can approximate as that:

1) if ***H***< = (time_duration/numbers),

$$F = (\text{numbers}+1) \times H$$

2) if ***H***>(time_duration/numbers),

$$F = ((\text{numbers}+1)/\text{numbers}) \times \text{time\_duration}+(H-(\text{time\_duration}/\text{numbers}))$$

The task scheduling algorithm can be described as follows.

1) Sorting all the tasks in ***T*** according to execution time;
2) Choose the task ***i*** which has the largest execution time;
3) Calculate ***F*** for task ***i,*** and choose the node ***j*** which has the smallest value of ***F***.
4) Assign task ***i*** to node ***j***.
5) Update the task status table and the virtual node state table with S_i = [i,j,t,0,H,t] and numbers=numbers+1, time_duration=time_ duration + completed_time in V_j.
6) Repeat step 2)- step 5) until T= null

V. Experiment

In order to verify the load balancing mechanism proposed, we utilize CloudSim [11] and a realistic data set [12] for experiment.

There are six data subset and we firstly explore them using statistical analysis, and the data is shown as Table 1. Based on the CloudSim, we set 50 same nodes.

	Group of Tasks	Total Number	The Total Tasks	Total Time/s	The Average Number of Tasks	The Average Amount of Tasks
1	379	10716	26854722	1418400	28.274	70856.79
2	625	20105	46157466	2653200	32.168	73851.95
3	817	29576	59830059	3391200	36.201	73231.41
4	1016	40217	76323057	4111200	39.583	75121.12
5	1268	50072	95396331	5155200	39.488	75233.71
6	1456	59701	1.16E+08	5839200	41.003	79809.18

In this experiment, we respectively use RR algorithm, Max-Min algorithm and our method (ECMM). The results are shown in Table 2 and Figure 1.

	Average Waiting Time /s		
	RR	MM	ECMM
Thunder01	1560.871	222.495	155.561
Thunder02	2211.931	1218.488	957.505
Thunder03	1689.074	852.434	666.217
Thunder04	1538.206	665.085	510.501
Thunder05	1502.758	553.495	420.921
Thunder06	1574.684	504.486	381.405

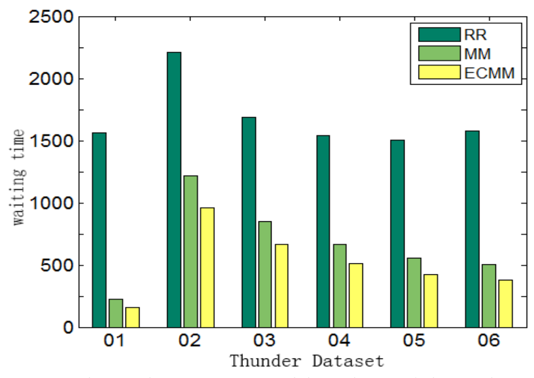

Fig.1 The average waiting time of the tasks

From Fig.1, we can see that ECMM effectively decrease the average waiting time of the tasks, and the average waiting time of the tasks are shorten 21.4%~30%. That means the nodes can tackle all the tasks more quickly.

VI. Conclusion

In the paper, we propose a novel QoS-aware load balancing mechanism in cloud environment, and by CloudSim based experiment, this method is proved to be effective.

In the future, we will carry out more experiments to verify the efficiency of and applicable scenario of the task scheduling method.

Acknowledgment

This work is supported by (1) National Natural Science Foundation of China (61300122); (2) A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions; (3) Water Science and Technology Project of Jiangsu Province (2013025); (4) Central University of basic scientific research and business fee of Hohai University (2009B21614)

References

[1] R. Buyya, J.Broberg, A.Goscinski (Editors), Cloud Computing: Principles and Paradigms. Hoboken, USA: John Wiley &Sons, 2011.

[2] L. Wang, R.Ranjan, J.Chen, et al (Editors), Cloud Computing: Methodology, Systems and Applications. Boca Raton, USA: CRC Press, 2012.

[3] G. Feng, R. Buyya, "Maximum revenue-oriented resource allocation in cloud," *International Journal of Grid and Utility Computing*, Vol. 7, No. 1, pp.12-21, 2016.

[4] Y. Wen, C. Chang, "Load balancing consideration of both transmission and process responding time for multi-task assignment," *International Journal of Space-Based and Situated Computing*, Vol. 4, No. 2, pp. 100-113, 2014.

[5] F. Ye, Z. Wang, F. Xu, et al, "A Novel Cloud Load Balancing Mechanism in Premise of Ensuring QoS," *Intelligent Automation & Soft Computing*, Vol. 19, No. 2, pp.151-163, 2013.

[6] T. Kokilavani, G.A. DI, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing," *International Journal of Computer Applications*, Vol. 20, No.2, pp.42-48, 2011.

[7] X. He, X. Sun, G. Laszewski, "QoS guided Min-Min heuristic for grid task scheduling," *Journal of Computer Science and Technology*, Vol.18, No.4, pp.442-45,2003.

[8] M.Singh, P K.Suri. QPS Max-Min, Min-Min: A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid[J]. Information Technology Journal, 2008, 7: 1176-1181.

[9] K.M. han. Managing Web Service Quality: Measuring Outcomes and Effectiveness, New York: Information Science Reference, 2009.

[10] M. Mountzia, G.D. Rodosek, "Using the Concept of Intelligent Agents in Fault Management of Distributed Services," *Journal of Network and Systems Management*, Vol. 7, No. 4, pp. 425-444, 1999.

[11] R.N. Calheiros, R. Ranjan, A. Beloglazov, et al, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*,Vol.41,No.1,pp.23-50, 2011.

[12] M. Jette. Parallel Workloads Archive: LLNL Thunder, http://www.cs.huji.ac.il/labs/parallel/workload/l_llnl_thunder/index.html