Enhancing Overall Performance of Cloud Computing with the MMALB Algorithm

Kuen-Fu Huang

Department of Information Management,

Chaoyang University of Technology

168, Jifong East Road, Wufong Township, Taichung County 41349,

Taiwan, R.O.C.

azsxdcfv7200000@gmail.com

Yuan-Liang Tang*

Department of Information Management,

Chaoyang University of Technology

168, Jifong East Road, Wufong Township, Taichung County 41349, Taiwan, R.O.C.

**yltang@cyut.edu.tw (Corresponding author)*

Abstract :

With the advance of cloud computing technologies, on-demand, self-service, and flexible computing resources are provided on the Internet and the users can access these resources very easily as long as they are connected to the network. For a cloud computing provider, to be able to serve a large amount of users, it is necessary to achieve high overall performance by organizing a lot of work nodes over the network. However, to manage these nodes and schedule jobs among them require tremendous efforts. In this paper, we propose a novel job scheduling algorithm which can decrease the total completion time and the average completion time, and also increase the load balancing and resource utilization.

Keywords-*cloud computing; scheduling algorithm; load balancing*

I. INTRODUCTION

Cloud computing has recently received a lot of attentions and discussions[1]. With cloud computing technologies, on-demand, self-service, and flexible computing resources are provided on the Internet and the users can access these resources very easily as long as they are connected to the network[2][3]. For a cloud computing provider, to be able to serve a large amount of users, it is necessary to achieve high overall performance by organizing a lot of work nodes over the network[4]. However, to manage the work nodes and schedule jobs among them require tremendous efforts. Many researchers have proposed algorithms to deal with the scheduling problem, and the approaches can be categorized into the following types: opportunistic load balancing scheduling, min-min scheduling, max-min scheduling, and Resource Aware Scheduling Algorithm [5□8].

The opportunistic load balancing (OLB) scheduling ensures every node is working and there is no idle nodes. The tasks are assigned to nodes in a random manner and every node will be assigned unless the number of

tasks is less than that of nodes. Although the algorithm is very simple, OLB scheduling does not consider the factors of a node's load and a task's execution time, so the overall performance is very poor.

The objective of min-min scheduling is to assign tasks to appropriate nodes in order to reduce each node's completion time. Supposing the completion times of node *i* is denoted C_i, the makespan, *m*, is defined as the greatest completion time among all nodes:

$$m=maxC_i \quad (1) \quad i$$

Before assigning tasks, the completion times of every task on each node is estimated. And then from the smallest task to the largest task, the task is assigned to the node with the minimum completion time, hence the name *minmin* scheduling. Due to the fact that small tasks are assigned before large tasks, large tasks will have more waiting time. In addition, some low computation power nodes may not be assigned tasks, which results in inefficient utilization of resources.

Max-min scheduling is the same as the min-min scheduling, except that the tasks are assigned in the order from large to small. Since large tasks are assigned before small tasks, small tasks will have more waiting time. Also, the same as min-min scheduling, some low computation power nodes may not be assigned tasks, which results in inefficient utilization of resources.

The resource aware scheduling algorithm (RASA) tries to solve the problems of the min-min and max-min scheduling by combing the two in order to achieve better performance. In RASA, the min-min scheduling and the max-min scheduling is use in turn and in alternate manner during task assignment for the purpose of leveraging the advantages of the two scheduling. However, some low computation power nodes may not be assigned tasks, too, which also results in inefficient utilization of resources.

One of the shortcomings of the above algorithms is that they only consider the *total completion time* of each node, another important factor, *load balancing* among work nodes, is generally overlooked. In this paper, we will indicate that load balancing will affect the overall performance, and we propose a novel scheduling algorithm called the *Minimum Makespan and Load Balancing* (MMALB) scheduling to enhance the overall performance. The rest of this paper is organized as follows. Section 2 presents in detail the proposed MMALB scheduling algorithm. Section 3 illustrates the experimental results and compares the results with those of other algorithms. Finally, Section 4 gives some concluding remarks.

II. THE MMALB SCHEDULING ALGORITHM

The MMALB scheduling takes into account both of the makespan and load balancing factors during task assignment in order to achieve better performance. The fundamental idea is that without load balancing, some resources may stay busy for a long time, whereas other resouces may stay idle, which results in inefficient utilization of resources. The algorithm consists of three phases: Phase I executes load balancing, Phase II lowers the makespan through tasks migration, and Phase III reduces the waiting times of tasks. Each phase is discussed in detail in the following.

In Phase I, a hybrid strategy is incorporated for assigning tasks, in which the max-max and min-min strategies is applied alternately. First, the task with the *maximum* execution time among all the remaining tasks is assigned to the node currently having the *maximum* computing capacity. Next, the task with the *minimum* execution time among all the remaining tasks is assigned to the work node currently having the *minimum* computing capacity. And then, these two strategies are applied in turn and alternately until the task assignment is completed. As a result, large tasks are assigned to high computing capacity nodes, whereas small tasks are assigned to low computing capacity nodes. Hence, the aim of the first phase is to ensure load balancing among all work nodes in order to maximize the resource utilization. In addition, every node has to receive a task before a certain node can receive another task. Such a mechanism achieves even more perfect load balancing, and hence better overall performance is expected. The complete algorithm of Phase I is listed in Figure. 1, with definitions of notations explained in Table I.

TABLE I. THE DEFINITIONS OF NOTATIONS

Notation	**Definition**
i	Task ID
j	Node ID
C_{ij}	Estimated completion time of task i on node j
E_{ij}	Execution time of task i on node j
R_j	Ready time of node j
T_k	The target task
N_k	The target node
T_q	The task to be exchanged with the target task

```
# Compute the completion time (tasks vs. nodes)
1     For every task, Ti, in taskset
2     For every node, Nj, in nodeset
3     Cij = Eij + Rj
4     End For
5     End For
  # Assigning tasks to nodes 6 While taskset is not empty
7     Select the largest task Tk in taskset
8     Select the highest computing capacity node Nk in nodeset
9     Assign Tk to Nk
10    Remove Tk from taskset
11    Update Nk's ready time Rk
12    Update Cik for all Ti's
13    Remove Nk from nodeset
```

14	If nodeset is empty
15	Add all nodes to the nodeset
16	Go to Step 7
17	End If
18	Select the smalles task T_k in taskset
19	Select the lowest computing capacity node N_k in nodeset
20	Assign T_k to N_k
21	Remove T_k from taskset
22	Update N_k's ready time R_k
23	Update C_{ik} for all T_i's
24	Remove N_k from nodeset
25	If nodeset is empty
26	Add all nodes to nodeset
27	Go to Step 7
28	End If
29	End While

Figure 1. Algorithm of Phase I

The objective of Phase II is to reduce the makespan by migrating tasks to appropriate nodes. Since the main objective of Phase I is to balance the loads, a node with high computing power may be assigned too many tasks, resulting in increased overall makespan. It is necessary, therefore, to examine the whole system and reduce the makespan. Suppose the current makespan is m. The procedures are as follows. First, select the node, N_k, with the maximum completion time, and then select the smallest task, T_k, on N_k. Second, find the node, N_p, that can provide minimum completion time for T_k. Third, if the resulting makespan, m_k, after migrating T_k to N_p is less than m, then perform migration. Otherwise, find one or more tasks, T_q, on a same node N_r, that, if we swap T_k with T_q, the resulting makespan, m_q, is less than m and m_q is the smallest. Finally, if T_q is found, perform the task swapping. The above procedure is repeated by finding the next smallest T_k on N_k, until all the nodes with current greatest makespan are explored. The algorithm of Phase II is listed in Figure. 2.

1	While makespan can still be lower
2	Find N_k with greatest makespan
3	Find smallest task T_k on N_k
4	Find node N_p that can have lowest completion time for T_k
5	If $m_k < m$ after migrating T_k to N_p
6	Migrate T_k to N_p
7	Update the ready time of both N_k and N_p
8	Else
9	Find T_q on node N_q that, if swapping T_k and T_q, $m_q < m$ and

10	$m_k < m$
11	If T_q is found
12	Swap T_k and T_q
13	Else
14	Find next smallest T_k on N_k
15	Go to Step 4
16	End If
17	End If
18	End While

Figure 2. Algorithm of Phase II

In Phase III, we adopted the concept of the RASA scheduling and made some improvements. The purpose is to reduce the waiting of the tasks in a node by rearranging the execution order on that node. For every node, the tasks in that node is rearranged as: largest, smallest, second largest, second smallest, third largest, third smallest, and so on. Such an order will minimize the waiting times of both large and small tasks.

III. EXPERIMENTAL RESULTS AND ANALYSIS

In order to compare the performances of the MMALB scheduling with those of other scheduling algorithms, we randomly generated 10 tasks and 5 working nodes for testing the MMALB, min-min, max-min, and RASA scheduling algorithms. Tables II, III, and IV list the parameters of the tasks, the parameters of the work nodes, and task execution time on different nodes, respectively.

TABLE II. PARAMETERS OF TASKS

Task	**The number of instructions (million)**
1	51
2	67
3	253
4	425
5	449
6	506
7	642
8	771
9	782
10	810

TABLE III. PARAMETERS OF WORK NODES

Node	Millions of instructions per second
1	18
2	55
3	77
4	82
5	100

Table IV. Task vs node execution times

Task	Node				
	1	2	3	4	5
1	2.83	0.93	0.66	0.62	0.51
2	3.72	1.22	0.87	0.82	0.67
3	14.06	4.6	3.29	3.09	2.53
4	23.61	7.73	5.52	5.18	4.25
5	24.94	8.16	5.83	5.48	4.49
6	28.11	9.2	6.57	6.17	5.06
7	35.67	11.67	8.34	7.83	6.42
8	42.83	14.02	10.01	9.40	7.71
9	43.44	14.22	10.16	9.54	7.82
10	45	14.73	10.52	9.88	8.10

This study use makespan (m), average completion time (c_a), load difference (l_d), average resource utilization (u_a) as four indicator to compare the scheduling algorithm. They are computed as follows:

$$m = \max C_i \quad (2) \quad i$$

$$c_a = (T_l + T_s)/2 \quad (3)$$

$$ld = Cl \square Cs \quad (4)$$

$$u_a = \frac{\sum_{i=1}^{N} C_i}{Nm} \times 100\% \quad (5)$$

where T_l and T_s are the largest and smallest tasks, respectively, C_l and C_s are the largest and smallest completion times among all nodes, respectively, C_i is the completion time of node i, and N is the total number of nodes. For an efficient data center, m, c_a, and l_d should be as small as possible, and u_a should be as large as possible.

The experimental results of min-min, max-min, and RASA, and the proposed MMALB scheduling algorithms are show in Figures. 3~6, respectively.

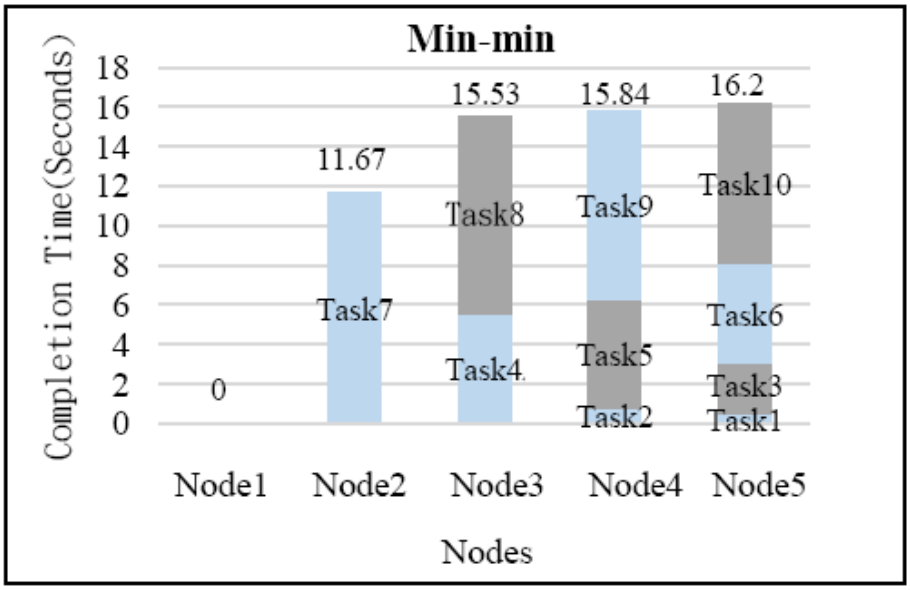

Figure 3. Performance of min-min scheduling

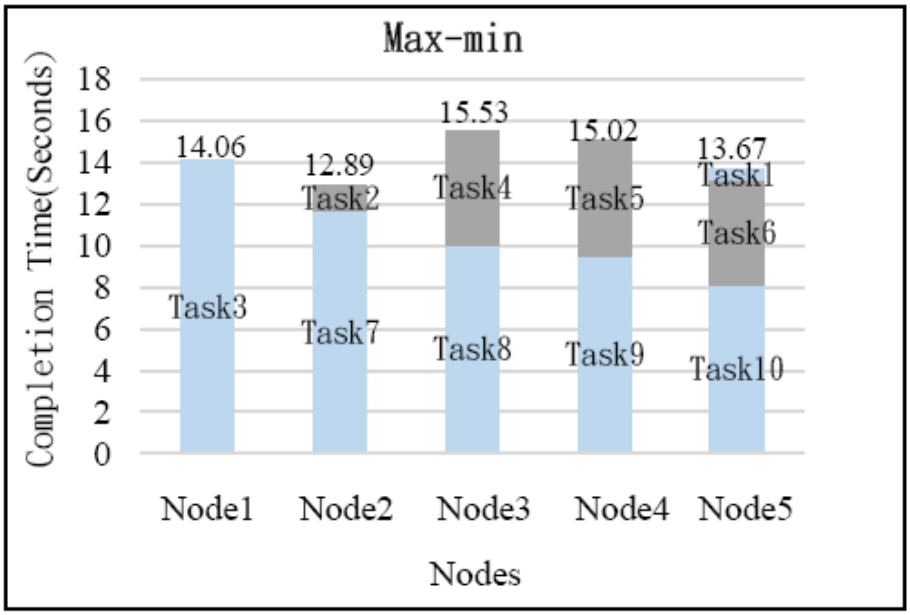

Figure 4. Performance of max-min scheduling

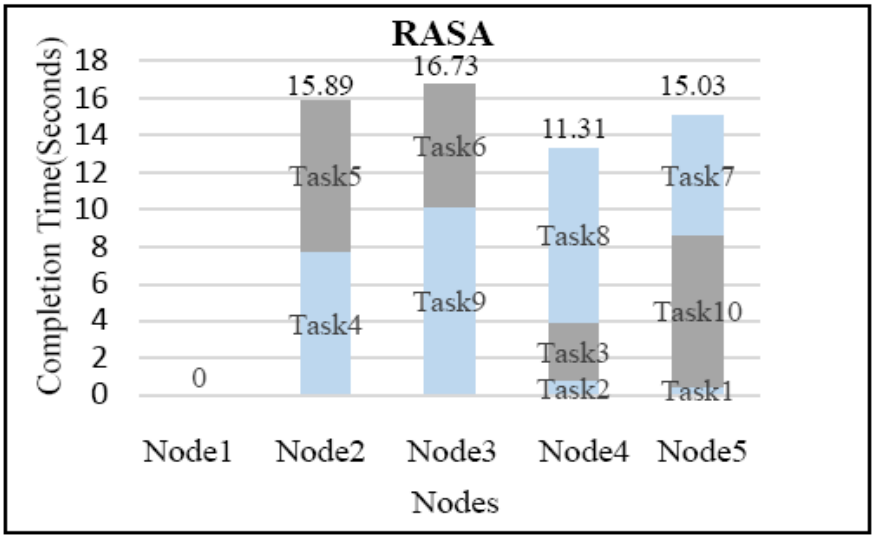

Figure 5. Performance of RASA scheduling

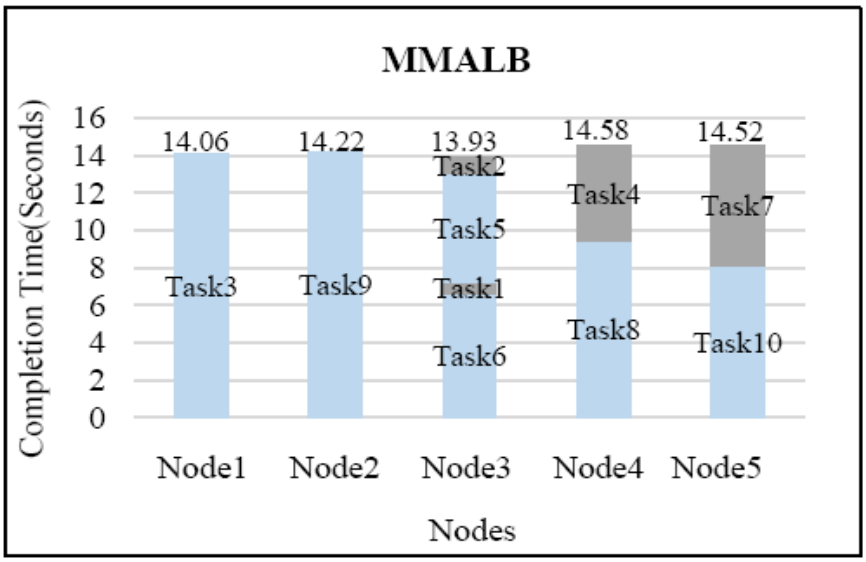

Figure 6. Performance of MMALB scheduling

The performances of all the four algorithms are summarized in Table V.

TABLE V. SUMMARY OF ALGORITHM PERFORMANCE

	m	c_a	l_d	u_a
Min-min	16.2	8.36	16.2	73.14%
Max-min	15.3	10.89	2.64	91.65%
RASA	16.73	4.31	16.73	70.48%
MMALB	14.58	7.76	1.65	97.82%

From Table V, some observations may be drawn:

(a) In the respect of makespan, our algorithm achieves the lowest value and it increases the efficiency by 12.85% comparing to the highest makespan.

(b) For the average completion time, our algorithm outperforms the min-min and max-min scheduling. RASA performs better because it does not consider load balancing and utilization.

(c) In the respect of load difference, our algorithm outperforms all others, which means our algorithm achieves the best load balancing and best resource utilization.

(d) In the respect of resource utilization, our algorithm still outperforms all others, which means we make best utilization of resources.

IV. CONCLUSIONS

In this paper, we propose the MMALB scheduling algorithm, which takes into consideration the factors of makespan, average completion time, load balancing, and resource utilization. By thorough analysis on the architecture of the cloud computing and careful design on the steps in each phase, our algorithm outperforms the other algorithms by a large margin. The experimental results also demonstrate that our algorithm can achieve low makespan, low average completion time, low load difference, and high resource utilization.

REFERENCE

[1] Michael Armbrust, Arm ando Fox, Rean Griffith, Anthony D. Joseph, Randy katz, Andy k onwinski, Gungo Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, " A View of Cloud Computing," Communications of the ACM CACM Homepage archive, VOL.53, Issue.4, pp.50-58,April, 2010.

[2] Aaron Weiss, "Computing in the cloud," netWorker, Magazine, pp.16-25, VOL.11, Issue.4, December, 2007.

[3] Giles Hogben,"Cloud computing- benefits, risks and recommendations for information security," European Network and Information Security Agency (ENISA), November, 2009.

[4] Shirley Radack **,** "Cloud Computing: A Review of Features, Benefits, and Risks, and Recommendations for Secure, Efficient Implementations," National Institute of Standards and Technology**(**NIST) , pp.1-7, 27 June, 2012.

[5] Brauna, T.D., Siegelb, H.J., Beckc, N.,Bölönid L.L., Muthucumaru Maheswarane, Albert, I.R., Robertsong, J.P., Theysh, M.D., Yaoi, B., Hensgenj, D. and Freundk,R.F., "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," Journal of Parallel and Distributed Computing , Vol. 61, Issue 6 , pp. 810-837, 2001.

[6] Devipriya, S., Ramesh, C.,"Improved Max-min Heuristic Model For Task Scheduling in Cloud," International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), pp.883-888, 12-14 Dec, 2013.

[7] Huankai Chen, Wang, F, Helian, N., Akanmu, G.,"User-Priority Guided Min-min Scheduling Algorithm For Load Balancing in Cloud Computing" 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), pp.1-8, 21-23 Feb, 2013.

[8] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, VOL.1,pp.108-113, 9-11 July, 2010.