IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Min-Min Scheduling Algorithm For Load Balancing In Cloud Computing

[1] Miss. Rajshree Devidas Patil , [2] Mr.Hirendra Hajare
[1] M.Tech. Scholar, Assistant Professor
[1]Computer Science & Engineering,
Ballarpur Institute of Technology,India

Abstract— Cloud computing is emerging as a new paradigm of large-scale distributed computing. In order to utilize the power of cloud computing completely, we need an efficient task scheduling algorithm. The traditional Min-Min algorithm is a simple, efficient algorithm that produces a better schedule that minimizes the total completion time of tasks than other algorithms in the literature [7]. However the biggest drawback of it is load imbalanced, which is one of the central issues for cloud providers. In this paper, an improved load balanced algorithm is introduced on the ground of Min-Min algorithm in order to reduce the makespan and increase the resource utilization (LBIMM). At the same time, Cloud providers offer computer resources to users on a pay-per-use base. In order to accommodate the demands of different users, they may offer different levels of quality for services. Then the cost per resource unit depends on the services selected by the user. In return, the user receives guarantees regarding the provided resources. To observe the promised guarantees, user-priority was considered in our proposed PA-LBIMM so that user's demand could be satisfied more completely. At last, the introduced algorithm is simulated using Matlab toolbox. The simulation results show that the improved algorithm can lead to significant performance gain and achieve over 20% improvement on both VIP user satisfaction and resource utilization ratio.

Keywords- Cloud Computing; User-priority Aware; Load Balance; Makespan; Min-Min Algorithm; Cloud Task Scheduling

I. INTRODUCTION

Currently Cloud computing has evolved as great potential technology that is known as a provider of dynamic services using very large scalable and virtualized resources over the Internet. Cloud is subject to User Requirement, Load Balance and other constraints that have direct effect on user consumption of resources controlled by cloud provider [11]. In order to utilize the power of cloud computing completely, we need an effective and efficient task scheduling algorithm. Task scheduling algorithm is responsible for dispatching tasks submitted by users to cloud provider onto heterogeneous available resources. This paper focuses on the efficient tasks scheduling considering the total completion time of tasks, resources utilization and user-priority in a cloud environment.

Cloud task scheduling is an NP-complete problem in general [7]. In the typical cloud scenario, cloud users submit their tasks to cloud scheduler. The Cloud scheduler firstly queries the Cloud Information Service for the availability of resources and to know their properties, and then scheduling the tasks on the resources that match tasks' requirements. After execution of tasks, the results are sent back to the users. How to schedule tasks in such cloud environment efficiently is a new challenge because of its nature of high heterogeneity in operating systems, architecture, resource providers and resource consumers [3]. The main purpose of a cloud task scheduling algorithm is to shorten overall the completion time of all tasks submitted by users, enhance the utilization of cloud resources and satisfy requirements of different users [2]. Unfortunately, it is difficult to find an optimal scheduling algorithm to meet those objectives at the same time. Most of the traditional scheduling approaches largely ignore user-priority issue, e.g. Min-Min, Max-min, they may not adapt to the cloud environment well as Cloud computing is not only a modelling technique but an economic model. Cloud providers offer computer resources to users on a pay-per-use base. In order to accommodate the demands of different users (e.g. VIP user, ordinary user), they may offer different level services. For instance, providers may offer a specific level service (e.g. VIP level service) and allow their users to select this level for each task individually to accommodate their needs. Then the price per resource unit arises for the users who select the VIP level service. In return, the VIP customers can enjoy better service than the other ordinary users with guarantee. To observe the promised guarantees, user-priority must be considered during tasks scheduling.

In this paper, two new frameworks of task scheduling algorithm is proposed to decrease job's completion time, improve the load balance and satisfy users' priority demands in the cloud. According to the result, the algorithms proposed in this paper outperform the Min-Min algorithm in terms of makespan, load balancing and user-priority aware. The remainder of the paper is organized as follows: Section II presents the overview of previous works about task scheduling with a strong emphasis on traditional Min-Min algorithm. In Section III and IV, the new Cloud task scheduling algorithm we proposed (LBIMM and PALBIMM) are introduced. The implementation and experiments are given in Section V. Finally, Section VI concludes the paper and presents future works.

II. RELATED WORKS

It is well know that the complexity of a general scheduling problem is NP-Complete. As mentioned in section I, the scheduling problem becomes more challenging because of the unique characteristics belonging to Cloud computing. Some of these characteristics are the following:

- **The high heterogeneity of resources**: Cloud systems act as large virtual supercomputers, yet the resources could be very disparate, ranging from laptops, desktops, supercomputers and even small devices of limited computational resources.
- **The high heterogeneity of tasks**: Tasks reaching to any Cloud system are diverse and heterogeneity in terms of their user demands.
- **User-priority**: this characteristic is an important issue in Cloud computing. User-priority must be considered during task scheduling with guarantee that users who pay more can enjoy better service.

In the literature, large numbers of task scheduling algorithm were proposed in the past. Braun et al [7] have studied the relative performance of eleven heuristic algorithms for task scheduling such as Opportunistic Load Balancing (OLB), Minimum Execution Time (MET), Minimum Completion Time (MCT), Min-Min, Max-Min, Duplex, Genetic Algorithm (GA), etc. They have also provided a simulation basis for researchers to test the algorithms. Their results show that the simple Min-Min algorithm produces a better schedule that minimizes the makespan than the other algorithms and performs next to GA which the rate of improvement is also very small in most of the scenarios.

A. Traditional Min-Min Scheduling Algorithm

The Min-Min algorithm is simple and still basis of present cloud scheduling algorithm [14]. It starts with a set S of all unmapped tasks. Then the resource R which has the minimum completion time for all tasks is found. Next, the task T with the minimum size is selected and assigned to the corresponding resource R (hence the name Min-Min). Last, the task T is removed from set S and the same procedure is repeated by Min-Min until all tasks are assigned (i.e., set S is empty).

The pseudo code of Min-Min algorithm is represented in Fig 1 assuming we have a set of n tasks (T_1, T_2, T_3 ... T_n) need to be scheduled onto m available resources (R_1, R_2, R_3 ... R_m). We denotes the Expected Completion Time for task i

($1 \le i \le n$) on resources j ($1 \le j \le m$) as Ct_{ij} that is calculated as in (1), Where rt_j represents the Ready Time of resource R_j and Et_{ij} represents the Execution Time of task T_i on resource R_j.

$$Ct_{ij} = Et_{ij} + rt_j \quad (1)$$

```
1.   For all submitted tasks in the set; Ti
2.   For all resources; Rj
3.   $Ct_{ij}=Et_{ij}+rt_j$; End For; End For;
4.   Do while tasks set is not empty
5.   Find task $T_k$ that cost minimum execution time.
6.   Assign $T_k$ to the resource $R_j$ which gives minimum expected complete time
7.   Remove $T_k$ from the tasks set
8.   Update ready time $rt_j$ for select $R_j$
9.   Update $C_{ij}$ for all $T_i$
10.  End Do
```

Figure 1. The Traditional Min-Min Scheduling Algorithm.

B. An Illustrative Example Of Min-Min Scheduling Algorithm

In order to illustrate the Min-Min algorithm, assume we have five tasks submitted by different users for scheduling on two available resources. Table I, represents the processing speed and service level of each resource while Table II, represents the task size and the user group of each task. Data given in Table I and Table II are used to calculate the expected completion time and execution time of the tasks on each of the resources.

TABLE I. RESOURCES SPECIFICATION

Resources	**Processing Speed (MB/Sec)**	**Service Level**
R1	10	VIP
R2	8	Ordinary
R3	5	Ordinary

TABLE II. TASKS SPECIFICATION

Tasks	**Task Size (MB)**	**User Group**
T1	10	Ordinary
T2	15	Ordinary
T3	20	Ordinary
T4	25	VIP
T5	50	Ordinary

Table III demonstrates calculated execution time of the tasks and expected complete time at the same time. On next step of the algorithm iteration, data in Table III will be updated until all tasks are allocated.

TABLE III. EXECUTION TIME (EXPECTED COMPLETE TIME) OF TASKS ON EACH OF THE RESOURCES : MIN-MIN SCHEDULING ALGORITHM

Task/Resource	**(VIP) R1**	**R2**	**R3**
T1	1 (1)	~~1.25(1.25)~~	~~2(2)~~
T2	~~1.5 (2.5)~~	1.875(1.875)	~~3(3)~~
T3	2 (3)	~~2.5(4.375)~~	~~4(4)~~
(VIP) T4	~~2.5 (5.5)~~	3.125(5)	~~5(5)~~
T5	**5(8)**	~~6.25(11.25)~~	~~10(10)~~

Figure 2 includes Gantt Charts representing the schedule result of using Min-Min algorithm on allocating tasks to resources. As shown in Fig 2, the Min-Min algorithm achieves total makespan 8 seconds but uses only two resources R1, R2. And without user-priority aware, the VIP task T4 is treated no different with other ordinary tasks.

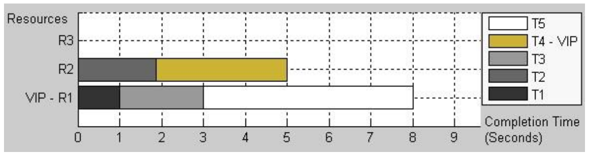

Figure 2. Gantt Chart Of Example Schedule: Min-Min Algorithm.

C. Challenges Of Min-Min Scheduling Algorithm In Cloud Computing

Based on the experimental result from the illustrative example from section 2.2, Min-Min algorithm fails to utilize the resources efficiently which lead to a load imbalance. And without use-priority aware during scheduling, the VIP users are not guaranteed with better services which lead to VIP users' dissatisfaction. Firstly, to avoid the drawback of load imbalance, LBIMM is proposed in this paper to optimize the load balance of Min-Min aims to increase the utilization of resources with light load or idle resources thereby freeing the resources with heavy load. Secondly, user-priority aware is introduced PA-LBIMM so as to observe the promised guarantees that VIP customers can enjoy better service than the other ordinary users.

III. LOAD BALANCE IMPROVED MIN-MIN SCHEDULING ALGORITHM (LBIMM)

In this paper, the Load Balance Improved Min-Min (LBIMM) scheduling algorithm is proposed that takes the characteristic of the Min-Min scheduling algorithm as foundation. The performance of Min-Min scheduling algorithm is considered to minimize the completion time of all works. However, the biggest weakness of Min-Min algorithm is it does not considers the work load of each resource. Therefore, some resources maybe always get busy but some nodes maybe still, as shown in Figure 2. The proposed LBIMM will improve the load unbalance of the Min-Min and reduce the execution time of each resource effectively.

The pseudo code of LBIMM algorithm is represented in Fig 3. It starts by executing Min-Min algorithm at the first step. At the second step it chooses the smallest size task from the most heavy load resource and calculates the completion time for that task on all other resources. Then the minimum completion time of that task is compared with the makespan produced by Min-Min. If it is less than makespan then the task is reassigned to the resource that produce it, and the ready time of both resources are updated. The process repeats until no other resources can produce less completion time for the smallest task on the heavy load resource than the makespan. Thus the heavy load resources are freed and the light load or idle resources are more utilized. This makes LBIMM to produce a schedule which improves load balancing and also reduces the overall completion time.

```
1.  For all submitted tasks in the set; $T_i$
2.  For all resources; Rj
3.  $Ct_{ij}=Et_{ij}+rt_j$; End For; End For;
4.  Do while tasks set is not empty
5.  Find task $T_k$ that cost minimum execution time.
6.  Assign $T_k$ to the resource $R_j$ which gives minimum expected complete time
7.  Remove $T_k$ from the tasks set
8.  Update ready time rj for select $R_j$
9.  Update $C_{ij}$ for all $T_i$
10. End Do
11. //Rescheduling to balance the load
12. Do while the most heavy load resource  is considered no need for rescheduling
13. Find task $T_i$ that cost minimum execution time on the heavy load resource $R_j$
14. Find the minimum completion time of $T_i$ produced by resource $R_k$
15. If such minimum completion time < makespan
16. Reassign Task $T_i$ to Resource $R_k$
17. Update the ready time of both $R_j$ and $R_k$
18. End If
19. End Do
20. //where Makespan represents maximum completion time of all tasks which equals to the
    completion time of the most heavy load resource
```

Figure 3. LBIMM Scheduling Algorithm.

A. An Illustrative Example Of LBIMM Scheduling Algorithm

We take the same example from section II, as shown on Table I and Table II. The first step of LBIMM algorithm is running Min-Min algorithm to produce the draft schedule, Figure 2, with makespan 8 second. At the second step, it find the heaviest load resource R1 in the draft schedule and then select the smallest size task T1 for considering being rescheduled. As we can calculate from the Table II, if task T1 is reassigned to resource R3, it produces a completion time of 2 seconds, which is less than the makespan 8 seconds. Thus, T1 is reassigned to R3 and both the ready time of R1 and R3 will be update. The makespan of the schedule will be update to 7 second as well. And the load balancing process repeats until no more tasks from the heaviest load resource need to be reassigned as demonstrates on Table IV. And then, the final schedule produced by LBIMM is represented in Figure 4.

From Figure 4 we can observe that LBIMM produce less makespan, 6 seconds, and more balanced load on all resources than the Min-Min algorithm. However, userpriority is still not considered in LBIMM algorithm. In order to meet the demands of different users, user-priority and LBIMM will be integrated in our next proposed PA-LBIMM scheduling algorithm.

TABLE IV. EXECUTION TIME (EXPECTED COMPLETE TIME) OF TASKS ON EACH OF THE RESOURCES : LBIMM SCHEDULING ALGORITHM

Task/Resource	**(VIP) R1**	**R2**	**R3**
T1	~~1 (1)~~	~~1.25(1.25)~~	2(2)
T2	~~1.5 (2.5)~~	1.875(1.875)	~~3(3)~~
T3	~~2 (2)~~	~~2.5(4.375)~~	**4(6)**
(VIP) T4	~~2.5 (5.5)~~	3.125(5)	~~5(5)~~
T5	5(5)	~~6.25(11.25)~~	~~10(10)~~

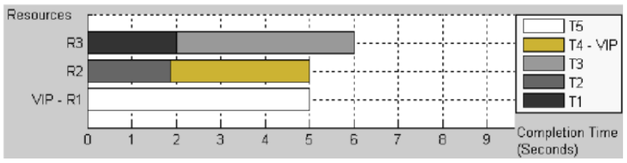

Figure 4. Gantt Chart Of Example Schedule: LBIMM Algorithm.

IV. USER-PRIORITY AWARED LOAD BALANCE IMPROVED MIN-MIN SCHEDULING ALGORITHM (PA-LBIMM)

Both the Min-Min and LBIMM algorithm do not consider the user-priority demand of tasks for scheduling. In our proposed PA-LBIMM algorithm, we consider the matching of the user-priority aware between the tasks and resources based on LBIMM. PA-LBIMM algorithm will first divide all the tasks into two groups G1 and G2. G1 is for the VIP users' tasks which demand higher priority requirement. G2 is for the ordinary users' tasks demanding lower priority requirement. Then, instead of scheduling all the tasks to the resources, we schedule the tasks in G1 with higher priority request first. For each task in G1, it runs the Min-Min algorithm to assign all the VIP tasks to the VIP qualified resources set. And then we schedule the tasks with lower priority request in G2 to assign them to all the resources by Min-Min algorithm. At the end, the load balancing function is processed to optimize the load of all resources to produce the final schedule. The modified algorithm is given in Figure 5.

A. An Illustrative Example Of PA-LBIMM Scheduling Algorithm

We are taking the same example from Section II, as shown on Table I and Table II. Firstly, all the tasks are divided into two groups, G1 and G2. G1 contains VIP task T4 and G2 contains ordinary task T1, T2, T3 and T5. Secondly, VIP Task T4 in G1 will be scheduled by Min-Min algorithm first and assigned to the resource that provides VIP service, VIP resource R1. Thirdly, Ordinary tasks, T1, T2, T3, and T5 in G2 will be scheduled by Min-Min algorithm and assigned to all the resources. At last, the load balance function is processed to optimize the schedule's resources load as demonstrated in Table V and Figure 6.

```
1.  Divide the tasks into two group according to the user-priority demand: VIP G1 and Ordinary G2
2.  For all submitted tasks in the high priority demand group; G1
3.  For all VIP resources; $R_j$
4.  $Ct_{ij}=Et_{ij}+rt_j$; End For; End For;
5.  Do while tasks set is not empty
6.  Find task $T_k$ that cost minimum execution time.
7.  Assign $T_k$ to the VIP resource $R_j$ which gives minimum expected complete time
8.  Remove $T_k$ from the tasks set
9.  Update ready time $rt_j$ for select $R_j$
10. Update $C_{ij}$ for all $T_i$ 11. End Do 12.
13. For all submitted tasks in the low priority demand group; G2
14. For all resources; $R_j$
15. $Ct_{ij}=Et_{ij}+rtj$; End For; End For;
16. Do while tasks set is not empty
17. Find task $T_k$ that cost minimum execution time.
18. Assign $T_k$ to the resource $R_j$ which gives minimum expected complete time
19. Remove $T_k$ from the tasks set
20. Update ready time $rt_j$ for select $R_j$
21. Update $C_{ij}$ for all $T_i$ 22. End Do 23.
24. //Rescheduling to balance the load
25. Do while the most heavy load resource is considered no need for rescheduling
26. Find task Ti that cost minimum execution time on the heavy load resource $R_j$
27. Find the minimum completion time of $T_i$ produced by resource $R_k$
28. If such minimum completion time < makespan
29. Reassign Task $T_i$ to Resource $R_k$
30. Update the ready time of both $R_j$ and $R_k$
31. End If
32. End Do
33. //where Makespan represents maximum completion time of all tasks which equals to the completion time of the most heavy load resource
```

Figure 5. PA-LBIMM Scheduling Algorithm.

TABLE V. THE EXECUTION TIME (EXPECTED COMPLETE TIME) OF TASKS ON EACH OF THE RESOURCES : PA-LBIMM SCHEDULING ALGORITHM

Task/Resource	(VIP) R1	R2	R3
(VIP) T4	~~2.5 (2.5)~~	3.125(3.125)	~~5(5)~~
T1	~~1 (3.5)~~	1.25(4.375)	~~2(2)~~
T2	~~1.5 (4)~~	~~1.875(3.125)~~	3(3)
T3	~~2 (4.5)~~	2.5(6.875)	~~4(7)~~
T5	5(5)	~~6.25(10)~~	~~10(13)~~

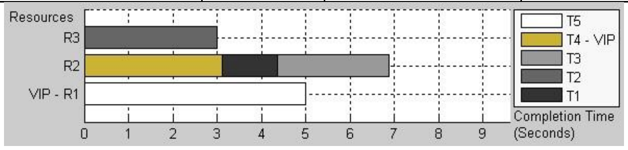

Figure 6. Gantt Chart Of Example Schedule: PA-LBIMM Algorithm.

According to the Table V, the makespan produced by PA-LBIMM is 6.875 seconds which is less than the makespan produced by Min-Min. The load balance on all resources out performs the Min-Min algorithm as well. And one important improvement is that the completion time of VIP task 1 reduces to 3.125 seconds. Thus the user-priority demand is more guaranteed by PA-LBIMM algorithm.

V. EXPERIMENTAL RESULTS

To simulate and evaluate the efficiency of the proposed algorithms, problems having resources heterogeneity and tasks heterogeneity are collected from various literature and executed for Min-Min, proposed LBIMM and PA-LBIMM algorithm. A program is designed in Matlab and the experiments are carried out. The program is composed of Resources-Generator and Task-Generator. ResourcesGenerator is responsible for the simulated cloud environment and generates heterogeneous cloud resources in the specified range of processing speed and service level. Task-Generator generates random tasks in the specified range of task size and user group. The experiment is carried out to compare the performance of the algorithm on the Makespan, Average Resource Utilization Ratio, Average VIP Tasks' Completion Time, and Average Ordinary Tasks' Completion Time.

- **Makespan:** Makespan is a measure of the throughput of the heterogeneous cloud system. It can be calculated as the following relation:

$$\text{Makespan}=\max(rt_j) \quad (2)$$

Where rt_j denotes the ready time of each resource after scheduled. The less the makespan of a scheduling algorithm the better it works.

- **Average Resource Utilization Ratio (ARUR):**
ARUR is calculated through the following relation:

$$\text{ARUR}= \text{mean}(rt_j)/\text{Makespan}*100\% \quad (3)$$

Where ARUR is in the range 0 to 1. The best and most efficient load balancing level is achieved if ARUR equals 1. So, scheduling algorithm will have better performance if ARUR is close to 1.

- **Average VIP Task Completion Time (AVIPCT)** is calculated by

$$\text{AVIPCT}= \text{mean}(CT_{j\text{-VIP}}) \quad (4)$$

The less the AVIPCT of a scheduling algorithm produced, the better it works.

- **Average Ordinary Task Completion Time**

(AORDCT) is calculated by

$$\text{AORDCT} = \text{mean}\ (CT_{j\text{-ORD}}) \qquad (4)$$

The less the AORDCT of a scheduling algorithm produced, the better it works. In order to meet userpriority demand, it will result in the tradeoff between
AVIPCT and AORDCT.

The experimental testing of our algorithms is performed in four scenarios:

a) ***Scenario A***: Low proportion of VIP tasks.
b) ***Scenario B****:* High proportion of VIP tasks.
c) ***Scenario C****:* Different numbers of Random tasks.

In **scenario A** and **B**, number of resources is chosen to be 5 and number of tasks is chosen to be 10 with different proportion of VIP tasks. In **scenario C**, number of random resources is chosen to be 50. Five different numbers of random tasks have been chosen: 200, 400, 600, 800 and 1000, to be sure of efficiency of the proposed algorithms working under the heterogeneity cloud environment.

A. Scenario A: Low Proportion of VIP Tasks

TABLE VI. SCENARIO A RESOURCES SPECIFICATION

Resources	Processing Speed (MB/Sec)	Service Level
R1	10	VIP
R2	8	Ordinary
R3	9	Ordinary
R4	5	Ordinary
R5	3	Ordinary

TABLE VII. SCENARIO A TASKS SPECIFICATION

Tasks	Task Size (MB)	User Group
T1	95	VIP
T2	24	VIP
T3	61	Ordinary
T4	49	Ordinary
T5	89	Ordinary
T6	76	Ordinary
T7	46	Ordinary
T8	3	Ordinary
T9	82	Ordinary
T10	45	Ordinary

The resources and tasks specification for simulating Scenario A are listed in Table VI and Table VII. The Gantt chart of different schedules produced by Min-Min, LBIMM and PA-LBIMM is shown on Figure 7. The detail performance results are shown on Table VII.

and PA-LBIMM is shown on Figure 7. The detail performance results are shown on Table VII.

TABLE IX. SCENARIO B RESOURCES SPECIFICATION

TABLE X. SCENARIO B TASKS SPECIFICATION

Tasks	Task Size (MB)	User Group
T1	16	VIP
T2	17	VIP
T3	32	VIP
T4	4	VIP
T5	36	VIP
T6	4	VIP
T7	80	VIP
T8	100	VIP
T9	12	Ordinary
T10	63	Ordinary

Resources	Processing Speed (MB/Sec)	Service Level
R1	10	VIP
R2	8	Ordinary
R3	4	Ordinary
R4	7	Ordinary
R5	5	Ordinary

Figure 7. Gantt Chart Of Different Schedules for Scenario A : Min-Min, LBIMM, PA-LBIMM Algorithm.

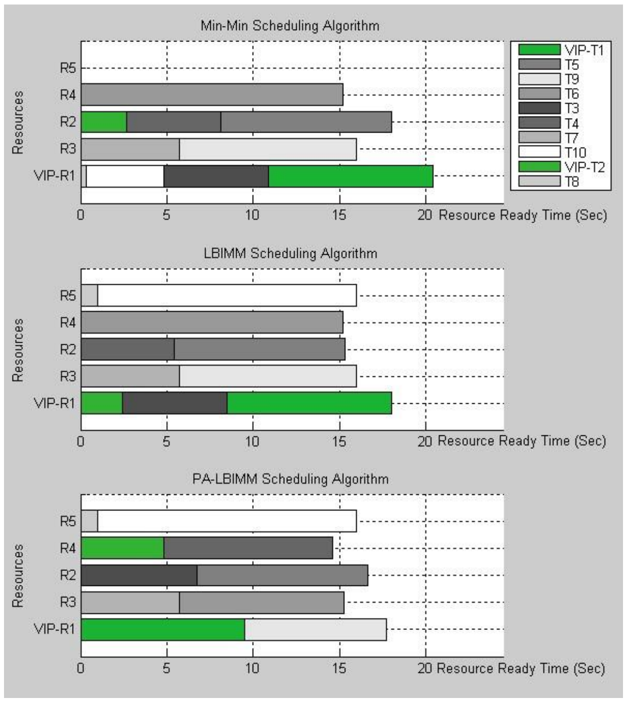

TABLE VIII. PERFORMANCE RESULTS : SCENARIO A

Scheduling Algorithm	Min-Min	LBIMM	PA-LBIMM
Number of Tasks	10	10	10
Proportion of VIP Tasks	20%	20%	20%
Number of Resources	5	5	5
Proportion of VIP Resources	20%	20%	20%
Makespan (Sec)	20.4	18	17.7
Average Resource Utilization Ratio	68.24%	89.48%	90.64%
Average VIP Task Completion Time (Sec)	11.53	10.2	7.15
Average Ordinary Task Completion Time (Sec)	9.88	10.4	11.71

As can be seen from Figure 7 and Table VIII, compare with Min-Min, both LBIMM and PA-LBIMM decreases the total completion time, the make span, of tasks and increases the average resource utilization ratio by over 20%. And PALBIMM does a great job to shorten the average completion time of VIP tasks, which is reduced by 4.38 sec and 3.05 sec compared with Min-Min and LBIMM respectively. But it result in the tradeoff between VIP tasks and Ordinary tasks, the average ordinary task completion time is increased by 1.83 sec and 1.31 sec compared with Min-Min and LBIMM respectively. However, this tradeoff is acceptable if we consider user-priority between VIP users and Ordinary users.

B. Scenario B:High Proportion of VIP Tasks

The resources and tasks specification for simulating Scenario B are listed in Table IX and Table X. The Gantt chart of different schedules produced by Min-Min, LBIMM

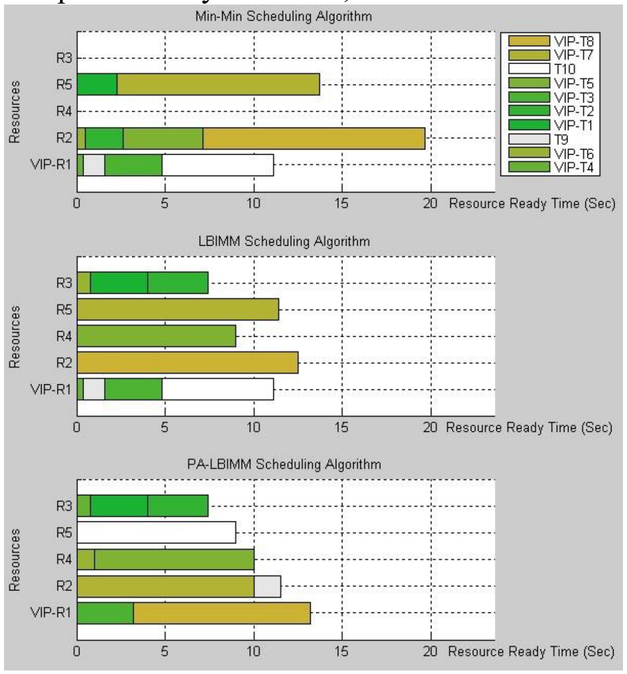

Figure 8. Gantt Chart Of Different Schedules for Scenario B : Min-Min, LBIMM, PA-LBIMM Algorithm.

As we can see from Figure 8 and Table XI, the load balance level of Min-Min is very poor, both resource R3 and R4 are not utilized in scheduling. Both LBIMM and PALBIMM gain a great improvement on Make span (Decreased over 6 sec) and average resource utilization ratio (Increased over 30%) compare with Min-Min. But in this scenario with PA-LBIMM algorithm, the average VIP task completion time is similar with Min-Min and LBIMM while the average Ordinary task completion time is increased about 4 sec. In this case, we may need to introduce more VIP resources to do with the high proportion of VIP tasks situation. However, with the great improvement with make span and resource utilization, PA-LBIMM is still a better scheduling strategy than Min-Min. And LBIMM acts as the best scheduling strategy among the three in Scenario B.

TABLE XI. PERFORMANCE RESULTS : SCENARIO B

Scheduling Algorithm	**Min-Min**	**LBIMM**	**PA-LBIMM**
Number of Tasks	10	10	10
Proportion of VIP Tasks	80%	80%	80%
Number of Resources	5	5	5
Proportion of VIP Resources	20%	20%	20%
Makespan (Sec)	19.625	12.5	13.2
Average Resource Utilization Ratio	45.29%	82.29%	77.42%
Average VIP Task Completion Time (Sec)	6.38	6.29	6.2

Average Ordinary Task Completion Time (Sec)	6.35	6.35	10.25

C. Scenario C: Different Numbers Of Random Tasks on Random Resources

TABLE XII. RANDOM TASKS AND RESOURCES SPECIFICATION

Number of Tasks	200	400	600	800	1000
Proportion of VIP Tasks	40%	50%	40%	35%	15%
Number of Resources	50	50	50	50	50
Proportion of VIP Resources	45%	15%	35%	40%	20%

In this scenario, five different numbers of random tasks have been chosen: 200, 400, 600, 800 and 1000, to be sure of efficiency of the proposed algorithms working under the heterogeneity cloud environment. The detail of five random generated tasks and resources specification is listed in Table XII. The Make span, Average Resources Utilization Ratio, Average VIP Task Completion Time, Average Ordinary Task Completion Time and Overall results of Min-Min, LBIMM and PA-LBIMM are shown as follows:

- **Make span Results:** Figure 9 shows the results for make span of task number 200,400,600,800 and 1000, respectively. Overall LBIMM and PALBIMM give better make span than Min-Min. In most cases PA-LBIMM produces similar make span as LBIMM. But in the case when task number is 400, PA-LBIMM produces less improvement on make span than LBIMM.
- **Average Resources Utilization Ratio Results:** Figure 10 shows the results for average resources utilization ratio of task number 200,400,600,800 and 1000, respectively. Overall LBIMM and PA-LBIMM give much better resource utilization results in each task case than Min-Min. In most cases PALBIMM works similar as LBIMM except in the case when task number is 400.
- **Average VIP Task Completion Time Results:** Figure 11 shows the results for Average VIP Task Completion Time of task number 200,400,600,800 and 1000, respectively. As seen on the Figure, PALBIMM is producing a remarkable improvement on shortening the average completion time of VIP tasks where the user-priority demand of task cannot be compromised by either Min-Min or LBIMM.
- **Average Ordinary Task Completion Time Results:** Figure 12 shows the results for Average Ordinary Task Completion Time of task number 200,400,600,800 and 1000, respectively. In most case the average completion time tradeoff between VIP task and Ordinary task is acceptable except in the case when task number is 400.
- **Overall Results:** Under all possible situation, LBIMM and PA-LBIMM outperform Min-Min for comparison of both make span and resources utilization. In most case when demanded user priority of the task need to be fulfilled, PA-LBIMM outperforms both Min-Min and LBIMM for decreasing the average completion time of VIP tasks with acceptable tradeoff the average completion time of ordinary tasks. Except in the case of the proportion of VIP tasks is far larger than the proportion of VIP resources.

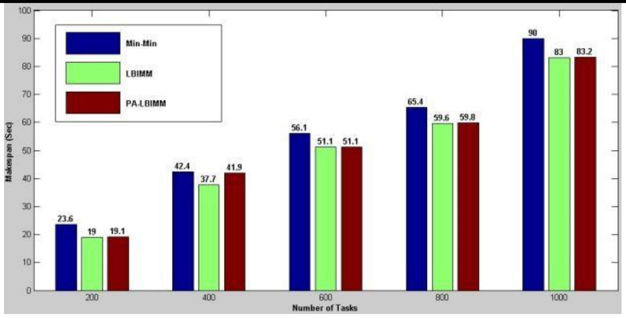

Figure 9. Gantt Chart Of Scenario C: Makespan Results.

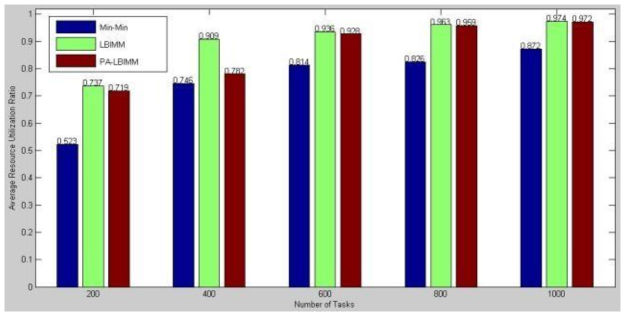

Figure 10. Gantt Chart Of Scenario C: Average Resources Utilization Ratio Results.

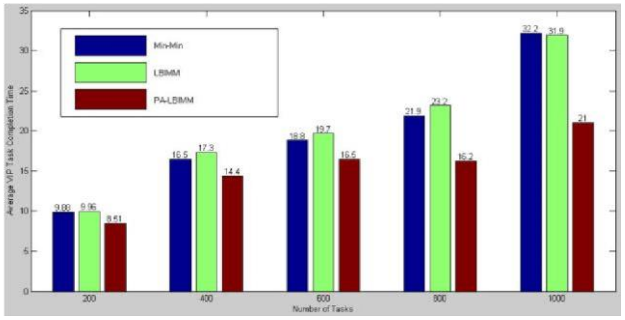

Figure 11. Gantt Chart Of Scenario C: Average VIP Task Completion Time Results.

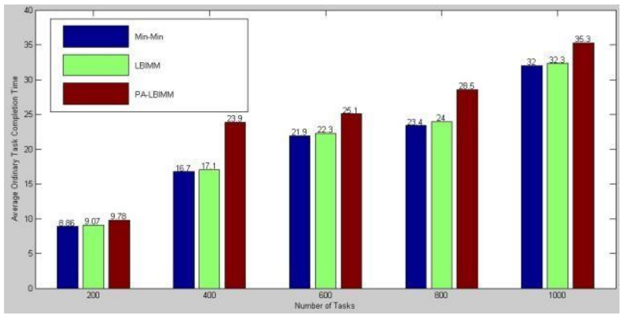

Figure 12. Gantt Chart Of Scenario C: Average Ordinary Task Completion Time Results.

VI. CONCLUSION AND FUTURE WORK

To achieve high computing throughput in a cloud environment, two new scheduling algorithms, LBIMM and PA-LBIMM, were proposed in this paper. Evaluation of our new algorithms was done through a simulation program under Matlab environment. The experimental results show that under all possible situations both the LBIMM and PALBIMM are capable of decreasing completion time of tasks, improving load balance of resources (Overall 20% improved on average resources utilization ratio in most case) and gain better overall performance than Min-Min algorithm. And in the case that demanded user-priority of tasks need to be fulfilled, which is an important issue in Cloud environment, PA-LBIMM out-performs both Min-Min and LBIMM for decreasing over 20% of the average completion time of VIP tasks.

This paper is only concerned with the make span, load balancing and user-priority for task scheduling based on Min-Min algorithm in Cloud environment. Many similar scheduling algorithms, e.g. Round Robin, Maximin, Genetic Algorithm (GA), can be devised. Many issues remain open. We did not consider deadline of each task, the high heterogeneity of interconnection networks, the geography location of tasks and resources, other QOS requirement and many other cases that can be topics of future research. Tasks

REFERENCES

[1] U Schwiegelshohn, A Tchernykh. “Online Scheduling for Cloud Computing and Different Service Levels”, Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, 1067 – 1074, 2012.

[2] RS Chang, CY Lin, and CF Lin. “An Adaptive Scoring Job Scheduling algorithm for grid computing”, Information Sciences – Elsevier, Volume 207, Pages 79–89, 2012

[3] T Kokilavani, GA DI. “Load Balanced Min-Min Algorithm for Static
Meta-Task Scheduling in Grid Computing”, International Journal of Computer Applications, Number 2 - Article 7, 2011

[4] J Liu, G Li. “An Improved MIN-MIN Grid Tasks Scheduling
Algorithm Based on QoS Constraints”, Optics Photonics and Energy Engineering (OPEE), 2010 International Conference on, Volume 1, Pages 281 - 283, 2010

[5] F Dong, SG Akl. “Scheduling Algorithms for Grid Computing:State of the Art and Open Problems”, Technical Report No. 2006-504, School of Computing, Queen’s University, Kingston, Ontario, Canada, 2006

[6] DDH Miriam, KS Easwarakumar. “A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems”, International Journal of Computer Science Issues, Volume 7, Issue 4, Pages 8-18, 2010

[7] Braun, Tracy D, et al. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems" Journal of Parallel and Distributed computing , Volume 61, Issue 6, Pages 810 – 837, 2001

[8] He, XiaoShan, XianHe Sun, and Gregor Von Laszewski. "QoS guided min-min heuristic for grid task scheduling", Journal of Computer Science and Technology, Volume 18, Number 4, Pages 442 – 451, 2003

[9] Wang, Shu-Ching, et al. "Towards a load balancing in a three-level cloud computing network", Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. Volume 1, Pages 108 - 113, 2010

[10] Kong, Xiangzhen, et al. "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction", Journal of Network and Computer Applications, Volume 34, Issue 4, Pages 1068 – 1077, 2011

[11] Elzeki, O. M., M. Z. Reshad, and M. A. Elsoud. "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications, Volume 50, Issue 12, Pages 22-27, 2012

[12] Chauhan, Sameer Singh, and R. C. Joshi. "QoS guided heuristic algorithms for grid task scheduling", International Journal of Computer Applications IJCA, Number 9, Article 4, Pages 24 -31, 2010

[13] Dong, Fang, et al. "A grid task scheduling algorithm based on QoS priority grouping", Grid and Cooperative Computing, GCC 2006 Fifth International Conference, Pages 58 - 61, IEEE, 2006.

[14] Yu, Xiaogao, and Xiaopeng Yu. "A new grid computation-based Min-Min algorithm", Fuzzy Systems and Knowledge Discovery, FSKD'09 Sixth International Conference on, Volume 1, Pages 43 – 45, IEEE, 2009.

[15] Hirales-Carbajal, Adán, et al. "Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid", Journal of Grid
Computing , Volume 10, Number 2, Pages 325 – 346, 2012