International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Resource-Aware Min-Min (RAMM) Algorithm for Resource Allocation in Cloud Computing Environment

Syed Arshad Ali, Samiya Khan, Mansaf Alam

Abstract*: Resource allocation (RA) is a significant aspect of Cloud Computing. The Cloud resource manager is responsible to assign available resources to the tasks for execution in an effective way that improves system performance, reduce response time, lessen makespan and utilize resources efficiently. To fulfil these objectives, an effective Tasks Scheduling algorithm is required. The standard Max-Min and Min-Min Task Scheduling algorithms are not able to produce better makespan and effective resource utilization. In this paper, a Resource-Aware Min-Min (RAMM) Algorithm is proposed based on basic Min-Min algorithm. The proposed RAMM Algorithm selects shortest execution time task and assigns it to the resource which takes shortest completion time. If minimum completion time resource is busy, then the RAMM Algorithm selects next minimum completion time resource to reduce waiting time of the task and improve resource utilization. The experiment results show that the proposed RAMM Algorithm produces better makespan and load balance than Max-Min, Min-Min and improved Max-Min Algorithms.

***Keywords* : Resource Allocation, Makespan, Task Scheduling, Min-Min, Max-Min.**

I. INTRODUCTION

Cloud Computing provides Internet-based dynamic computing services using large-scale virtualized resources. Distributed and parallel computing gives life to Cloud Computing [1]. It serves distributed computing resources to globally located users and delivers resource scalability, economic use of resources and on-demand services [2]. Resource allocation in Cloud Computing is very multifaceted because of dynamic nature of the Cloud environment [3]. The user's demand changes dynamically and availability of resources are also changing very frequently. From the Cloud provider's perspective, a huge amount of resources is needed to allocate among the globally distributed Cloud users dynamically in a cost-effective way while from the Cloud user's perspective a reliable and economic computing services are needed on demand [4]. There must be a Service Level Agreement (SLA) between the Cloud provider and the Cloud user which consider multiple parameters like the cost of service, the completion time of service (makespan), and throughput in addition to several others [5]. Resource allocation in Cloud Computing, is the process of allocating virtual machines, (storage, computing, and networking) to the Cloud user's applications. Cloud resource allocation comprises of both Cloud resource provisioning and scheduling. Cloud Computing mainly relies on virtualization, which enables a physical device to be virtually distributed into one or more virtual machines (VMs) [6]. Virtual machines are used for computation of user applications. Due to virtualization, unutilized resources of physical machines can be further used by another virtual machine to speed up the task's execution and resource utilization [7]. Resource allocation strategy should overcome the problems of over and under provisioning of resources, scarcity of resources, contention, and fragmentation of resources [8]. Scheduling is an important aspect of any computing system. In general, CPU scheduling deals with the execution of user-submitted jobs. First, all the user submitted jobs wait in ready queue for their turn of execution. The time spent in the ready queue by the job is known as waiting time. CPU scheduler selects jobs from the ready queue based on some criteria fulfilled by the job and assigns the same to the CPU for execution [9].

Manuscript published on 30 September 2019
* Correspondence Author
Syed Arshad Ali*, Department of Computer Science, Jamia Millia Islamia, New Delhi, India. Email: arshad158931@st.jmi.ac.in
Samiya Khan, Department of Computer Science, Jamia Millia Islamia, New Delhi, India. Email: samiyashaukat@yahoo.com
Mansaf Alam*, Department of Computer Science, Jamia Millia Islamia, New Delhi, India. Email: malam2@jmi.ac.in

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

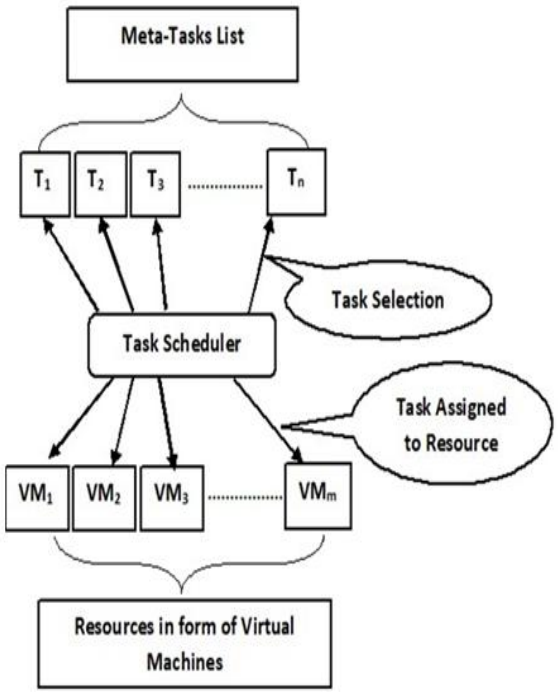

Fig. 1. Tasks-Resource Mapping by Cloud-Scheduler

The waiting time depends on several factors including resource availability, the priority of the job and the load on the system. Total time for execution of all jobs is known as makespan. The scheduling process should minimize the makespan to improve the system performance. Cloud user submits a task to Cloud scheduler which is responsible to select the available Virtual Machine and allocate it to the user's submitted task to fulfill Cloud user and Cloud provider's requirements effectively. Fig. 1 illustrates how Cloud scheduler schedule user's task to available resources. Various Task Scheduling algorithms consider task completion time and task execution time as scheduling criteria for resource allocation to user's tasks. The Max-Min [10], Min-Min [11], RASA [12] and improved Max-Min [13] algorithms also use these scheduling criteria for resource allocation. In Min-Min algorithm [11] the smallest completion time task schedule first to the fastest execution time resource. The major drawbacks of the Min-Min algorithm are imbalanced load and starvation of tasks with large service time. To solve these problems, another algorithm named Max-Min [10] is proposed, which schedules the largest completion time task to the smallest execution time resource. When the numbers of small tasks are more than large tasks, then Max-Min seems a better choice for Task Scheduling. However, in some cases, if large tasks are more than small tasks then total completion time and throughput of the system decreases. One more resource aware scheduling algorithm (RASA) [12] has been proposed for task scheduling which combines the features of both Max-Min and Min-Min algorithms. In this algorithm, the author also used completion time for each task and applied Max-Min and Min-Min algorithms one after another according to the number of resources available, If the number of resources is even then it applies Max-Min else it applies Min-Min algorithm. Another improved Max-Min algorithm [13] has been proposed, it schedules largely running time job first to the smallest completion time machine compared to basic Max-Min which assign large completion time job first to the smallest running time machine.

In this paper, the authors propose a better Task Scheduling algorithm, Resource-Aware Min-Min (RAMM) algorithm based on Min-Min algorithm [11], that schedules minimum execution time task to a resource with minimum completion time as against the method used by basic Min-Min algorithm, which selects the smallest completion time task to the smallest execution time resource. The basic Min-Min algorithm suffers from the load imbalance problem because it always selects the virtual machine (VM) which has smallest burst time for the job, if that resource (VM) is not ready then the job must wait for it while rest of resources are idle in this situation, which causes resource's load imbalance and increased makespan in Cloud system. On the contrary, in this proposed RAMM algorithm, if the smallest completion time resource is not free, then it will select next minimum completion time resource for that task, which allows load balancing among resources and decrease makespan, as no task will wait for the busy resource if next minimum completion time resource is available.

This paper makes the following contributions --

1) Various Task Scheduling algorithms have been studied and improvements and limitations of these algorithms have been assessed.
2) Based on this study a new Task Scheduling algorithm named Resource-Aware Min-Min algorithm has been proposed, which is based on a Min-Min algorithm.
3) The experiments are done on the data given in paper [13] in MATLAB. The experimental results show that the proposed algorithm gives better resource utilization and minimize makespan.
4) The proposed algorithm has been compared with the Basic Max-Min, Min-Min, and Improved Max-Min algorithms. The results show that the proposed algorithm outperforms its predecessors.

Rest of the paper comprises of following sections. In Section-II related works are discussed and, Section-III describes the proposed algorithm. Implementation of the algorithm has been provided in section-IV while experiment on the proposed algorithm is explained in Section-V. Section-VI illustrates the comparison and discusses the results. Section-VII gives insights on the conclusion and future work.

II. RELATED WORKS

Resource allocation is an NP-hard problem in Cloud computing. Task scheduling is one of the important aspects of resource allocation, which refers to the allocation of tasks to the available resources in an efficient way that fulfills task requirement and utilizes resources, efficiently. Many researchers are working in this field and have proposed many Task Scheduling algorithms, which can fulfill the need for Cloud environment.

In [14], the author uses Min-Min algorithm to propose a user's priority-based Min-Min scheduling algorithm. The focus of this algorithm is on user's priority to fulfill the Service Level Agreement (SLA).

It prioritized user task to overcome the unbalanced workload problem of the basic Min-Min algorithm.

In Max-Min algorithm [10], small jobs starved due to the priority given to large jobs. To overcome this problem, in [15], the author proposed an algorithm named as Max-Min spare time (MMST), which reduces waiting time of small jobs and utilizes resource efficiently. The algorithm also reduces service cost of Cloud resources.

In [16], the author used the improved Max-Min algorithm as a base to propose an enhanced Max-Min algorithm with some changes to minimize the makespan and load imbalanced problem. Instead of selecting the largest execution time task to resource produces minimum completion time, it assigns average running time user's application to the smallest completion time resource.

A new enhanced Load Balancing algorithm is developed in [17], which is based on load balanced Min-Min algorithm. Load balanced Min-Min algorithm works in two steps- In the first step, it applies Min-Min algorithm while in the second step, it reschedules tasks to unutilized resources to improve makespan as well as resource load balancing.

However, sometimes it does not give appropriate results because it schedules task with minimum completion time. On the other hand, enhanced load balanced Min-Min algorithm also works in two steps- the Min-Min algorithm is applied in the first step, and in the second step, it reschedules the largest completion time task to suitable resource for better resource utilization and to improve makespan.

Cloud Computing is more popular due to its elastic property. A user can expand or reduce his infrastructure resources according to his requirement using Cloud Computing. In [18], an improved Max-Min algorithm for elastic Cloud is proposed which balances the workload among the resources by maintaining two tables namely task table and virtual machine status table. The task-status table maintains the predictable end time for each task and virtual machine status table maintains the estimated load of each virtual machine. It works in two steps- In the first step, it executes VM task estimation algorithm while in the second step, it executes task allocation algorithm. It improves the resource utilization and response time of tasks.

In [13], an improved Max-Min algorithm is proposed based on a basic Max-Min algorithm. The Max-Min algorithm gives priority to large task for execution by assigning them to fastest resource and small tasks are executed concurrently by other resources. If small tasks are more than large tasks, then concurrent execution is not possible, which increases makespan. To overcome this problem, an improved Max-Min algorithm is proposed, which schedules maximum execution time task to minimum completion time resource. In [19], the author provides analysis of resource usage and attempts to give an insight about benevolent of production trace related to the ones in a cloud environment. Some of the researchers are also work and find open challenges in the field of scientific workflow scheduling in Cloud Computing [20]. Table-I provides an analysis of pros and cons of these Task Scheduling Algorithms.

Table-I: Various Task Scheduling Algorithms

Task Scheduling Algorithm	Method or Strategy	Improvements	Limitations
User-Priority Guided Min-Min [14]	Two algorithms are proposed for user priority Algorithm1 LBIMM Algorithm2 PA-LBIMM	User-priority consideration, reduce Makespan and improve load balance	Deadline of each task, Heterogeneity of network and QoS requirements are not considered
Max-Min Spare time [15]	MMST algorithm is proposed based on basic Max-Min	Load balancing, cost-effective and resource utilization	Scalability of the system is not considered
Enhanced Max-Min [16]	Use Average completion time of task for scheduling	Improve Makespan and resource utilization	Improper Load-balancing
Enhanced load-balanced Min-Min [17]	The two-phase algorithm developed based on basic Min-Min	Reduce Makespan and efficient resource utilization	Time complexity is O(n)
Improved Max-Min for Elastic Cloud [18]	Task-Status Table is maintained to estimate the actual load of the virtual machine	Resource utilization and reduce response time	Not cost-effective
Improved Max-Min [13]	Based on basic Max-Min, the expected execution time of Task is considered for scheduling	Supports Concurrent execution of tasks and reduced Makespan	Scalability and QoS requirements are not considered

III. RESOURCE-AWARE MIN-MIN (RAMM) ALGORITHM

The algorithm proposed in this paper is an advanced version of traditional Min-Min Task Scheduling algorithm [11] for the grid system. Smallest completion time task is scheduled first to the fastest execution time resource in the basic Min-Min algorithm. If fastest (minimum execution time) resource is busy in executing another task then that minimum completion time task will have to wait until that resource will ready for it, which means that the waiting time of the task will increase. This in turn, will also increase makespan and decrease the system performance. The proposed Resource-Aware Min-Min (RAMM) algorithm works differently as it schedules the minimum execution time task to the minimum completion time resource.

If the minimum completion time resource is busy in executing of another task, then it assigns the next minimum completion time resource to that task, which reduces the waiting time of the task and decreases makespan. Another problem with the basic Min-Min algorithm is that if all tasks have minimum execution time on a single resource, then all tasks will assign
to that resource, which increases load on a resource while rest of the resources are idle at that time. It is the drawback of this algorithm that, causes load imbalance problem and minimize effective resource utilization. The proposed algorithm overcomes the basic Min-Min load imbalance problem by effectively using idle or available resources. Several Scheduling criteria can be considered for performance enhancement in Cloud Computing systems [21]. Cloud's system performance can be measured in resource utilization, throughput, load-balancing, system usage, turnaround time, response time and waiting time.
In addition, some other criteria could also be considered for characterizing the Cloud system's performance such as user priority, quality of service (QoS) and resource failure apart from many others. One of the most prevalent and extensively considered Scheduling criteria in Cloud resource allocation is the minimization of makespan and minimization of energy consumption [22] [23]. Makespan is the time taken by the system to complete all submitted tasks. Smaller makespan value shows that the schedule is providing efficient scheduling of the tasks to the system resources. In our proposed algorithm, the author considers the following aspects of the tasks for scheduling.

Expected execution time

Execution time (ET_{ij}) is a unit of time taken by the resource R_j to complete task T_i when resource R_j has no previous task for execution. It is also known as burst time of task T_i.

Expected completion time

Completion time (CT_{ij}) is an amount of time, resource R_j is taking to complete a task T_i. If resource R_j is busy in executing of a previous submitted task, then the ready time of resource R_j will be added to the execution time of task T_i to get the completion time of task T_i on resource R_j.

IV. IMPLEMENTATION OF RAMM ALGORITHM

Let m be the number of resources which must process n number of tasks. In the proposed algorithm we have considered T as the set of all tasks, $T = \{T_i \mid i = 1, 2, 3, 4, \ldots\ldots, n\}$ and R as the set of all resources, $R = \{R_j \mid j = 1, 2, 3, 4, \ldots\ldots, m\}$. Resources will be mapped to tasks. The Cloud scheduler has the expected execution time of each task on each resource, which is given in form of ET matrix in which et_{ij}, $1 \le i \le n$, $1 \le j \le m$ is the expected execution time of task i on resource j, Eq. (1). The algorithm calculates the expected completion time and stored in a matrix named CT in which ct_{ij}, $1 \le i \le n$, $1 \le j \le m$ is the expected completion time of each task i on each resource j, Eq. (2).

$$ET = \begin{array}{c} \\ T_1 \\ T_2 \\ . \\ . \\ T_n \end{array} \begin{array}{c} \begin{array}{cccc} R_1 & R_2 & . & . & R_m \end{array} \\ \begin{pmatrix} et_{11} & et_{12} & . & . & et_{1m} \\ et_{21} & et_{22} & . & . & et_{2m} \\ . & . & . & . & . \\ . & . & . & . & . \\ et_{n1} & . & . & . & et_{nm} \end{pmatrix} \end{array} \quad (1)$$

$$CT = \begin{array}{c} \\ T_1 \\ T_2 \\ . \\ . \\ T_n \end{array} \begin{array}{c} \begin{array}{cccc} R_1 & R_2 & . & . & R_m \end{array} \\ \begin{pmatrix} ct_{11} & ct_{12} & . & . & ct_{1m} \\ ct_{21} & ct_{22} & . & . & ct_{2m} \\ . & . & . & . & . \\ . & . & . & . & . \\ ct_{n1} & . & . & . & ct_{nm} \end{pmatrix} \end{array} \quad (2)$$

Each resource may have some previously assigned task for execution, because of which it takes some time to ready this resource for coming task T_i. This amount of time is known as the ready time of resource R_j or waiting time of task T_i which is denoted as $RT = \{RT_j \mid j = 1, 2, 3, 4, \ldots\ldots, m\}$. Therefore, each task's expected execution time will be added to the waiting time of each resource to get the expected completion time on each resource. Initially, all the resources are free, which is why the initial expected task completion time is same as expected task execution time. If small tasks are more than large tasks in the Min-Min algorithm, then the fast computing resources will always be busy and slow computing resources are idle most of the time. As a result, starvation for large tasks is caused, leading to the problem of load imbalance because it does not support concurrent execution of the tasks.

Therefore, the traditional Task Scheduling algorithms are not well suited in Cloud environment due to its dynamic nature. To overcome these problems of starvation and unbalance load, the proposed approach uses an alternate method in which no task will have to wait if there is any available resource, which supports simultaneous execution of the tasks. When task will complete its execution, it will be removed from the meta-task sets. Moreover, the projected end time of remaining task on each resource will be updated for further scheduling and this process will repeat until all tasks complete their execution. Table-II shows the variable's definition. The pseudo code of the proposed RAMM algorithm is represented in Table-III.

Table-II: Variables and Notations

Variable	Definition
T_i	Task-i
R_j	Resource-j
CT_{ij}	Completion time of Task-T_i on Resource-R_j
ET_{ij}	The execution time of Task-T_i on Resource-R_j
RT_j	Ready time of Resource-R_j
MI_i	Machine instruction of Task-T_i
MB_i	Data Volume of Task-T_i in Mega-Byte
$MIPS_j$	Processing Speed of Resource-R_j
$MBPS_j$	Bandwidth of Resource-R_j

Table-III: Resource-Aware Min-Min (RAMM) algorithm

Phase	
	Input: Meta-Tasks List and Set of Resources. **Output:** Mapping of Tasks to the System Resources for execution.

Initial Calculation of Completion time of each task	Begin 1. For every tasks T_i in Meta-Tasks List 2. For every resources R_j 3. Calculate $CT_{ij} = ET_{ij} + RT_j$ 4. end 5. end
Scheduling Iteration	6. do until all tasks in Meta-Tasks List are mapped
Scheduling Criteria	7. find task T_k that has Smallest Execution time and resource R_l that gives Minimum Completion time 8. if Resource R_l is busy then 9. find the next resource R_l with the Next Minimum Completion time 10. goto step-8
Task Mapping	11. else 12. Execute task T_k on the resource R_l 13. end if
Complete Task Execution	14. delete task T_k from Meta-Tasks List
Update Ready time and Completion time of tasks	15. update RT_l 16. update CT_{il} for all i
Scheduling Iteration	17. end do End

V. EXPERIMENT ON PROPOSED ALGORITHM

The experimental work for the proposed algorithm is performed using MATLAB [24]. The Intel Core i5 system with 12 GB of RAM is used to create experimental setup. The experimental environment and dataset are same as that used by [13], because this proposed work is compared with the improved Max-Min algorithm [13]. To get a practical experiment on our proposed algorithm, we assume there are four tasks {T = T1, T2, T3 and T4} and two resources {R = R1 and R2} in the system. All tables data is taken from [13] for an exact match of experiment setup and comparison. Table-IV has Meta-Tasks requirement and contains instruction volume and data volume of each task which must be scheduled for execution. Table-V represents resource specification including processing speed and bandwidth of the resource. Here resources are basically virtual machines which must be scheduled by the proposed algorithm to the user submitted task for their execution.

Table-IV: Meta-Tasks Requirement (MTR)

Task	Instruction Volume (MI)	Data Volume (MB)
Task(T1)	256	88
Task(T2)	35	31
Task(T3)	327	96
Task(T4)	210	590

Table-V: Resource Specification (RS)

Resource	Processing Speed (MIPS)	Bandwidth (MBPS)
R1	150	300
R2	300	15

Table-VI shows the predicted completion time of individually job/task (T_i) on individually VM/resource (R_j). Initially, the expected completion time (CT) of the individual task is equal to its expected execution time (ET), because initially, all the resources are in idle state i.e. ready time (RT=0) for all the resources. The predictable running time of each job on each resource is calculated by the given data of Table-IV and Table-V. Expected execution time is calculated by Eq. (3).

$$ET_{ij} = \left[\left(MI_i \div MIPS_j \right) + \left(MB_i \div MBPS_j \right) \right] \quad (3)$$

After the first iteration, Eq. (4) is used to compute the probable completion time (CT) of each task T_i on each resource R_j.

$$CT_{ij} = ET_{ij} + RT_{ij} \quad (2)$$

Table-VI: Expected Completion Time

Task/Resource	R1	R2
Task(T1)	2	6
Task(T2)	1	3
Task(T3)	3	8
Task(T4)	3	40

In the first iteration, the proposed algorithm selects task T2 which has minimum execution time for execution on the resource R1 that has smallest completion time. At the same time, resource R2 is free and it has the next minimum completion time for the task T1 so that task T1 will schedule to the resource R2 to allow concurrent execution of tasks and improve load balance among the resources. Now both the resources are busy in executing the tasks. The resource R1 will get free after 1 unit of time and at that time, resource R2 will remain busy for executing task T1. Now, calculation of expected completion time of remaining tasks and assignment of minimum execution time task T3 to the resource R1 is to be done. In next iteration, the resource R1 will be freed and resource R2 is still busy in the execution of task T1. Therefore, task T4 will assign to the resource R1 and complete its execution. To show the execution order of tasks and resource utilization, Gantt-Chart is used. It helps to illustrate start and end time of task execution and makespan of the system can be found easily by Gantt-chart. Fig. 2 shows the Gantt-Chart of RAMM algorithm, which represents the execution order of the tasks. From the figure, it can be inferred that the makespan of the system using proposed Resource-Aware Min-Min (RAMM) algorithm is 7 units and both resources are sharing the load of the system.

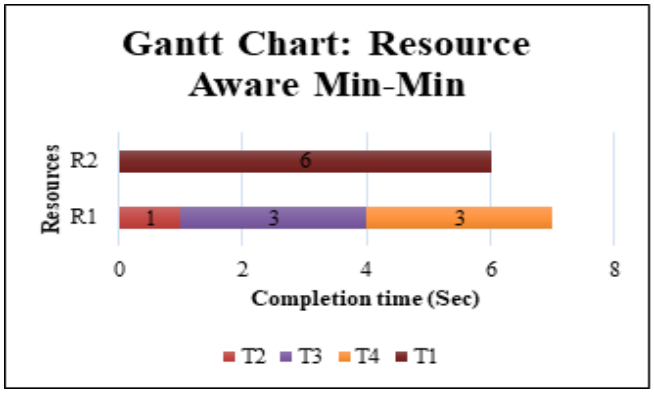

Fig. 2. Gantt-Chart of RAMM Algorithm

VI. COMPARISON AND RESULT DISCUSSION

In the previous section, it has been shown that the makespan of the tasks is determined to be 7 units of time using the proposed RAMM algorithm. This section shall evaluate the makespan for Min-Min, Max-Min, and improved Max-Min algorithms and compare the same with the proposed algorithm. The mentioned algorithms have been tested one by one on the data given in Table-VI. In Fig. 3, execution of the Min-Min algorithm is shown. The makespan of Min-Min algorithms is 9 units of time and only one resource R1 is utilized while resource R2 is idle all the time, which results in load imbalance. Fig. 4 shows the Gantt-Chart of Max-Min algorithm that executes larger task first and the makespan using Max-Min algorithm is found to be same as that for the Min-Min algorithm, 9 units, and it also suffers from the load imbalance problem. Finally, the application of improved Max-Min algorithm gives makespan of 8 units and the load is balanced, Fig. 5 shows the Gantt-Chart of the improved Max-Min algorithm. From these results, proposed RAMM algorithm is far better than Min-Min and Max-Min algorithm in both aspects- makespan and load balancing. Proposed RAMM algorithm and improved Max-Min algorithm balance the load among the resources and execute tasks concurrently. However, the proposed RAMM algorithm gives less makespan than improved Max-Min. Therefore, we can say that our proposed RAMM algorithm is better than all these algorithms and gives a better result.

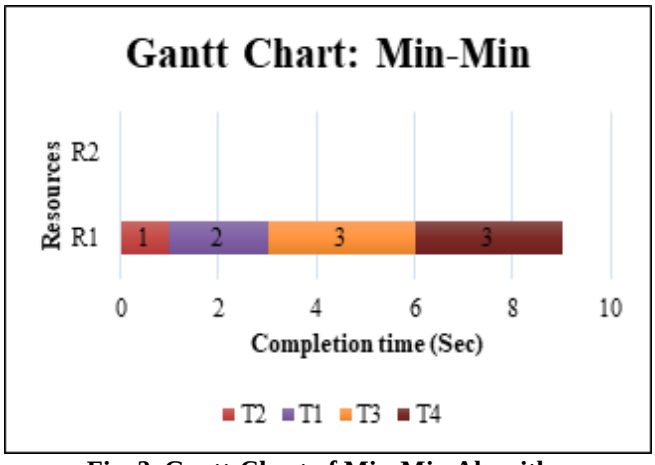

Fig. 3. Gantt-Chart of Min-Min Algorithm

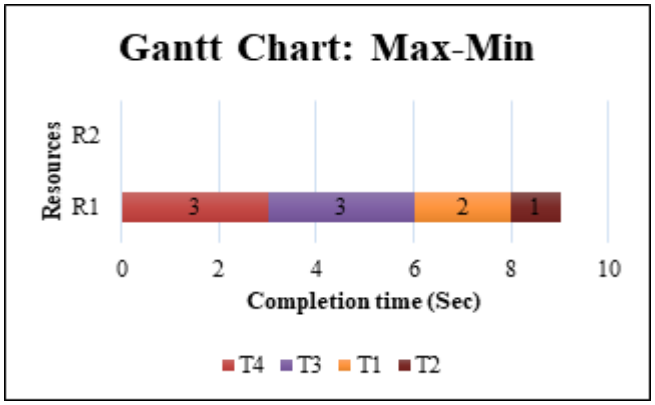

Fig. 4. Gantt-Chart of Max-Min Algorithm

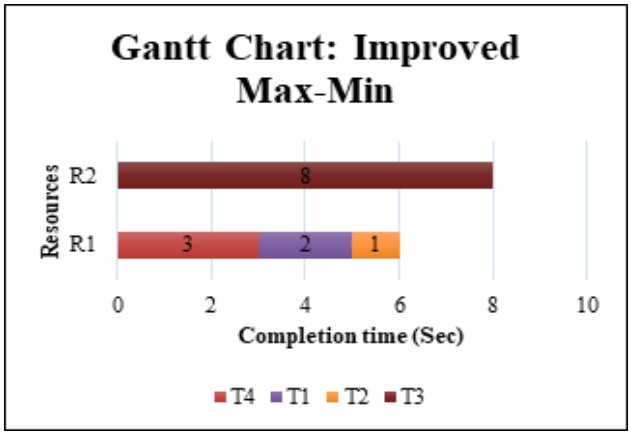

Fig. 5. Gantt-Chart of Improved Max-Min Algorithm

To validate these results, we are taking three more set of problem samples considered by [13]. Table-VII has Meta-Tasks requirement of three problem samples and Table-VIII has resource specification of this problem sample. Table-IX contains the initial end time of tasks on each resource. Min-Min, Max-Min, improved Max-Min and proposed Resource-Aware Min-Min algorithms are executed on this problem sample. The makespan has been calculated and is shown in Table-X. Fig. 6 shows the comparison of makespan of these algorithms. The proposed algorithm (RAMM) gives better makespan in all three-problem sample than other algorithms.

Table-VII: Meta-Tasks requirement (MTR)

Problem	Task	Instruction Volume (MI)	Data Volume (MB)
P1	Task(T1)	256	88
	Task (T2)	35	31
	Task (T3)	327	96
	Task (T4)	210	590
P2	Task (T1)	128	44
	Task (T2)	69	62
	Task (T3)	218	94
	Task (T4)	21	59
P3	Task (T1)	88	20
	Task (T2)	31	350

	Task (T3)	100	207
	Task (T4)	50	21

Table-VIII: Resource Specification (RS)

Problem	Resource	Processing Speed (MIPS)	Bandwidth (MBPS)
P1	R1	150	300
	R2	300	15
P2	R1	50	100
	R2	100	5
P3	R1	300	300
	R2	30	15

Table-IX: Completion Time

Problem	Task	R1	R2
P1	Task(T1)	2	6
	Task(T2)	1	2
	Task(T3)	3	8
	Task(T4)	3	40
P2	Task(T1)	3	10
	Task(T2)	2	13
	Task(T3)	5	21
	Task(T4)	1	12
P3	Task(T1)	1	7
	Task(T2)	1	14
	Task(T3)	1	14
	Task(T4)	1	4

Table-X: Comparison of Makespan of Different Algorithms

Prob	Min-Min	Max-Min	Imp Max-Min	RAMM
P1	9	9	8	7
P2	11	11	13	10
P3	4	4	14	4

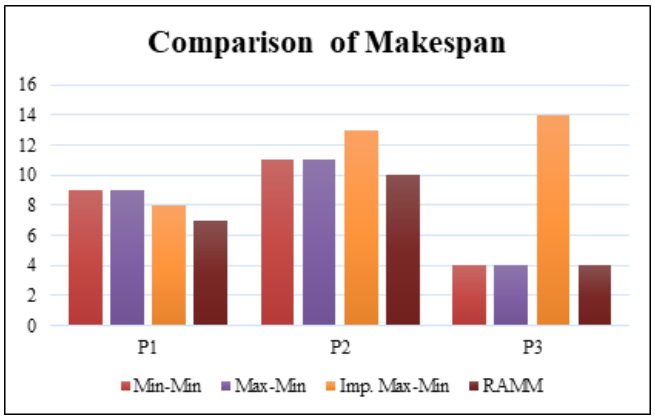

Fig. 6. Makespan Comparison of different Task Scheduling Algorithm

VII. CONCLUSION AND FUTURE WORK

Resource allocation in Cloud Computing is an important aspect. Due to the novelty of Cloud Computing many Task Scheduling algorithms have been proposed. In this paper, we proposed a Resource-Aware Min-Min (RAMM) algorithm for Cloud environment. Various previous proposed algorithms like improved Max-min, Max-Min and Min-Min have been compared with the RAMM algorithm. The proposed algorithm gives better results in the form of Minimum makespan and effective resource load balancing. The resources were not utilized in the Min-Min and Max-Min algorithms, which causes an increase in waiting time of tasks and results in high makespan of the system. On the other hand, in the proposed Resource-Aware Min-Min algorithm the main concern is to reduce makespan and improve resource load balancing. The results confirm that the proposed algorithm provides improved makespan and balance the resource load.

The proposed algorithm works for single cloud environment in which resources are present in a single Cloud. In the future, we shall extend this algorithm for the multi-Cloud environment in which resources (Virtual Machines) are in multiple Clouds environment.

ACKNOWLEDGEMENT

This work was supported by a grant from “Young Faculty Research Fellowship” under Visvesvaraya Ph.D. Scheme for Electronics and IT, Department of Electronics & Information Technology (DeitY), Ministry of Communications & IT, Government of India.

REFERENCES

1. M. Vaezi and Y. Zhang, "Virtualization and Cloud Computing", Wireless Networks, pp. 11-31, 2017.
2. P. Mell and T. Grance, “The NIST definition of cloud computing”, 2011.
3. Z. Xiao, W. Song and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1107-1117, 2013.
4. R. Buyya, C. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009.
5. M. Gill and R. Bawa, "Service Level Agreement Based Fault Tolerant Workload Scheduling in Cloud Computing Environment", International Journal of Grid Computing & Applications, vol. 7, no. 4, pp. 01-08, 2016.
6. M. García-Valls, T. Cucinotta and C. Lu, "Challenges in real-time virtualization and predictable cloud computing", Journal of Systems Architecture, vol. 60, no. 9, pp. 726-740, 2014.
7. W. Voorsluys, J. Broberg, R. Buyya, R. Buyya, J. Broberg, A. Goscinski, “Introduction to Cloud Computing”, John Wiley & Sons, Inc., 2011.
8. M. Mohamaddiah, A. Abdullah, S. Subramaniam and M. Hussin, "A Survey on Resource Allocation and Monitoring in Cloud Computing", International Journal of Machine Learning and Computing, pp. 31-38, 2014.
9. S. Pllana, F. Xhafa, “Programming Multicore and Many-core Computing Systems”, Hoboken, New Jersey, USA:John Wiley & Sons, Inc, 2017.
10. X. He, X. Sun and G. von Laszewski, "QoS guided Min-Min heuristic for grid task scheduling", Journal of Computer Science and Technology, vol. 18, no. 4, pp. 442-451, 2003.
11. Kobra Etminani and M. Naghibzadeh, "A Min-Min Max-Min selective algorihtm for grid task scheduling," 2007 3rd IEEE/IFIP International Conference in Central Asia on Internet, Tashkent, pp. 1-7, 2007.
12. Parsa, "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications, 2009.
13. O. M.Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications, vol. 50, no. 12, pp. 22-27, 2012.

14. Huankai Chen, F. Wang, N. Helian and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), Bangalore, pp. 1-8, 2013.
15. G. Ming and H. Li, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling", Recent Advances in Computer Science and Information Engineering, pp. 217-223, 2012.
16. U. Bhoi, P. N. Ramanuj and W. S. Email, "Enhanced max-min task scheduling algorithm in cloud computing", International Journal of Computer and Information Technology, vol. 2, no. 4, pp. 259-264, 2013.
17. G. Patel, R. Mehta and U. Bhoi, "Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing", Procedia Computer Science, vol. 57, pp. 545-553, 2015.
18. X. Li, Y. Mao, X. Xiao and Y. Zhuang, "An Improved Max-Min Task-Scheduling Algorithm for Elastic Cloud", 2014 International Symposium on Computer, Consumer and Control, 2014.
19. M. Alam, K. Shakil and S. Sethi, "Analysis and Clustering of Workload in Google Cluster Trace Based on Resource Usage", 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), 2016.
20. S. Khan, S. Ali, N. Hasan, K. Shakil and M. Alam, "Big Data Scientific Workflows in the Cloud: Challenges and Future Prospects", Studies in Big Data, pp. 1-28, 2018.
21. S. Madni, M. Latiff, Y. Coulibaly and S. Abdulhamid, "Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities", Journal of Network and Computer Applications, vol. 68, pp. 173-200, 2016.
22. S. A. Ali and M. Alam, "A relative study of task scheduling algorithms in cloud computing environment," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, pp. 105-111, 2016.
23. S. A. Ali, M. Affan and M. Alam, "A Study of Efficient Energy Management Techniques for Cloud Computing Environment," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, pp. 13-18, 2019.
24. G. Sharma and J. Martin, "MATLAB®: A Language for Parallel Computing", International Journal of Parallel Programming, vol. 37, no. 1, pp. 3-36, 2008.

AUTHORS PROFILE

Syed Arshad Ali received his MCA degree from Jamia Hamdard, New Delhi, India. Currently, he is working as a Senior research fellow in the Department of Computer Science, Jamia Millia Islamia, New Delhi, India. His research areas are Dynamic Resource allocation in Cloud Computing, Big data and IoT.

Samiya Khan is a Ph.D. Student in the Department of Computer Science, Jamia Millia Islamia, New Delhi, India since October 2015. Samiya received B.Sc. degree in Electronics and M.Sc. degree in Informatics from University of Delhi, India. She is currently working on a Cloud- based Framework for Big Data Analytics and has several research papers related to the field in reputed publications. Her research interests also include data-intensive computing, big data, machine learning, deep learning and natural language processing.

Mansaf Alam received his doctoral degree in Computer Science from Jamia Millia Islamia New Delhi in the year 2009. He is currently working as an Associate Professor in the Department of Computer Science, Jamia Millia Islamia. He is also the Editor-in-Chief, Journal of Applied Information Science. He is an Editorial Board of some reputed International Journals in Computer Sciences and has published about 24 research papers. He also has two books entitled “Concepts of Multimedia, Book” and “Digital Logic Design, PHI” to his credit. His areas of research include Cloud Computing, Big data analytics, Object-Oriented Database System (OODBMS), Genetic Programming, Bioinformatics, Image Processing, Information Retrieval and Data Mining.