IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 4, Ver. VI (Jul.-Aug. 2016), PP 18-25
www.iosrjournals.org

Optimizing Task Scheduling and Resource allocation in Cloud Data Center, using Enhanced Min-Min Algorithm

Mubarak Haladu[1], Joshua Samual[2]
[1]*MSc. Computer Systems Engineering, Department of Engineering and Computer Science, FTMS College Kuala Lumpur, Malaysia.*
[2]*Senior Lecturer, Department of Engineering and Computer Science, FTMS College Kuala Lumpur, Malaysia.*

Abstracts: *Cloud Computing provide the chance to use computing resources over the internet without owning the infrastructure. The main content of Cloud Computing is to manage Software application, data storage and processing capacity which are assigned to external users on demand through the internet and pay only for what they use. Task scheduling in cloud computing is the biggest challenges because many tasks need to be executed by the available resources in order to meet user's requirements. To achieve best performance, minimize total completion time, minimize response time and maximize resources utilization there is need to address these challenges. This paper studies different task scheduling algorithms and an Enhanced Min-min algorithm is developed. The algorithm uses the advantages of Min-min and avoids its drawbacks. The main idea of the proposed algorithm is to allocate tasks to resources appropriately in order to achieve an effective load balancing and decrease completion time. The experimental results indicate that the proposed algorithm compared to Min-min and Max-min produces better Makespan and improved resources utilization.*

Keywords: *Cloud Computing, Task scheduling, Makespan, Min-Min Algorithm, Max-Min Algorithm.*

I. Introduction

Cloud Computing means both applications and services are moved into the internet (cloud). Cloud computing provides the users with any kind of software or hardware services through the internet on the basis of pay as you use. Cloud computing is defined as a parallel and distributed system which comprise of group of inter-connected and virtualized computers that are provisioned dynamically and presented as one or more unified computing resources based on service level agreements (SLA) established through mediation between the cloud services providers and users [(1).]

Cloud computing delivers infrastructure as a service (IaaS), Platform as a service (PaaS), and Software as a service (SaaS) [(2)]. IaaS provide the users with infrastructure in form of a virtual machine (VM) to run any applications. PaaS provide an application development platform to users or developers to develop their own cloud applications and SaaS allow users to run existing software applications. Cloud Computing is rapidly developing Computation model which has moved the management of hardware, software and other computing resources from users to cloud service providers.

There are so many issues regarding cloud computing data security, energy conservation, service availability, expandable storage management, task scheduling. But task scheduling is usually the main topic of research in cloud computing [(3)]. The primary objective of this paper is to enhance the scheduling policies. In cloud computing, many tasks need to be executed by the resources available in order to achieve high performances, optimal completion time, reduce response time and effective resources utilization [(4)]. Because of these different objectives, there is a need to develop and propose a scheduling algorithm that will be used by task scheduler to appropriately allocate tasks to resources.

The traditional Min-min task scheduling algorithm select the task with minimum execution time and allocate it to the resource expected to give minimum completion time (5). The drawbacks of the min-min algorithm is that it select the smaller tasks first while the task with large execution time will wait until the smaller tasks are done as a result will increase the response time and the overall finishing time. This paper is aim at addressing the drawbacks of the min-min algorithm.

1.1 Task Scheduling in Cloud Computing

Scheduling is referred to as the set of policies for managing the order of work to be executed by a computing system [(6), (7)]. The task scheduling in Cloud data center is a set of instructions and factors that determines and select the task to be executed on the available resources between a collections of possible tasks at a particular time (8). In Cloud data center, the task scheduling algorithms are responsible for allocating the tasks submitted by the users to the available resources. The main advantage of task scheduling algorithm is to achieve a highly performance computing and the best system throughput [(2), (7), (9)]. Scheduling manages the CPU memory and to achieve maximum resource utilization, requires good scheduling policies.

In Cloud Computing Environment, Tasks are submitted to the Data Center Broker by the Users. The Data Center Broker is an intermediary between the Cloud Users and Providers and is responsible for scheduling tasks on Virtual machines (VM). Data Center is a virtual Infrastructure for housing Resources and consists of a number of Hosts. The submitted tasks are scheduled according to the scheduling policies used by the Data Center Broker. Broker communicates directly with the Cloud controller and assigns tasks to Virtual machines in the Host of the Data Center [(10).]

There are many types of Scheduling according to different policies such as preemptive and non-preemptive scheduling, static and dynamic scheduling, Immediate and batch scheduling, centralize and distributed scheduling [(4)]. Scheduling of tasks is considered as the process of selecting the best resource available for tasks execution. Task scheduling Algorithms aim at minimizing the completion time of tasks and maximizing resource utilization in order to meet user requirements.

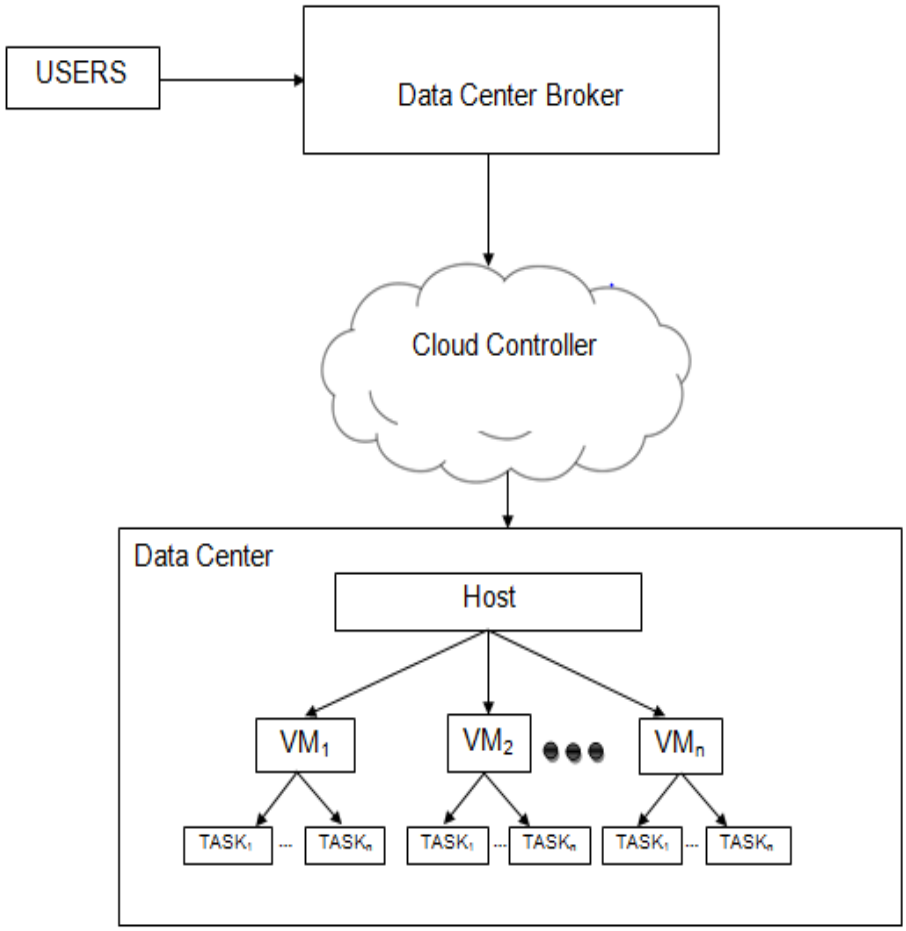

Fig 1: Task scheduling structure

II. Related Work

Cloud computing is a new technology which is advancing day by day. Many scheduling algorithms have been proposed in the past. Some of these task scheduling algorithms are presented below.

First come first serve (FCFS) algorithm assigns tasks to Resources in order of their arrival, the task in the queue that come first is executed [(11)]. The drawback of FCFS is that small tasks at the back of the queue need to wait for the large tasks at the front to be executed.

Opportunistic load balancing algorithm (OLB) Schedules each task in a random order to the next available resource not regarding its expected execution time [(12)]. This algorithm tries to keep all resources busy. If more than one resource becomes idle, then a resource is chosen randomly. OLB produces a very poor Makespan because it doesn't consider the expected execution time.

Minimum Execution Time (MET), allocates each task to the resource with minimum expected execution time, without considering the resource availability at that time [(13)]. The idea of this approach is to assign each task to its best resource. This method results to a severe load imbalance across the resources.

Minimum Completion Time (MCT) assigns each task randomly to the resource expected to give the minimum completion time (MCT) for that task [14]. It combines the advantages of OLB and MET, and avoids there disadvantages.

Round Robin Algorithm allocates resources in circular order and it does not use priority of the tasks [15]. The scheduler allocates a fixed unit of time to each task which is executed in turn. Tasks that are unable to complete during its turn, will go back to the queue waiting for another turn. The drawback of this algorithm is that if the virtual machine is heavy loaded it will take long time to complete execution of all tasks.

Min-Min Algorithm begins with a set of all tasks that are unmapped and compute the completion time of all tasks on each resource [12]. It selects the smallest execution time and assign to the resource expected to complete the job at earliest time. This process is continuously repeated until all the unmapped tasks are assigned to a machine. Min-min algorithm produces a better Makespan when most tasks have small execution time. The algorithm selects small tasks to be executed first and as a result large tasks are delayed.

Max-min Algorithm selects the Machine that has the minimum completion time for all tasks. Then the task with the maximum completion time is chosen and assigned to that machine that has the minimum completion time. The ready time of the machine is then updated. The process is repeated until all the unmapped tasks are assigned to a Machine. The main idea of this algorithm is to minimize the waiting time of large tasks.

Sufferage Algorithm computes the completion time for each task on all available resources. The next two minimum completion times for all tasks are selected. The suffrage value is the difference between the two successive minimum completion times. The task that has the maximum suffrage value is assigned to a machine with minimum expected completion time [12]. If there is more than one similar maximum Sufferage value, it selects the first one without considering the other and this result to a starvation problem.

Saeed P et al., [16], proposed a new Task Scheduling Algorithm, Resource Aware Scheduling Algorithm (RASA) that combines Min-Min and Max-Min algorithms. The Algorithms begin with Min-min when the number of available Resources is odd and allocates the first task; but if it's even Max-min is adopted; the remaining tasks are assigned to an appropriate machine based on this strategy. The simulation result shows this algorithm is perform better than the Min-min and Max-min Algorithms.

Rajwinder K et al., [5] introduced a rescheduling based task scheduling algorithm named improved Min-Min Algorithm (I Min-Min). The algorithm has two stages; in the first stage the traditional Min-Min algorithm is executed and in the other stage the jobs are rescheduled to use the virtual machines that are idle. The algorithm compared with Min-Min produce a better Makespan.

Amit Agarwal et al., [7] proposed a Generalized Priority algorithm and it's compared with FCFS and Round Robin Algorithms. The algorithm prioritized tasks based on their size, such that the task with largest size has the highest priority. The resources are also prioritized according to their MIPS value also the resource with the largest MIPS has the highest priority and therefore the task with highest priority is assigned to the resource with highest MIPS respectively.

Anthony Thomas et al., [17] the author introduced an improved Scheduling algorithm which is based on User Priority and Task length. The algorithm considers two factors; task length and User priority and based on these two criteria order of execution of the tasks will be decided by the datacenter broker. The proposed algorithm is compared with two scheduling algorithms which are based on tasks length and the other based on task priority.

Somayeh T. Dehkordi et al., [11] Proposed TASA (Task-aware scheduling algorithm), which combine both the features of Min-min and Sufferage algorithms. The algorithm is also based on minimum completion time. If the number of tasks is even Sufferage strategy is applied and if the number is odd Min-min strategy is applied. Task allocation strategy changes arbitrary. The Algorithm only concentrated on the number of tasks instead of the available resources.

Guarang Patel et al., [18] present a modification of Load Balanced Min-min (ELBMM) algorithm for Static Meta Task scheduling. At the first stage Min-min strategy is adopted and then the tasks are rescheduled in order to make use of the idle resources effectively and improve the overall completion time. The enhanced Load Balanced Min-min chooses the task with highest completion time and allocates it to the appropriate resource. Theoretical result of ELBMM indicates that it produces better Makespan and better resource utilization as compared to LBMM.

Prerit Chawda et al., [4] proposed The Load Balanced Improved Min-Min (ILBMM) algorithm. At first Min-min strategy is applied, next is reschedule to balance the heavy load resources in order to improve the load unbalance and decrease the overall completion time. The theoretical results show that ILBMM gain better overall performance than Min-min.

1.2 Min-Min Scheduling Algorithm

Step 1: For all submitted tasks in task set; T_i

Step 2: For all Resources; R_j

Step 3: Compute $Ct_{ij} = Et_{ij} + rt_{ij}$ //Expected Completion Time of Task i on Resource j

Step 4: Do while Set is not empty

Step 5: Find Task T_k that give **minimum** execution time

Step 6: Assign Task T_k to Resource R_j that gives minimum expected completion time.

Step 7: Delete Task T_k from the Set

Step 8: Update ready time rt_j for the selected Resources j

Step 9: Update Ct_{ij} for all T_i

Step 10: End Do

Fig 2: Pseudo code of Min-Min Algorithm

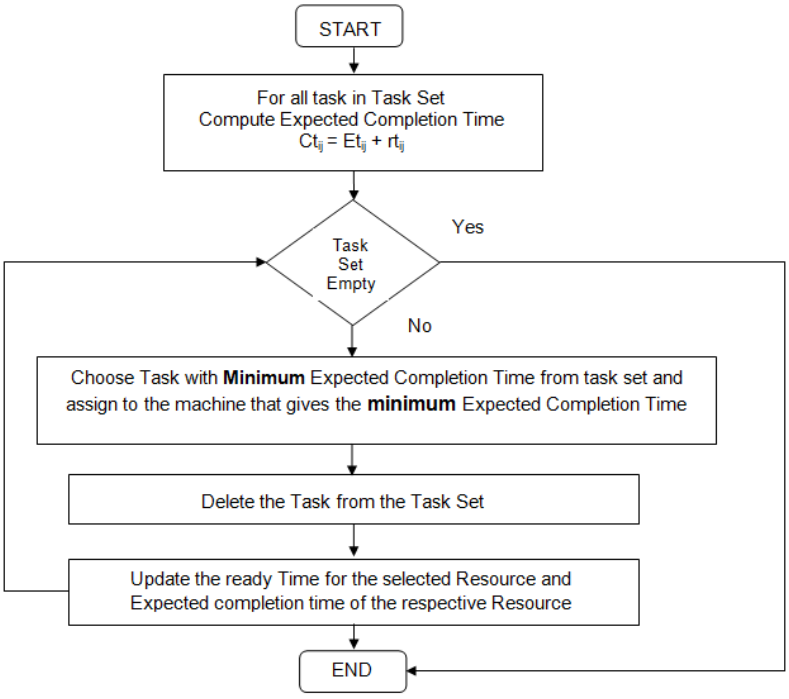

Fig 3: Flow Chart of Min-Min Algorithm

III. Proposed Algorithm

The proposed Enhanced Min-min scheduling algorithm is presented in order to decrease the overall completion time and to have maximum resource utilization in cloud computing environment. The algorithm is a modification of Min-min scheduling algorithm. Min-min selects task with minimum execution time and assign to the resource that gives minimum expected completion time. The drawback of Min-min algorithm is that it executes small tasks first while the large tasks are left in the waiting phase. Some resources may be idle while others overloaded and this would result to increases in the Makespan.

The proposed algorithm starts by calculating the expected completion time for all tasks on each resource. It then selects the task with the minimum execution time and the task with maximum execution time on the resource expected to complete in earliest time. Next compute the difference between the maximum execution time and the minimum execution. If the difference is less or equal to the minimum execution time; it assigns the minimum to the resource that produce it, otherwise assigns the maximum execution time. Remove

the task and update the resource ready time. The process is repeated until only one task is left which is then assign to the resource that gives the minimum execution time. The Pseudo code is shown below.

Step 1: For all tasks in task set; T_i

Step 2: For all Resources; R_j

Step 3: Calculate $Ct_{ij} = Et_{ij} + rt_{ij}$ //Expected Completion Time of Task i on Resource j

Step 4: Do while Set is not empty

Step 5: Find the Task with **minimum** execution time (MinET)

Step 6: Find the Task with **maximum** execution time (MaxET)

Step 7: Find the difference between MaxET and MinET *//Diff = MaxET – MinET*

Step 8: If Diff < = MinET, assign MaxET to the Resource that produces it minimum completion time.

Step 9: Else, assign MinET

Step 10: Delete the task assigned from the Set of tasks

Step 11: Update ready time rt_j for the selected Resources Rj

Step 12: Update Ct_{ij} for all T_i //Expected Completion Time

Step 13: End If

Step 14: End Do

Fig 4: Pseudo code of the proposed algorithm

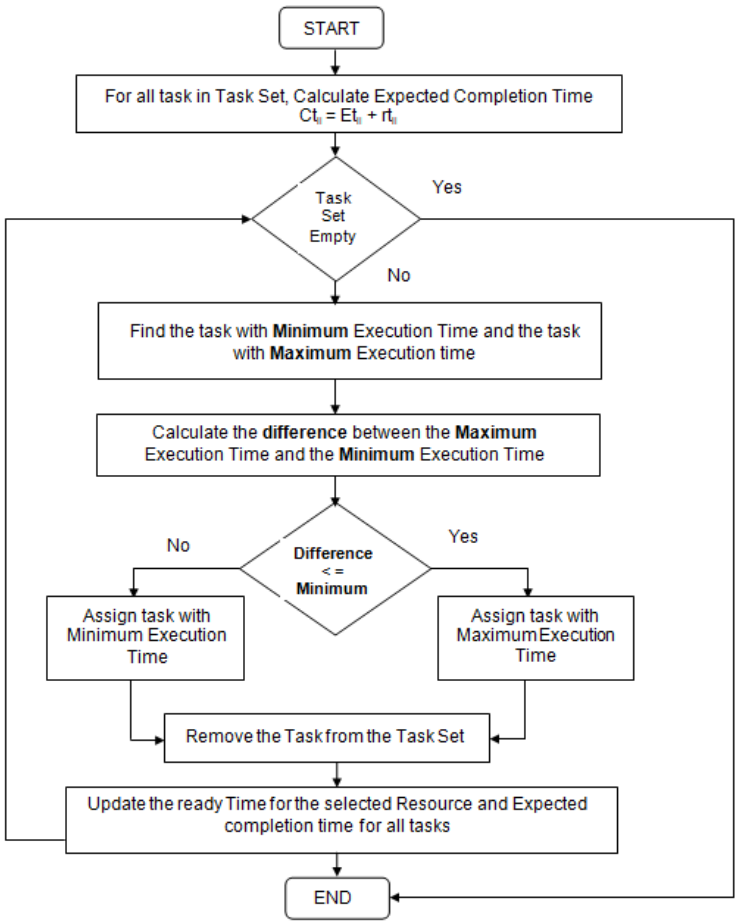

Fig 5: Flow Chart of the Proposed Algorithm

IV. Experiments And Results

To verify the proposed algorithm, the simulation was done using Cloudsim 3.0 Simulator on a 32-bit Windows 7 Operating System, Pentium (R) Dual core CPU 2.30GHz 4GB RAM. The Cloudsim 3.0 is run using Netbeans IDE 8.1. We have created 5 Virtual machines and used Montage and CyberShake Workflows to test the algorithms. We used 25, 50, 100 and 1000 Cloudlets for Montage Workflows while 30, 50, 100 and 1000 Cloudlets for CyberShake with one Data center and one Data center broker. The proposed Enhanced Min-min Algorithm is compared with Min-min and Max-min task scheduling Algorithms.

Table 1: Makespan of comparison using Montage

Algorithm	**Makespan**			
	Montage_25	Montage_50	Montage_100	Montage_1000
Max-Min	132.72	557.36	938.82	4821.82
Min-Min	121.71	536.23	781.76	5147.57
Enhanced Min-Min	105.91	433.79	615.35	4365.35

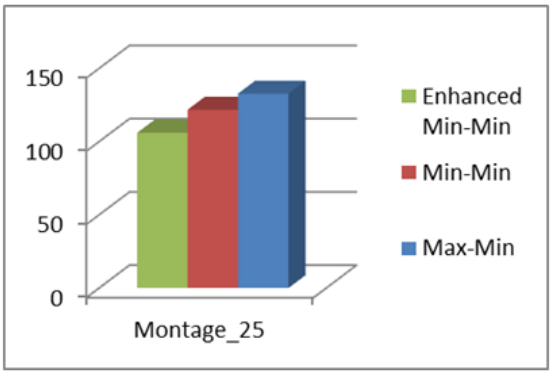

Fig 6: Makespan Bar Chart using Montage_25

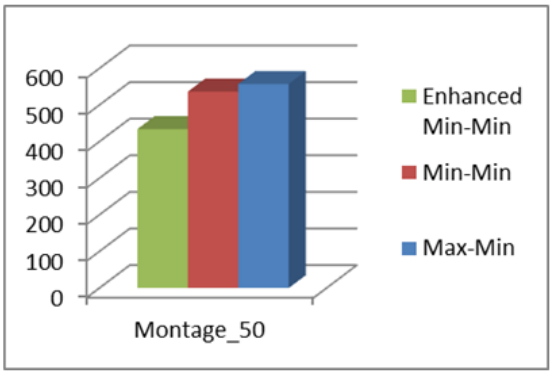

Fig 7: Makespan Bar Chart using Montage_50

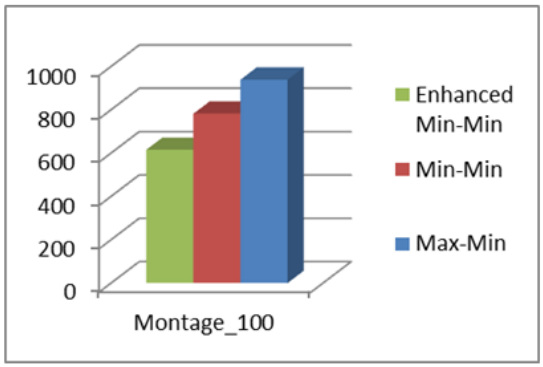

Fig 7: Makespan Bar Chart using Montage_100

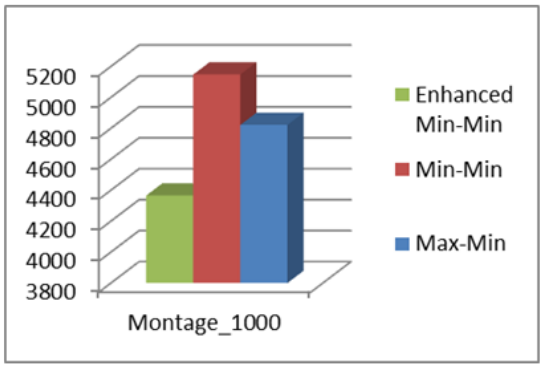

Fig 8: Makespan Bar Chart using Montage_1000

In Fig. 6, Fig. 7 and Fig. 8 above, it can be seen that the proposed Enhanced Min-Min algorithm has better Makespan than Min-Min and Max-min with 25, 50 and 100 tasks respectively and also Min-Min outperforms Max-Min. While in Fig. 8, the proposed approach produced a far better result than the other two, but Max-Min here outperforms Min-Min with 1000 tasks.

Table 2: Makespan of comparison using CyberShake

Algorithm	**Makespan**			
	CyberShake_30	CyberShake_50	CyberShake_100	CyberShake_1000
Max-Min	697.32	881.76	2840.1	8239.72
Min-Min	747.29	1142.01	3372.82	11729.51
Enhanced Min-Min	465.05	784.36	2017.21	9076.38

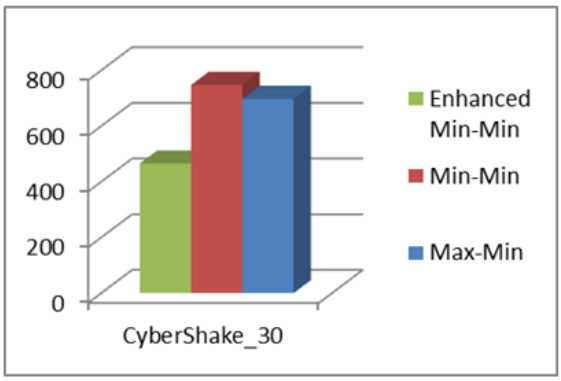

Fig 9: Makespan Bar Chart using CyberShake_30

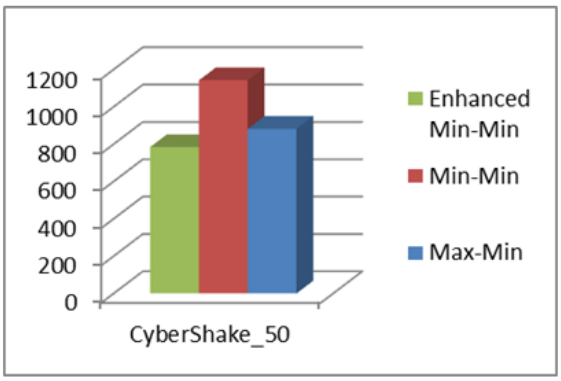

Fig 10: Makespan Bar Chart using CyberShake_50

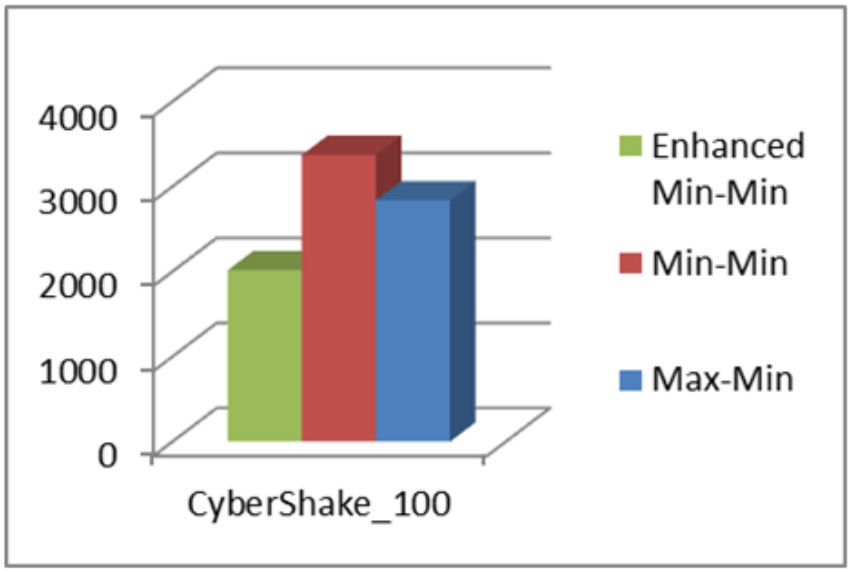

Fig 11: Makespan Bar Chart using CyberShake_100

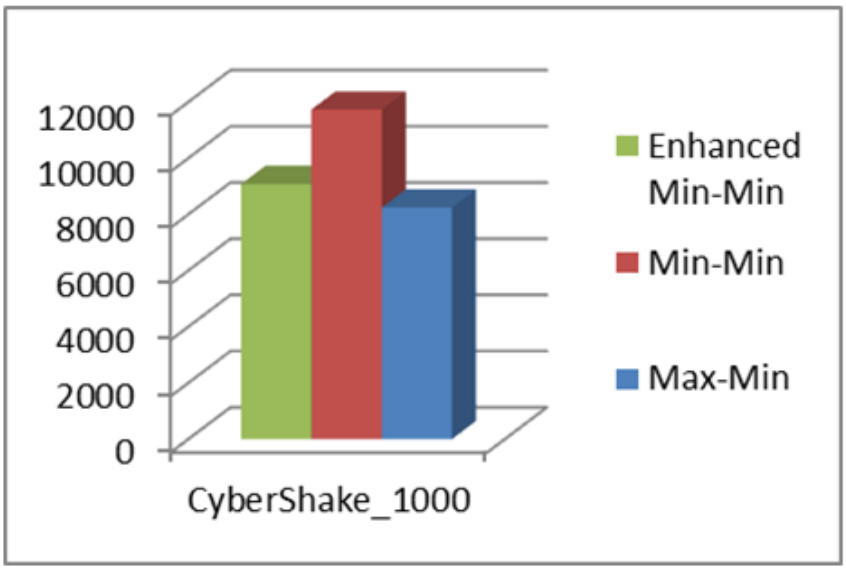

Fig 12: Makespan Bar Chart using CyberShake_1000

Fig. 9, Fig. 10 and Fig. 11 above, clearly indicates that the proposed Enhanced Min-Min algorithm has better Makespan than Min-Min and Max-min with 30, 50 and 100 tasks respectively. But in Fig. 12 above, Max-Min outperforms Min-Min and the proposed approach when the number of tasks was 1000.

V. Conclusion And Future Work

This paper introduces an Enhanced Min-Min Task Scheduling algorithm which is concerned with the overall finishing time (Makespan) and resources utilization. The experiment was done through a simulation environment called Cloudsim. The simulation results are compared with Min-Min and Max-Min Algorithms. The proposed algorithm has a better Makespan than the other two algorithms.

In future, the proposed algorithm can be extended to consider other parameter such as cost of execution and to make comparison between the proposed algorithm with other existing scheduling algorithms.

References

[1]. **Singh, Raja Manish, Paul, Sanchita and Kumar, Abhishek.** *Task Scheduling in Cloud computing: Review.* 6, 2014, International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 5, pp. 40-44. 0975-9646.

[2]. **Lakshmi, R Durga and Asu, N Srinivasu.** *A Dynamic Approach to Task Scheduling in Cloud Computing Using Genentic Algorithm.* 2, 2016, Journal of Theoritical and Applied Information Technology, Vol. 85. 1992-8645.

[3]. **Saxena, Deepika, Chauhan, R K and Kait, Ramesh.** *Dynamic Fair Priority Optiumization Task Scheduling Algorithm in Cloud Computing: Concepts and Implementations.* 2016, I. J Computer Network and Information Security, pp. 41-48.

[4]. **Chawda, Prerit and Chakraborty, Partha Sarathi.** *An Improved Min-Min Task Scheduling Algorithm for Load Balancing in Cloud Computing.* 4, 2016, International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 4, pp. 60-64. 2321-8169.

[5]. **Kaur, Rajwinder and Patra, Prasenjit Kumar.** *Resource Allocation with improved Min-Min Algorithm.* 15, 2013, International Journal of Computer Applications, Vol. 76, pp. 61-65. 0975-8887.

[6]. **Thomas, Anthony, G, Krishnalal and V P, Raj Jagathy.** *Credit Based Scheduling Algorithm in Cloud Computing Environment.* s.l. : Elsevier B.V, 2015. International Conference on Information and Communication Technologies. pp. 913-920. 1877-0509.

[7]. **Agarwal, Amit and Jain, Saloni.** *Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment.* 7, 2014, International Journal of Computer Trends and Technology (IJCTT), Vol. 9, pp. 344-349. 2231-2803.

[8]. **Er-Raji, Naoufal, Benabbou, Faouzia and Eddaoui, Ahmed.** *Task Scheduling Algorithms in the Cloud Computing Environment: Survey and Solutions.* 1, 2016, International Journal of Adavnced Research in Computer Science and Software Engineering, Vol. 6, pp. 604-608. 2277-128X.

[9]. **Bhavisha, Kanani and Bhumi, Maniyar.** *Review on Max-Min Task Scheduling Algorithm for Cloud Computing.* 3, 2015, Journal of Emerging Technologies and Innovative Research (JETIR), Vol. 2, pp. 781-784. 2349-5162.

[10]. **Himani and Harmanbir, Singh Sidhu.** *Comparative Analysis of Scheduling Algorithms of Cloudsim in Cloud Computing.* 16, s.l. : International Journal of Computer Applications, 2014, Vol. 97. 0975-8887.

[11]. **Dehkordi, Somayeh Taherian and Bardsiri, Vahid Khatibi.** *TASA: A New Task Scheduling Algorithm in Cloud Computing.* 4, 2015, Journal of Advances in Computer Engineering and Technology, Vol. 1, pp. 26-32.

[12]. **Reda, Naglaa M.** *An Improved Sufferage Meta-Task Scheduling Algorithm in Grid Computing Systems.* 10, 2015, International Journal of Advanced Research, Vol. 3. 2320-5407.

[13]. **Kokilavani, T and Amalarethinam, D I George.** *Load Balanced Min-min Algorithm for Static Meta-Task Scheduling in Grid Computing*. 2, 2011, International Journal of Computer Applications, Vol. 20. 0975-8887.

[14]. **Patel, Guarang, Mehta, Rutvik and Bhoi, Upendra.** *Enhanced Load Balanced Min-Min algorithm for Static Meta Task Scheduling in Cloud Computing.* s.l. : Elsevier B. V, 2015. 3rd International Conference on Recent Trends in Computing. pp. 545-553. 1877-0509.

[15]. **Rajeshkannan, R and Aramudhan, M.** *Comparative Study of Load Balancing Algorithms in Cloud Computing Environment.* 20, May 2016, Indian Journal of Science and Technology, Vol. 9, pp. 1-6. 0974-6846.

[16]. **Parsa, Saeed and Entezari-Maleki, Reza.** *RASA: Anew task scheduling algorithm in Grid Environment*. 2009, in World Applied Sciences Journal 7, pp. 152-160.

[17]. **Santhosh, B. and H, Manjaiah D.** *An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing.* 2014, International Journal for innovative Research in Computer and Communication Engineering, pp. 84-88.

[18]. **Lakra, Atul Vikas and Yadav., Dharmendra Kumar.** *Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization.* 2015, International Conference on Intelligent Computing, Communication & Convergence, pp. 107-113.

[19]. **Mohammadi F., Jamali S. and Bekravi M.** *Survey on Job Scheduling algorithms in Cloud Computing*. 2014, International Journal of Emerging Trends & Technology in Computer Science, pp. 151-154.

[20]. **Gokilavani M., Selvi S., Udhayakumar C.** *A survey on Resource Allocation and Task Scheduling Algorithms in Cloud Computing Environment*. 2013, International Journal of Engineering and Innovative Technology (IJEIT), PP. 173-178.