An Optimized Task Scheduling Algorithm in Cloud Computing Environment

Shrinidhi Chaudhari[1] Dr. Chirag Thaker[2] Dr. Bhavesh Borisaniya[3]
[1]Student [2,3]Professor
[1,2,3]Department of Computer Engineering
[1,3]SSEC, Bhavnagar Gujarat, India [2]LDCE, Ahmedabad, Gujarat, India

Abstract— Cloud Computing is widely known area in IT industry now a days. Cloud computing provides software (Operating systems, Management software, Applications etc.) as well as hardware (Servers, Data centre, Network systems, Storage, etc.) services to the user. As the number of users increases the usage of cloud also increases. Hence for availability of resources and faster execution of user's request the task scheduling is important aspect. In this paper we have proposed a novel task scheduling algorithm for cloud environment that will cause parametric improvement in make span and throughput of task execution.

Key words: Optimized Task Scheduling Algorithm, Cloud Computing Environment

I. INTRODUCTION

Cloud computing offers versatile services over network. The term versatile computing refers to the ability to dynamically acquire computing resources. Scheduling can be done at three levels task, resource or workflow. Task scheduling of and resource allocation are the important features of the cloud which affect the performance of a system. In this paper we have focused on the task scheduling.

Scheduling considers different parameters to increase the system performance. Tasks can be data entering, data processing, software access, or storage functions. In task scheduling the users submit their tasks to the cloud scheduler. The cloud scheduler gets the status of available resources, properties and allocates the different tasks on different resources. Better scheduling always gives the optimal solution in terms of efficient utilization of cloud resources.

The cloud became very popular now a day because of its pay per usage model. As the user's requests and cloud users increases scheduling is a big challenge to provide QoS to the users. So to satisfy the user's demand we need a dynamic scheduling algorithm that will schedule the user's requests optimal way with fairness of small tasks and bigger tasks, with good load balancing, less makespan and less waiting time to the bigger tasks as possible.

Rest of the paper is organized as follows: Section 2 discusses the classification of task scheduling. Section 3 discusses the existing scheduling algorithms. Section 4 proposed method and its implementation with conclusion and references at the end.

II. CLASSIFICATION OF TASK SCHEDULING

There are three types of scheduling [2] as shown in fig 1 Resource scheduling, Task scheduling and Workflow scheduling.

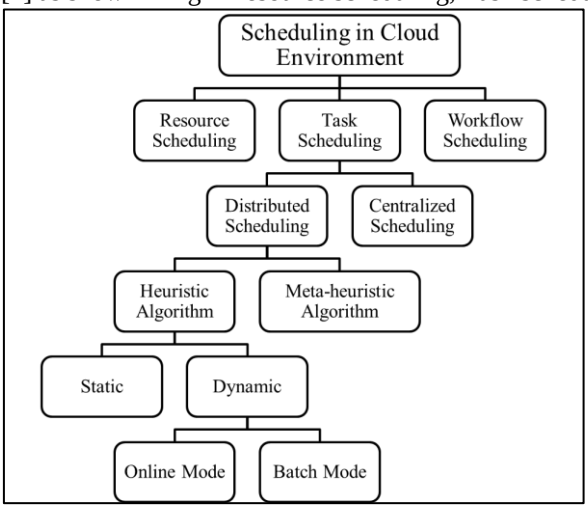

Fig. 1: Classification of scheduling in cloud environment

Resource scheduling schedules virtual machine into physical machine. Workflow scheduling schedules workflows consists of an entire job in a suitable order. Task scheduling divided into two types distributed scheduling and centralized scheduling. Distributed scheduling has only one scheduler for mapping, while centralized scheduling uses partitioned among different schedulers. Distributed scheduling further categories in two, Heuristic algorithm and Meta-heuristic algorithm.

Heuristic algorithm gives optimal solution at a reasonable computational cost without guarantee, while meta-heuristic algorithm tries to find the global optimal solution. Heuristic algorithms are sub-divided into, Static scheduling and dynamic scheduling. In static scheduling virtual resources are assigned statically and in dynamic scheduling tasks are scheduled as soon as they arrive in the system. Dynamic scheduling is better than static scheduling. Dynamic scheduling is again sub-divided into, online mode and batch mode. In online mode tasks are assigned one by one once they arrive in the system. Most fit task scheduling algorithm [4], Minimum Completion Time [4], Minimum Execution Time [4], and Online Batch [4] belongs to online mode. In Batch mode Jobs are taken into a collection when they arrive in the system. Examples are Round Robin [4], First Come First Serve [4], Min-Min [4], and Max-Min [4].

III. EXISTING ALGORITHM FOR SCHEDULING

Following existing algorithms are used in cloud environment.

A. Round Robin (RR)

This algorithm runs the processes in circular queue. The main focus of this algorithm is to distribute the equal load over all resources with the help of time slice or quantum [4]. This algorithm gives better response time and load balancing compared to other algorithms. It is very simple process.

B. First Come First Serve (FCFS)

This algorithm allocates the resources to the task in order in which task arrives. First task will execute [4]. It is very easy one. This algorithm is non-pre-emptive.

C. Shortest Job First (SJF)

In this algorithm the jobs are queued with shortest execution time taken into considered first so that longest job will execute in last [2]. This algorithm is non-pre-emptive.

D. Priority Scheduling

This algorithm is pre-emptive algorithm in which all task has its priority predefined. According to the priority jobs are executed whereas higher priority jobs executed first than lower priority jobs has to wait. The main disadvantage of this algorithm is starvation [6].

E. Min-Min Algorithm

Min-Min algorithm executes the short tasks first and then takes long task to execute [4].

F. Max-Min Algorithm

Max-Min algorithm executes the long tasks first and then takes short task to execute [4].

G. Earliest Deadline First (EDF)

In this algorithm, which task has earliest deadline will be executed first [6].

H. Sufferage Algorithm

This algorithm calculates the sufferage value of each task as difference between its second earliest completion time and earliest completion time. Each task is then scheduled as per the sufferage value [5].

I. Resource Aware selection Algorithm (RASA)

If the amount of tasks is an even number it uses Max-min method to allocate resources to the tasks and if the amount of tasks is an odd number it uses Min-Min method [6].

Algorithm	Parameters considered	Advantage	Disadvantage
RR	Arrival time Time quantum	Simplest algorithm Load balance more fairly Starvation free	Waiting time is more than all. Pre-emption of task is required.
FCFS	Arrival time	Simple, fast and fair	Waiting time is more.
SJF	Average waiting time	Increase throughput Better performance than RR [3] Waiting time is lesser than FCFS [3]	Leads to unfairness and starvation.
Priority	Average Waiting time	Waiting time is less	Indefinite blocking or starvation.
Min-Min	Makespan Expected completion time	Better makeapan compared to other [4]	Poor load balancing. Delays bigger tasks for long.
Max-Min	Makespan Expected completion time	Reduces makespan	Smaller tasks have to wait for longer time.
EDF	Deadline	Optimal all deadlines are met	Behaves badly under overloaded. Need dynamic priority

RASA	Makespan	Reduces the makespan Concurrency in small and large tasks.	Doesn't consider the deadlines and arriving time.
GA	Makespan	Better performance and efficiency in terms of makespan	QoS factors and heterogeneous environments can be considered [7].
Sufferage Heuristic	Minimum completion Time	Better makespan and load balancing.	Based on a sufferage value.

Table 1: Existing Scheduling Algorithms

J. Genetic Algorithm (GA)

It is a search technique to find optimized solution. In this algorithm firstly random population is generated then evaluated and from that best will be selected. After that Selection, Crossover, Mutation and Evaluation function is applied [1].

Table I summarizes the described scheduling algorithms.

IV. PROPOSED ALGORITHM AND IMPLEMENTATION

Most of the existing algorithm focuses on one or two parameters and have disadvantages like poor load balancing, starvation to the bigger tasks, more waiting time and unfairness so to overcome from these disadvantages.

We proposed an algorithm that will improve the performance of the scheduling process. Our proposed algorithm tries to fulfil scheduling criteria and better parametric improvement compare to existing algorithms.

Algorithm starts with the expected completion time.VM monitor checks the resource availability at the resource pool if resource is available then it will create a task pool and find the task that have minimum execution time. However, if the resource is unavailable then it will wait for some time. After finding the task which is having minimum execution time the fitness function is applied to that task and based on the fitness function the fittest resource will be assigned to that task that will gives minimum expected completion time to that task. Expected completion time and task set will be updated for the respective resources.

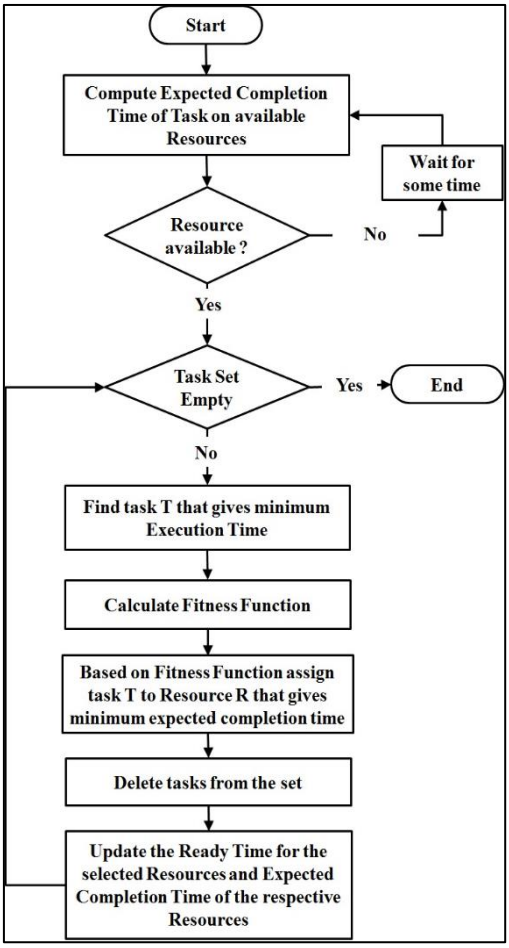

Fig. 2: Flowchart of proposed algorithm

A. Fitness Function

The fitness function is a function that finds the resource to be allocated based on the input parameters. A fitness function finds an optimal solution for the problem. It can be different for different cases. The fitness function will be design such as it will optimize makespan of the task scheduling, increase the throughput of the system, fairness to the larger tasks and better load balancing. The fitness function we used to improve the makespan is as follows:

- Input: Queue (Task)
- Output: Tasks are allocated to the resources

1) *Step 1: Create a queue for nodes and sort on resource utilization.*
2) *Step 2: Check the priority of request.*
   - If the priority of task is high, allocate that task to less utilized instance from queue.
   - Else the priority of task is low, allocate that task to any random instance.

B. Theoretical Analysis

According to the proposed task scheduling algorithm here is the theoretical analysis is done which shows the performance of the system.

Assume that the task scheduler has five meta-tasks and four resources as given below, Table 2 shows the volume instruction and priority of task. Table 3 shows the processing speed and CPU Utilization of resources.

Tasks	T1	T2	T3	T4	T5
Inst. Vol.	620	400	344	700	500
Priority	3	2	1	4	5

Table 2: Volume of Instruction in Tasks

Resources	R1	R2	R3	R4
Processing Speed	100	200	300	50
CPU Utilization	25%	18%	10%	28%

Table 3: Processing Capabilities of Each Resources

Table 4 shows calculations of expected execution time of each task on each resource by equation (T1/R1) [8]. After calculating the expected execution time of all task on each resources the task are sorted as per the minimum execution time. Table V shows the sorted task.

Resources are also sorted by CPU utilization as shown in Table 6. One by one all tasks are executed by fitness function and resource assigned to that task according to the priority type.

	R1	**R2**	**R3**	**R4**
T1	6.20	3.1	2.07	12.4
T2	4.00	2.0	1.33	8.00
T3	3.44	1.72	1.15	6.88
T4	7.00	3.5	2.33	14.00
T5	5.00	2.5	1.67	10.00

Table 4: Expected Execution Time of Each Task on Each Resource

Tasks	T3	T2	T5	T1	T4
Inst. Vol.	344	400	500	620	700
Priority	1	2	5	3	4

Table 5: Volume of Instruction in Tasks after Sorting

Resources	R3	R2	R1	R4
Processing Speed	300	200	100	50
CPU Utilization	10%	18%	25%	28%

Table 6: Processing Capabilities of Each Resource after Sorting

The fig 3 shows the assignment of each task to respective resource as per the proposed algorithm. Tasks assign as per the minimum execution time and resources are assigned as per the minimum resource utilization. Proposed algorithm assigns the task in the sequences of T3-T2-T5-T1-T4 to respective R4-R1-R3-R3-R2 resources. Performing the theoretical analysis of proposed algorithm for five tasks it gives makespan 6.88. In proposed algorithm all the resources are utilized properly and parallel so that it gives the less makespan than the existing Min-Min algorithm. Existing Min-Min algorithm theoretical calculation also carried out and it gives makespan 8.55 shown in fig 4. The resource R3 utilized every time for all tasks and rest of the resources doesn't utilized properly so it giving the more makespan. Min-Min algorithm allocates task as per the minimum execution time on resources.so the allocation of task sequence is T2-T4-T1-T3-T5 on the resource.

C. Experimental Setup and Results

The proposed algorithm is implemented in Amazon EC2. We carried out our experiments on 10 machine instances of Ubuntu on EC2. For the experiments we assigned 100 tasks to the scheduler. We then fetch the completion time of each task. Then we calculated the makespan on all the resources and maximum makespan is considered for analysis.

Fig. 5 shows the comparison of proposed algorithm with Min-Min algorithm. From the figure we can deduce that proposed algorithm take less makespan than Min-Min algorithm.

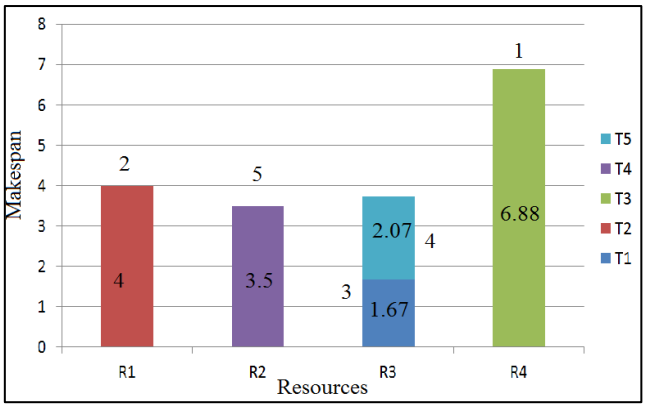

Fig. 3: Proposed algorithm with reduced makespan

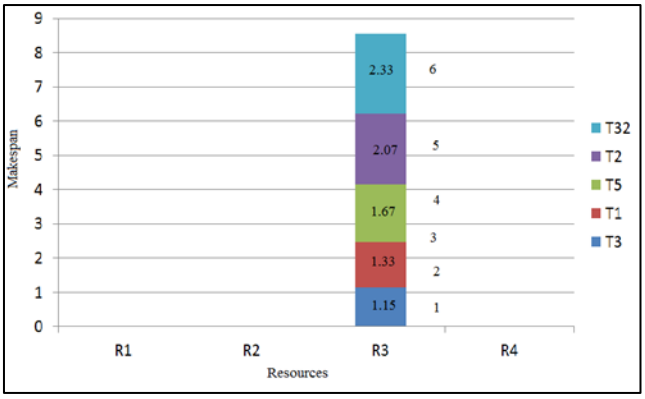

Fig. 4: Makespan of Min-Min algorithm

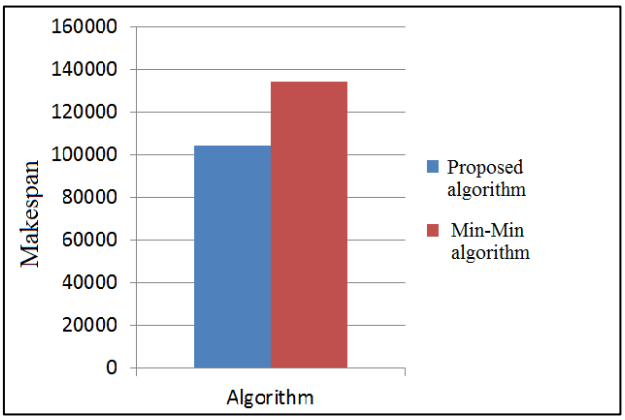

Fig. 5: Makespan of Min-Min algorithm and proposed algorithm

V. CONCLUSION

A study of scheduling algorithm is carried out in this paper. After analysing the various scheduling algorithms for cloud computing environment, this proposed work tries to eliminate the drawback of existing algorithm that can achieve the parametric improvement in scheduling algorithm. This algorithm will decrease the makespan, do better load balancing, and increase the throughput of the system.

REFERENCES

[1] Rajveer Kaur and Supriya Kinger, "Analysis of Job Scheduling Algorithms in Cloud Computing", International Journal of Computer Trends and Technology (IJCTT), vol 9, no 7, March 2014.
[2] Teena Mathew, K. Chandra Sekaran and John Jose,"Study and Analysis of Various Task Scheduling Algorithms in the Cloud Computing Environment", in International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014. IEEE, pp. 658-664.
[3] Monica Gahlawat and Priyanka Sharma, "Analysis and Performance Assessment of CPU Scheduling Algorithms in Cloud using Cloud Sim", International Journal of Applied Information Systems (IJAIS), vol. 5, No. 9, July 2013.
[4] Zhao Yong, Chen Liang, Li Youfu, Tian Wenhong, "Efficient Task Scheduling for Many Task Computing with Resource Attribute Selection", China Communications, vol. 11, no. 12, pp. 125-140, December 2014.
[5] V. M. Sivagami and K. S. Easwarakumar, "Study on Resource Management and Scheduling in Computational Grid", International Journal of Scientific Engineering and Applied Science (IJSEAS), vol. 2, no. 1, January 2016.
[6] Savitha. P and J Geetha Reddy, "A Review Energy-Efficient Task Scheduling Algorithms in Cloud Computing", International Journal Of Scientific & Technology Research, vol. 2, no. 8, August 2013.
[7] Pinal Salot, "A Survey of Various Scheduling Algorithm in Cloud Computing Environment", International Journal of Research in Engineering and Technology (IJRET), vol. 2, no. 2, February 2013.
[8] Vandna Kumari, Mala Kalra and Sarbjeet Singh, "Independent Task Scheduling in Cloud Environment Using Big Bang-Big Crunch Approach", In 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), 2015. IEEE, pp. 1-4.