# TBS: A Threshold Based Scheduling in Grid Environment

Sanjaya Kumar Panda
Department of Computer Science and Engineering
National Institute of Technology
Rourkela, India
sanjayauce@gmail.com

Pabitra Mohan Khilar
Department of Computer Science and Engineering
National Institute of Technology
Rourkela, India
pmkhilar@nitrkl.ac.in

*Abstract*— **Task scheduling is the most challenging work in grid environment. Scheduling approach is based on various criteria's like Quality of Service (QoS), deadline, security, the arrival time of the task, suffrage value etc. It is difficult to get the optimal solution with a unified approach. In this paper, we propose a threshold based technique to decide the task assignments to the machines. It also describes a value to balance the load. Our experimental analysis shows better results than traditional task scheduling algorithms such as Minimum Execution Time (MET), Minimum Completion Time (MCT), Min-Min, Max-Min, K-Percent Best (KPB) and sufferage heuristic.**

***Keywords-threshold based scheduling; grid environment; heterogeneous; task scheduling; quality of service***

## I. INTRODUCTION

Grid computing is used to apply many resources present in an environment for a single job. Job is divided into the huge number of tasks [4]. It is possible if and only if there is no interdependency. Here, the divided chunk is called Meta Task [14]. Directed Acyclic Graph is used to find the interdependency among tasks [15]. Grid Resource Broker (GRB) is responsible for splitting the job into small tasks [3]. Scheduling problem is categorized as Non Polynomial Complete (NPC) problem [6], [10], [11], [14]. There are many grid tasks scheduling such as minimum completion time, minimum execution time, min-min, max-min, RASA [11], sufferage [6], k-percent best (KPB) [6], SIM$^2$ [10] used to schedule the tasks. But, each approach has some limitations. The aim of scheduling is to minimize the Makespan [9], maximize the resource utilization and balance the load. Scheduling can be splited into two phases. Resource selection is done in first phase according to job requirements. Job assignment is carried out in the second phase [13].

Quality of Service can be speed of machines, bandwidth, deadline and task requirements [1] [17]. Tasks are divided into two types. First one is the high QoS. Second one is the low QoS. High QoS has more priority than low QoS [16]. Some tasks do not require QoS. Such tasks are not assigned to high QoS machine. Otherwise, it leads to more Makespan [1]. If a task has completed before the deadline then QoS of that task is satisfied [5].

Grids are of various types like data, computational, service grid. A scheduling is said to be perfect if it combines all types of grid [7]. Data grid consists of data collection, storage, and access from a huge number of databases. To manage huge amount of data, data replication is used. It improves fault tolerance, response time and decrease schedule cost and transfer time [7]. Computational grid gives computation power for scheduling jobs. In scheduling, computation and communication are two important things. The time spent by a task on a machine, called Computation time. In other way, time requires to transfer a task from GRB to a machine, called Communication time. In a heterogeneous environment, each machine has different processing power, memory requirement and interfaces (I/O). In computational grids, each machine has different communication cost [9].

The rest of the paper is organized as follows: the related works are discussed in Section 2. Section 3 represents the preliminaries and traditional scheduling approaches. It explains MET, MCT, Min-Min, Max-Min, KPB and sufferage heuristic with a numerical example. The scheduling mechanism of TBS is proposed in Section 4. It also introduces the performance metrics. Section 5 shows the experimental analysis with a numerical illustration. Section 6 concludes the paper.

## II. RELATED WORKS

Many researchers have been proposed grid task scheduling algorithm based on quality of service, security, genetic algorithm and many more. Maheswaran et al. introduced one batch mode heuristic and two immediate mode heuristic [6]. Sufferage heuristic is a batch mode heuristic. It schedules the tasks based on sufferage value. Two immediate mode heuristics are Swithing Algorithm (SA) and K-Percent Best (KPB). SA uses MET and MCT in a cyclic manner to balance the load. KPB uses only a subset of machines. Xiaoshan et al. introduced QoS task scheduling algorithm based on min-min heuristic [16]. It is a significant improvement over traditional min-min method. Load imbalance is reduced in QoS min-min heuristic. Dong et al. proposed a grid task scheduling based on QoS task priority grouping and deadline of the task [1]. It groups the task based on resources. If tasks are not divided into groups, it underperforms [8]. Munir et al. developed a QoS sufferage heuristic [8] which uses traditional sufferage heuristic introduced by Maheswaran et al. It runs the high QoS tasks only in high QoS hold machines. Patel et al. addresses workflow based jobs in a multi cluster grid [12]. It satisfies QoS requirements as well as better workload allocation policy. It increases Coefficient of Variation (CV) to measure the performance between workflow failures and arrival rates.

## III. PRELIMINARIES

Grid task heuristics are of two types. Firstly, if a task is assigned to a machine as early as it reach to the scheduler called as immediate mode heuristic. Secondly, if the tasks are gathered together and assigned to the machines at prescheduled time called as batch mode heuristic [6]. In this section, we discuss traditional heuristics using an example shown in Table I. "X" sign indicates the task cannot be executed in the corresponding machine. Values are in seconds.

TABLE I. EXECUTION TIME OF TASKS

| **Tasks** | **Machines** | | |
|---|---|---|---|
| | ***$M_1$*** | ***$M_2$*** | ***$M_3$*** |
| ***$T_1$*** | 10 | X | X |
| ***$T_2$*** | 5 | 9 | 13 |
| ***$T_3$*** | 8 | 14 | 15 |
| ***$T_4$*** | X | 5 | 7 |
| ***$T_5$*** | 10 | 12 | 20 |

### *A. Scheduling Algorithms*

*1) Minimum Execution Time (MET):* MET is based upon First Come First Served (FCFS) order. The resource which takes less time for the task will be scheduled first. It doesn't focus on earliest completion time. So, it leads to load imbalance. The time require to schedule a given task to the resources are O (m) [2]. Here, m is the total number of tasks. Makespan of MET is 33. The gantt chart of MET is shown in Figure 1.

*2) Minimum Completion Time (MCT):* MCT is also based upon FCFS order. The resource which takes less completion time for a task will be scheduled first. It focuses on earliest completion time. Load imbalance is reduced to some extent using this approach. The time complexity of MCT is same as MET. Makespan is 20. The gantt chart of MCT is shown in Figure 2.

*3) Min-Min:* It is a mixed approach. It considers both MET and MCT. It selects small task first, it means the task requires less execution time scheduled first. Then, it assigns the task using MCT. It requires $O(m^2n)$ time [2]. Here, n is the total number of machines. As it selects small task first, larger tasks are facing starvation problem. Makespan is 23. The gantt chart of Min-Min is shown in Figure 3.

*4) Max-Min:* It is very similar to Min-Min. It avoids starvation of larger tasks. Instead of choosing small tasks, it selects a larger task first. Then, it applies MCT to assign the tasks. The time complexity of Max-Min is same as Min-Min. Makespan is 19. The gantt chart of Max-Min is shown in Figure 4.

*5) Sufferage:* Sufferage heuristics finds the sufferage value. Sufferage value is the difference between the second earliest completion time of a task in a machine and first earliest completion time of a task in a machine [6]. Let us consider task $T_3$ shown in Table I. Sufferage value is 7 – 5 = 2. The task which suffers most is scheduled first. The time complexity of sufferage is $O(m^2n)$ [6]. Makespan is 22. The gantt chart of sufferage is shown in Figure 5.

*6) K-Percent Best (KPB):* To map the tasks in KPB heuristics, KPB considers a subset of machines [6]. The value K is used to map a task to a machine. Subset is created by choosing m × (k/100) best machine where k is in between (100/m) and 100 [6]. The time complexity is O(m log m) [6]. Makespan is 21. We assume k value as 67%. The gantt chart of KPB is shown in Figure 6.

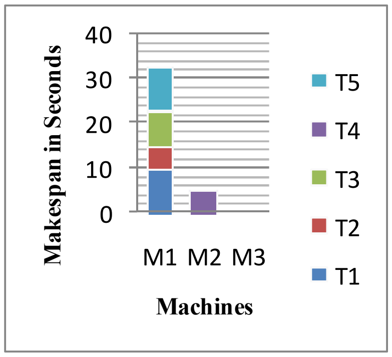

Figure 1. Gantt chart for MET

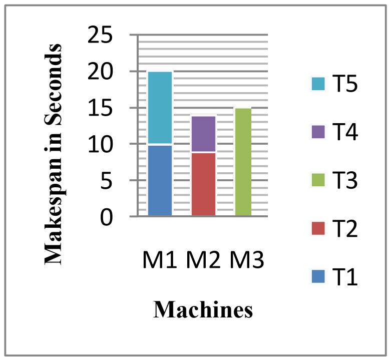

Figure 2. Gantt chart for MCT

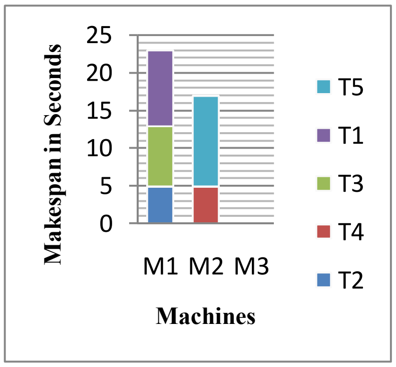

Figure 3. Gantt chart for Min-Min

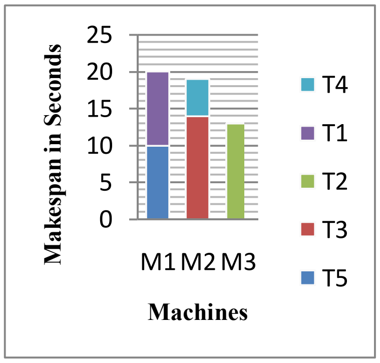

Figure 4. Gantt chart for Max-Min

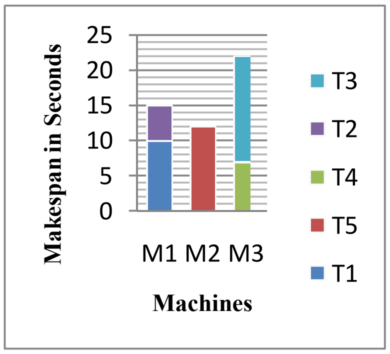

Figure 5. Gantt chart for Sufferage

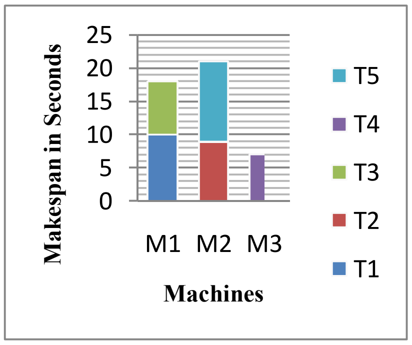

Figure 6. Gantt chart for KPB

## IV. Proposed Algorithm

### *A. Description*

In our proposed algorithm, two queues are used. First queue is used to place Meta Tasks (MTs) with respect to arrival times. This queue is termed as Task Queue (TQ). Second queue is maintained by the tasks individually. This queue is termed as Machine Queue (MQ). Threshold is a value used to balance the load as well as assign the task to the machine. At each steps the value is updated and assignment is done until there is no task in TQ. Tasks are categorized into two types. Firstly, the task can run only on a single machine, it is termed as Enforce Meta Task (EMT). Secondly, the task can run on more than one machine, it is termed as Non-Force Meta Task (NMT). In scheduling, Makespan can be reduced if and only if we schedule NMT properly. We assume that NMT in TQ is sorted order. In our approach, two functions are used. First ($F_1$) is for execution time and second ($F_2$) is for count function. $F_1$ takes two parameters: task $T_i$ and machine queue $MQ_j$. But, $F_2$ takes only one parameter machine $M_k$. M denotes the number of tasks and X denotes the number of EMT. MMQL stands for Maximum Machine Queue Length. It is used to determine the index of MQ.

### *B. Algorithm*

```
1.  Store the MTs in TQ according to the arrival time.
2.  Find EMTs.
3.  Assign EMTs to respective machine.
4.  Update $F_2$ [$M_k$].
5.  Calculate Pre-compute Makespan (PM).
6.  Set the PM as the Threshold (T).
7.  Delete EMTs from TQ and its MQ.
8.  Update TQ.
9.  Set j = 0.
10. for i = 0 to M – X – 1
11.     Set $M_k = MQ_j$.
12.     if (( $F_1$ [ $T_i$, $MQ_j$ ] ) + $F_2$ [ $M_k$ ] <= T )
13.             Assign NMT $T_i$ to $MQ_j$.
14.             Set $F_2$ [ $M_k$ ] + = $F_1$ [ $T_i$, $MQ_j$ ].
15.             Delete the NMT from TQ and its MQ.
16.     else (( $F_1$ [ $T_i$, $MQ_j$ ] ) + $F_2$ [ $M_k$ ] > T)
17.             NMT $T_i$ is not assigned to $MQ_j$.
18.     end if
19. end for
20. j = j + 1.
21. if ( j <= MMQL )
22.         Go to step 10.
23. else if ( TQ >= 2 )
24.         Assign a NMT which takes less ET to EMT    machine
25. Update $F_2$ [$M_k$] and go to step 8.
26. else if ( TQ == 1 )
27.         Assign the NMT to a machine which takes earliest
    completion time.
28. Update $F_2$ [$M_k$]
29. else    Stop.
30. end if
31. Calculate Makespan.
```

## V. Performance Metrics

Performance metrics are the measures to compare our approach with the traditional approach. We have considered two metrics shown below

### *A. Makespan*

It measures the throughput of the heterogeneous grid environment. It is the maximum time to which tasks are finished. It is like turnaround time. It means the difference between start times of the first task to finish time of last scheduled task [5]. It is calculated using a formula shown in equation 1.

$$\text{Makespan (M)} = \max (F_2 [M_k]) \tag{1}$$

where k = 1, 2, …, n

## B. *Load Balance*

It is used to show the overloaded and under load machine. It is represented using a metric shown in equation 2.

$$\text{Load Balance} = <\text{Overloaded, Under load}> \quad (2)$$

The calculation of overloaded and under load machine is shown in equation 3.

$$\text{Load Balance} = \begin{cases} \text{Overloaded} & \text{if } \lfloor (F_2\,[M_k]) >= M \rfloor \\ \text{Under load} & \text{otherwise} \end{cases} \quad (3)$$

where k = 1, 2, …, n

## VI. EXPERIMENTAL RESULTS

### A. *Illustration*

Let us consider an example to illustrate TBS algorithm. Table I shows the execution time of tasks. MQ of tasks are shown in Table II.

TABLE II. MACHINE QUEUE OF TASKS

| Tasks | Machines | | |
|---|---|---|---|
| | ***First Best*** | ***Second Best*** | ***Third Best*** |
| ***$T_1$*** | $M_1$ | X | X |
| ***$T_2$*** | $M_1$ | $M_2$ | $M_3$ |
| ***$T_3$*** | $M_1$ | $M_2$ | $M_3$ |
| ***$T_4$*** | $M_2$ | $M_3$ | X |
| ***$T_5$*** | $M_1$ | $M_2$ | $M_3$ |

*1) First Iteration:* In Table I, there is only one EMT. It needs $M_1$. So, $T_1$ is assigned to $M_{0.}$ The threshold is initialized to 10. Then, it searches tasks to assign other machines. But, it must be less than or equal to 10. $T_2$ and $T_3$ are not assigned to best machine in first iteration first round. $T_4$ requires $M_2$ as first best machine. It is assigned to $M_2$ because the execution time is less than the threshold. No other task is assigned in the first iteration. $T_1$ and $T_4$ are deleted from TQ.

*2) Second Iteration:* In the second iteration, the threshold is updated to 15. $T_2$ is assigned to $M_1$. So, the scheduler finds tasks to assign other machines. In this iteration, $T_3$ is assigned to $M_3$. $T_3$ is deleted form TQ.

*3) Third Iteration:* As there is only one task remains in the TQ, it is assigned to the machine which takes earliest completion time i.e. $M_2$. Makespan of our approach is 17. The gantt chart is shown in Figure 7.

In Table I scenario, our proposed approach will outperform then MET, MCT, Min-Min, Max-Min, Sufferage and KPB heuristic.

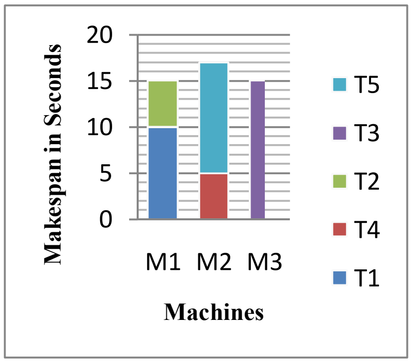

Figure 7. Gantt chart for TBS

### B. *Experimental Analysis*

Our experimental analysis shows that our proposed approach TBS is more efficient than traditional task scheduling in comparison of Makespan and load balance. The results for three cases are shown in Table III, Table IV and Table V respectively. Makespan of all heuristics with respect to the number of machines is shown in Figure 8. In some cases, the Makespan of sufferage, max-min and our approach are approximately equal. TBS may not guarantee optimum Makespan but always very close to optimum. Sometimes, it is much better than others.

TABLE III. CASE 1 RESULT

| **Algorithms** | **Makespan** | **Load balance** |
|---|---|---|
| MET | 33 | < 1, 2 > |
| MCT | 20 | < 1, 2 > |
| Min-Min | 23 | < 2, 1 > |
| Max-Min | 19 | < 1, 2 > |
| Sufferage | 22 | < 2, 1 > |
| KPB | 21 | < 2, 1 > |
| TBS | 17 | < 3, 0 > |

TABLE IV. CASE 2 RESULT

| **Algorithms** | **Makespan** | **Load balance** |
|---|---|---|
| MET | 90 | < 3, 1 > |
| MCT | 110 | < 2, 2 > |
| Min-Min | 110 | < 3, 1 > |
| Max-Min | 90 | < 3, 1 > |
| Sufferage | 123 | < 2, 2 > |
| KPB | 147 | < 1, 3 > |
| TBS | 90 | < 3, 1 > |

TABLE V. CASE 3 RESULT

| Algorithms | Makespan | Load balance |
|---|---|---|
| MET | 154 | < 2, 3 > |
| MCT | 149 | < 1, 4 > |
| Min-Min | 174 | < 1, 4 > |
| Max-Min | 96 | < 2, 3 > |
| Sufferage | 96 | < 2, 3> |
| KPB | 149 | < 2, 3> |
| TBS | 96 | < 3, 2 > |

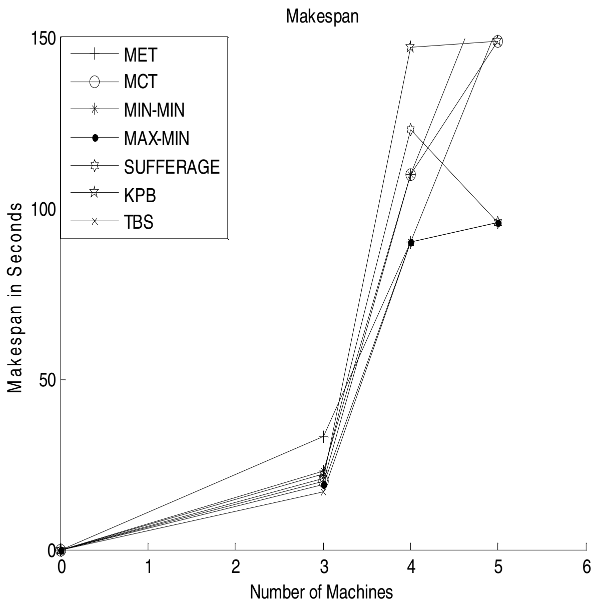

Figure 8. Comparison of Makespan vs Machines

## VII. CONCLUSION

TBS scheduling has significant improvement over traditional algorithms in grid environment. It minimizes Makespan and balance the load properly. Finally, it provides semi optimal or optimal solution.

In future work, we can extend the TBS algorithm by security, adding deadline to the tasks and priority among the tasks. Communication cost between task and resource are ignored. It can be another future work. Tasks are non-preemptive in nature. It means once a resource is assigned, it cannot be removed before execution of the task is over. Preemption is effective, if machines are very close to each other. This study illustrates with some scheduling heuristics, performance metrics, comparisons and future work.

## REFERENCES

1. Dong, F., Luo, J., Gao, L., and Ge, L. 2006. A Grid Task Scheduling Algorithm Based on QoS Priority Grouping. IEEE Proceedings of the Fifth International Conference on Grid and Cooperative Computing.
2. Etminani, K., and Naghibzadeh, M. 2007. A Min-Min Max-Min Selective Algorithm for Grid Task Scheduling. IEEE / IFIP International Conference on Internet (Sep. 2007), DOI= 10.1109/CANET.2007.4401694.
3. Hemamalini, M. 2012. Review on Grid Task Scheduling in Distributed Heterogeneous Environment. International Journal of Computer Applications 40, 2 (Feb. 2012), 24-30.
4. Ingole, A., Chavan, S., and Pawde, U. 2011. An Optimized Algorithm for Task Scheduling based on Activity based Costing in Cloud Computing. 2nd National Conference on Information and Communication Technology Proceedings published in International Journal of Computer Applications 34-37.
5. Ligang, H., Stephen, A. J., Daniel, P. S., Xinuo, C., and Graham, R. N. 2004. Dynamic Scheduling of Parallel Jobs with QoS Demands in Multiclusters and Grids. Proceedings of the Fifth IEEE / ACM International Workshop on Grid Computing.
6. Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., and Freund, R. F. 1999. Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. Journal of Parallel and Distributed Computing 59 (Jul. 1999), 107-131.
7. Mansouri, N., Dastghaibyfard, G., and Horri, A. 2011. A Novel Job Scheduling Algorithm for Improving Data Grid's Performance. IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing DOI= 10.1109/3PGCIC.2011.30 142-146.
8. Munir, E. U., Li, J., and Shi, S. 2007. QoS Sufferage Heuristic for Independent Task Scheduling in Grid. Information Technology Journal 6, 8 1166-1170.
9. Navimipour, N. J., and Khanli, L. M. 2008. The LGR Method for Task Scheduling in Computational Grid. International Conference on Advanced Computer Theory and Engineering IEEE DOI= 10.1109/ICACTE.2008.24.
10. Panda, S. K., Bhoi, S. K., and Khilar, P. M. 2012. A Semi-Interquartile Min-Min Max-Min (SIM$^2$) Approach for Grid Task Scheduling. International Conference on Advances in Computing Springer.
11. Parsa, S., and Maleki, R. E. 2009. RASA: A New Grid Task Scheduling Algorithm. International Journal of Digital Content Technology and its Applications 3, 4 (Dec. 2009), 91-99.
12. Patel, Y., and Darlington, J. 2006. A Novel Approach To Allocating QoS-constrined Workflow-based Jobs In A Multi-Cluster Grid. ACM 1-59593-581-9/06/11 (Nov. 2006).
13. Rasooli, A., Aghatabar, M. M., and Khorsandi, S. 2008. Second Asia International Conference on Modelling and Simulaion DOI= 10.1109/AMS.2008.83.
14. Senthilkumar, B., Chitra, P., and Prakash, G. 2009. Robust Task Scheduling on Heterogeneous Computing Systems using Segmented MaxR-MinCT. International Journal of Recent Trends in Engineering 1, 2 (May. 2009), 63-65.
15. Sun, W., Zhu, Y., Su, Z., Jiao, D., and Li, M. 2010. A Priority-based Task Scheduling Algorithm in Grid. 3rd International Symposium on Parallel Architectures, Algorithms and Programming IEEE DOI= 10.1109/PAAP.2010.24.
16. Xiaoshan, H., XianHe, S., and Laszewski, G. V. 2003. QoS Guided Min-Min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology 18, 4 (Mar. 2003), 442-451.
17. Panda, S. K., Khilar, P. M., 2012. A Two-Step QoS Priority for Scheduling in Grid. The Second IEEE International Conference on Parallel, Distributed and Grid Computing (Dec. 2012), 513-518.