Effective Job Scheduling in Grid Computing using Deadline Environment

R. Ananthi Lakshmi, Phd Scholar, Dept Of Cs, Kg College Of Arts And Science, Coimbatore.
Dr. R. Ravichandran, Secretary, Kg College Of Arts And Science, Coimbatore.

Abstract- The challenging issues in grid computing are to design efficient and reliable task scheduling algorithm for efficient utilization of grid computing. Grid approach provides the ability to access, utilize, and manage variety of heterogeneous resources in virtual organizations. Grid differs from normal distributed computing by the way of resource sharing and monitoring.Job scheduling is an important task of a grid computing system.In this paper we proposing a new Improved Prioritized Deadline (IPD) based scheduling algorithm for effective job scheduling with deadline constraints. Performance comparison of the algorithm has been done with the other task scheduling algorithms such as Earliest Deadline First (EDF) and Round Robin Scheduling algorithm(RRS).The proposed algorithm has more processing power of the resources while in job scheduling and shows a good results with respect to the number of job.

Keywords- Job Scheduling, Task, IPD, Deadline

I. INTRODUCTION

Grid is a collection of different nodes where in all of them contribute any combination of resources. The basic idea of Grid Computing is to create a large and powerful virtual computer which is a collection of heterogeneous distributed environment. Job Scheduling is used to choose the most suitable resource for a job to be considered.(i) The job scheduling system is responsible to select best suitable machines in a grid for user jobs.(ii) The management and scheduling system generates job schedules for each machine in the grid by taking static restrictions and dynamic parameters of jobs and machines.

In recent years, the researchers have proposed several efficient job scheduling algorithms that are used in grid computing to allocate grid resources with a special emphasis on job scheduling [4].Usually Improved Prioritized Deadline algorithm (IPD) It has considered the task deadline constraint associated with the task for its execution. Many grid users are highly interested in the timely execution of the tasks under the given deadline constraints. Most of the existing scheduling algorithms have not considered deadline perspective for task execution. To evaluate the performance of the scheduling algorithms we have used synthetic workload traces.

There are three main job scheduling[1] in a grid. Phase one is a resource discovery, which in turn generates a record involving initial resources. Level two consists of accumulate the resources as well as selecting most effective set to the application elements. During the last level the task will be executed

II. JOB SCHEDULING

Job Scheduling are types of applications responsible for the management of jobs, such as allocating resources needed for any specific job, partitioning of jobs to schedule parallel execution of tasks, data management, event correlation, and service-level management capabilities. These job scheduling[1] form a hierarchical structure, with meta-schedulers that form the root and other lower level schedulers while providing specific scheduling capabilities that form the leaves. These schedulers may be constructed with a local scheduler implementation approach for specific job execution, or another meta-scheduler or a cluster scheduler for parallel executions. The jobs submitted to Grid Computing schedulers are evaluated based on their service-level requirements, and then allocated to the resources for execution. This will involve complex workflow management and data movement activities to occur on a regular basis.

The job scheduling system is responsible to select best suitable machines in a grid[5] for user jobs. The management and scheduling system generates job schedules for each machine in the grid by taking static restrictions and dynamic parameters of jobs and machines. The various types of Scheduling Infrastructures in Grid Computing are:

- Centralized
- Hierarchical
- Decentralized

Centralized defines a Single job scheduler on one instance. Hierarchical defines two job schedulers, global and local level. Decentralized means no central instance, distributed schedulers interact and perform scheduling. Centralized Scheduling is divided into two types.

III. RELATED WORK

A dynamically schedules the tasks without requiring any prior information on the workload of incoming jobs. This approach models the grid system in the form of a diagrams such as state transition, employing a prioritized IPD algorithm with task replication[2] to optimally schedule tasks, using prediction information on resource utilization of individual nodes. Simulations, comparing the proposed

method with the round-robin and earliest deadline first have exposed the heuristic to be more effective in scheduling tasks as compared to the later. Some jobs require a large amount of data to be processed and it may not always reside on the machine running the job. The bandwidth available for such communications can often be a critical resource that can limit utilization of the grid.

It has been proved that the scheduling problem is an NP complete problem. The mapping criteria are mainly classified into two modes which are online mode and batch mode. In the online mode a task is mapped to the resource as soon as it arrives at the scheduler on the other hand in the batch mode mapping, a set of tasks is made called the meta-task. Mapping of meta-task is performed at prescheduled times called mapping event [5]. Many algorithms have been proposed by various researchers to schedule the tasks in grid environment. The selection of the algorithm for scheduling the tasks in grid environment is the most critical due to performance major of the grid.

The selection also depends on the type of the tasks, number of resources and other constraints like the deadline[4] of the task, processing speed of the processing element, bandwidth of the communication network. Based on these constraints suitable algorithm is used for scheduling the tasks. The various Performance metrics are used to evaluate the results of one algorithm with other existing algorithms like, makespan, tardiness, resource utilization, response time and many more depending upon the scenarios for which the algorithm have been designed.

Input: A set of R task and N number of
processor with computational capacity Pj

Output: A Schedule of R tasks

1. Construct a set of queues
2. q_size < R/N
3. While tasks present in queue do,
4. Assign demand rate of the task Xi
5. k=P/R
6. If Di < v
7. Assign Di to i^{th} task as fair rate
8. Else
9. Assign v to i^{th} task as fair rate
10. Calculate fair completion time t(D)
11. End while
12. End loop
13. Arrange the task in increasing order based on their t(D) and submitted to processor
14. Calculate mean waiting time each scheduled task
15. If ZXY >0
16. Every processor having small amount capacity is selected for relocation.
17. End If
18. End While.

The scheduling algorithms do not adequately address congestion, and they do not take fairness considerations into account. Fairness is needed for proper scheduling of jobs. In Job Scheduling, the jobs are allocated to multiple processors so that the tasks with unsatisfied demand get equal shares of time. The completion time of the jobs is used to determine scheduling queue of the jobs. The evaluation of completion time of the job is done by task rate using a max min fair sharing algorithm.

The job is allotted to processor in accordance with growing degree of completion time. In scheduling algorithm[3], higher order tasks are completed first which means that tasks are taken a higher priority than the others which leads to starvation that increases the completion time of tasks and load balance is not guaranteed. To overcome this we put forward Improved Prioritized Deadline algorithm (IPD) algorithm to provide the resources so that all jobs are uniformly assigned to processor depending on balanced fair rates. The main aim of this algorithm is to reduce the overall time required to complete the processing.

IV. PROPOSED JOB SCHEDULING

Improved Prioritized Deadline (IPD) based scheduling algorithm is proposed in this paper. The proposed algorithm is an improved version of the Prioritized Deadline based Scheduling Algorithm (PDSA). In the system design the resources are ranked according to their total processing power, i.e., the product of the number of the processing elements and the processing power of each element. The processing speed of each processing element in one resource is same. The allocation of the resources to the tasks[2] is based on the time delay which is the difference between the deadline of the task and the expected computation time of the task. Further, the list of resources is maintained for the allocation of the queued task[2] based on their requirements. After making the list of the resources which are suitable for the task. The selection of the resource is done on the basis of the processing speed of the resource. The highest priority is assigned to the resource with the highest processing speed for the faster execution of task.

Let us assume T_i is the i^{th} task, n is the number of tasks, a_i is the arrival time of task I, d_i is deadline of task I, ET_i is the expected execution time of task i, TD_i is the time delay of task i, TR_i is the tardiness of task I, Avg_TR is the average tardiness of the schedule and f_i is the finish time of the task i;Time Delay is referred as the difference between the deadline of the task and the expected execution time of the task as defined in Eq-1.

$$TD_i = d_i - ET_i \quad (1)$$

Tardiness refers to the time delay between the finishing time of task and the deadline of the task as defined in Eq-2.

$$TR_i = d_i - FT_i \quad \text{.....} \quad (2)$$

Total Tardiness is the sum of the tardiness of the each task which did not get executed under the provided deadline. The average Tardiness is defined in Eq-3.

$$Avg\_TR = \frac{\sum_{i=1}^{n} TR_i}{n} \quad \text{.....} \quad (3)$$

The number of non delayed tasks is the total number of tasks whose finishing time was less than the deadline of the task, i.e., which finished inside the deadline given to them. The expected completion time is calculated as the mean of the completion time for the task at every resource.

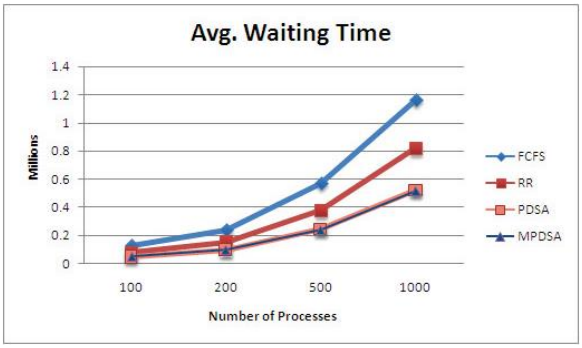
T_i, arrival time a_i, expected execution time ET_i, computational length CL_i, deadline d_i and number of processors NP_i required. Then we compute the value of the time delay TDi for each task by using Eq. (1).

The tasks in the ready queue are arranged in the ascending order based on the computed time delay (task with minimum time delay will be given priority) of the tasks. If the two tasks have a same computing delay, then the Improved Prioritized Deadline (IPD) based scheduling algorithm task will ordered on the basis of the first come first serve method in the ready queue. The tasks[2] are executed according to the suitable processors for the task.

Then from that list we select the resource having the best processing speed. The task is then executed on that resource for the time depending on its computational length and the processing speed of the resource. The finishing time f_i of the task is calculated and the tardiness TR_i is calculated using Eq. (2). If there is no tardiness for that task then the number of non-delayed tasks is incremented. When all the tasks in the ready queue are finished then we calculate the average tardiness *Avg_TR* using Eq. (3). The flowchart of the proposed algorithm.

The proposed scheduling algorithm is compared with Prioritized Deadline Based Scheduling Algorithm (PDSA) and Earliest Deadline First (EDF). The PDSA algorithm is used in [12] is for the task that requires a single processing element for execution. Here we are further extending it and using it for the tasks requiring more than one processing element in heterogeneous environments. In addition to this, the proposed algorithm considers the processing speed of the

V. RESULT AND DISCUSSION



(a).Average Waiting Time

Improved Prioritized Deadline (IPD) based scheduling algorithm executes the job with the closest deadline time delay in the cyclic manner using a dynamic time quantum. Based on our algorithm perform the allocation for a single processor based on the deadline[2] criteria dependent on the minimum time delay of job execution, turnaround time, waiting time and maximum tardiness.

The performance metrics for the scheduling algorithms is based on the average tardiness and the percentage of non-delayed tasks. In the deadline based system, our main emphasis is to make as much as tasks to be completed inside their deadline. So these two performance metrics give us the clear idea of the performance algorithms for these types of systems where the deadline of the task is the main constraint. The comparison of the proposed algorithm with other algorithms is described below using the performance metrics.

Table 1: Average Tardiness for Scheduling Algorithms (in Seconds)

NO. OF TASKS	EDF	PDSA	IPD
1000	783.22	783.22	106.59
2000	612.34	496.32	83.23
3000	751.35	458.37	110.52

VI. CONCLUSION

In this paper, a scheduling algorithm for executing jobs on grid systems is proposed. Just like real-life scenarios, we have considered the dynamic arrival of jobs as well as the deadline requirement of each job to be processed. The experiment has been performed by varying workload by increasing jobs from 100 to 1000 in a scalable manner. The result has shown maintained performance under dynamic environment. Based on the comparative performance analysis IPD has shown the best performance as compared to RRS, FCFS scheduling algorithm under variable and scalable workload.We have developed a new simulator using Java language to facilitate this research.Various possible input patterns were experimented with all the CPU scheduling algorithms. We can say that IPD is a scheduling policy from the system point of view, it satisfies the system requirements (i.e. short Average Waiting Time and short Turnaround Time) and also supports scalability under heavy workload. In the future, we will evaluate and propose a computational scheduling algorithm on the grid based on multiple processors and perform detailed comparative performance analysis with other scheduling approaches.

VII. REFERENCES

[1]. Menglan Hu and Bharadwaj Veeravalli, "Requirement-Aware Scheduling of Bag-of-Tasks Applications onGrids with Dynamic Resilience", IEEE Transactions onComputer, vol. 62, no. 10, pp. 451-459, 2013.

[2]. Jing Wang, Gongqing Wu, Bin Zhang, Xuegang Hu, "A heuristic algorithm for scheduling on grid computing environment", Seventh ChinaGrid Annual Conference 2012, pp. 36-42.

[3]. Yun-Han Lee, Seiven Leu, Ruay-Shiung Chang, "Improving task scheduling algorithms in a grid environment", Future Generation Computer Systems , vol. 27, pp. 991-998, 2011.

[4]. Menglan Hu and Bharadwaj Veeravalli, "Requirement-Aware Scheduling of Bag-of-Tasks Applications onGrids with Dynamic Resilience", IEEE Transactions onComputer, vol. 62, no. 10, pp. 451-459, 2013.

[5]. Manoj Kumar Mishra, Raksha Sharma, Vishnu KantSoni, Bivasa Ranjan Parida, Ranjan Kumar Das, "AMemory-Aware Dynamic Task Scheduling Model in Grid Computing", International Conference OnComputer Design And Applications, vol. 1, 2010, pp. 545-549.