Runtime Estimation Aware Scheduling Algorithm for Handling Deadline based Jobs in Grid Environment

Rajkumar Rajavel[1], Mala T[2]

[1] Research Scholar, Department of IST, Anna University, Chennai, Tamil Nadu, India
[2] Assistant Professor, Department of IST, Anna University, Chennai, Tamil Nadu, India
[1] rajkumarprt@gmail.com
[2] malanehru@annauniv.edu

Abstract. One of the major issues in Grid Environment is scheduling the deadline based jobs in the meta-scheduler level. In many approaches Bulk Scheduling is done in meta-scheduler by evenly distributing all the jobs present in the request handler queue to the available resources. All the submitted jobs will fall into the resource queue and get executed in the First Come First Served (FCFS) or Shortest Job First (SJF) fashion and it will works better for non-deadline based jobs where prioritization of job is not required. This type of scheduling algorithm will leads to starvation and increases the waiting time of deadline based jobs, and finally results in user dissatisfaction. Moreover, these types of scheduling algorithm in the meta-scheduler will not provide the feasible solution to the deadline based job. So a Runtime Estimation Aware Scheduling is proposed in the meta-scheduler, which provides high priority for deadline based jobs to dispatch first, during the scheduling process. Most algorithm will works good if the job have the runtime input and may not good enough if the job doesn't have the runtime input. Our scheduling algorithm will works well with irrespective of jobs nature (either job with runtime or without runtime), obviously reduces the waiting time of deadline based (high priority) jobs and lights up the grid providers to achieve Service Level Agreement (SLA).

Keywords: Meta-Scheduler, Deadline based jobs, Bulk Scheduling, Runtime Estimation Aware Scheduling, Waiting time and SLA.

1 Introduction

Grid computing is the dynamic and coordinated resource sharing to solve the problem by dynamically linking the various resources of the Virtual Organizations. Here the Virtual Organization represents the group or an individual who engaged in designing rules for the sharing of resource. According to Ian Foster and Kesselman, Grid Computing is hardware and software infrastructure which offers a cheap, distributable, coordinated and reliable access to powerful computational capabilities [4]. Since multiple grid or any applications may require numerous resources which is often not available for them so that in order to allocate resource to input jobs, having a scheduling system in the meta-scheduler is essential. Because of vastness and separation of resource in the computational grid, meta-scheduling is seems to be most

important issues. In the grid environment Bulk Scheduling is mostly often preferred in the Meta-Scheduler level for scheduling the tasks or jobs. Since more number of jobs is arrived to meta-scheduler it cannot be scheduled one after another. So as soon as the job is arrived to the meta-scheduler it should be distributed to the underlying resources as early as possible [10], [2], [13]. This resource will adopt various scheduling algorithms for the execution of the jobs. As per the grid environment is concern each and every individual personal computer and cluster computer is considered as a resource. Cluster computer consist of one head node and any number of computing nodes. Usually job is submitted to the head node of the cluster and from the head node job is distributed over the computing nodes for running the jobs. Here the main role of the head node is to distribute and maintain the job running in the computing nodes. In the cluster nodes, if any problem occurs in computing nodes head node will take the decision to migrate the job to other computing node by using the fault tolerance technique [9], [11], [3].

In the Grid Environment for scheduling the jobs in the meta-scheduler, there might be lots of resource available for the execution of job and hence have an issue of choose the best resource for deadline based jobs. To sort out this issue it is mandatory to know about the nature of grid environment whether the meta-scheduler maintains the jobs history or not. Because, some scheduling strategy like FCFS, JR-backfilling and SLOW-coordinated will works well in the environment were the runtime of the job is known prior to scheduling, either the runtime of job is indicated in job input or the job runtime is estimated from the job history [5]. Similarly the Application Demand Aware algorithm and DIANA Scheduling algorithm works well only in the heuristic environment in which jobs runtime is known for exploiting in estimation of jobs completion time [6], [1]. So novel scheduling algorithms is proposed in the meta-scheduler to adapt in different nature of grid environment. Here the Runtime Estimation Aware Scheduling algorithm is proposed to improve the efficiency of the scheduling strategy in the grid environment where the history of jobs is maintained. Hence the overall performance of the meta-scheduler is improved by the proposed algorithms by reducing the waiting time of the jobs and moreover helps in achieving the Service Level Agreement (SLA) made in the Grid Environment.

2 Related Work

The important issue in grid environment is scheduling deadline based jobs which in turn leads to performance degradation and users dissatisfaction due to improper scheduling of the jobs [1]. In the previous paper, Dynamic Load Balancer algorithm proposal is done for estimating the resource load by using little's formula which will evenly distribute the jobs from the meta-scheduler to the underlying resource. It works well for handling non-deadline based jobs [12]. One more important issue in the scheduling is guaranteed to complete the execution of the jobs within the deadline given by the user for achieving the SLA made between the user and the provider. There are so many scheduling algorithms exist in the meta-scheduler level as shown in paper [13], [8], [6] which provides feasible solution for the non-deadline based jobs and it is not guaranteed for the completion of jobs within the stipulated time. In paper

[7] List Scheduling (LS) algorithm is proposed which gives better in the situation where the meta-scheduler have limited jobs and that too with non deadline based jobs. Main reason for this problem is there may be the possibility to have too many number of jobs waiting in the meta-scheduler where cannot be able to prioritize the deadline based jobs and finally leads to increase in waiting time of job. In paper [3] Proposed Scheduling Algorithm (PSA) is used which will works efficiently in the case were the runtime of the job is known in advance to the user. This algorithm will not work well in the situation where the runtime of the job is not specified in the user's job specification. So in-order to handle this situation runtime estimator is used in meta-scheduler level to estimate the runtime of the job which is not having the execution time or runtime in the job specification.

3 Need of Runtime Estimation Aware Scheduling

The Runtime Estimation Aware Scheduling will helps in making intelligent decision regarding the selection of best resource to meet the user requirement, by estimating completion time of the job using waiting time of job in queue and runtime or execution time of jobs in each resource. This scheduling algorithm in turn exploits the Runtime Estimator component for estimating the runtime of jobs whose runtime is not mentioned in the job specification. These estimations are used by the scheduling algorithm for predicting the completion time of deadline based jobs to choosing the best resource for job submission. By exploiting this Runtime Estimation Aware Scheduling in the meta-scheduler can make the decision making process as good as possible. This runtime estimator is not an actual component of the meta-scheduler, but it is added in our proposed model to improve the efficiency of the meta-scheduler. This scheduling approach works well for the environment were the job has runtime specification and as well as the meta-scheduler were the job history is maintained.

4 Meta-Scheduler in the Grid Environment

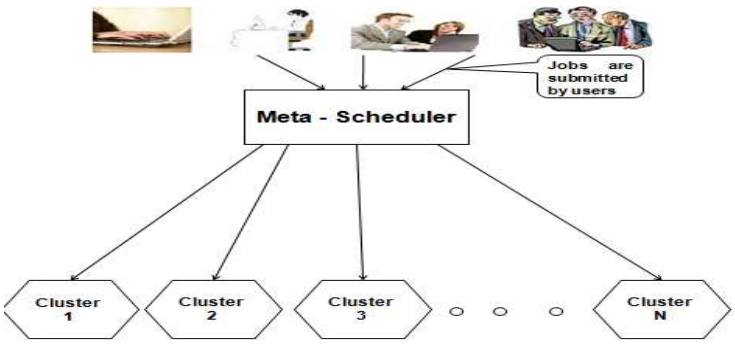

Fig. 1. Meta-Scheduling Model

In this paper, Runtime Estimation Aware Scheduling and Multilevel Feedback Queue Scheduling algorithm is proposed in the Meta-scheduler Model as shown in Fig. 1. The structure of the cluster consists of one Head Node and 'N' number of computing Nodes as is shown in the Fig. 2. The users will submit their jobs described using Job Submission Description Language (JSDL) specification to the underlying meta-scheduler which in turn falls into the queue of Request Handler component. This queue is called as the request handler queue or ready queue. The structure of meta-scheduler model consist of various component such as Request Handler (RH), Run-time Estimator (RE), Database, Dispatch Manager (DM), Transfer Manager (TM), Execution Manager (EM) and Information manager (IM) as shown in Fig. 3. Here the RH provides the user interface and it maintains the ready queue for obtaining the jobs submitted by different user.

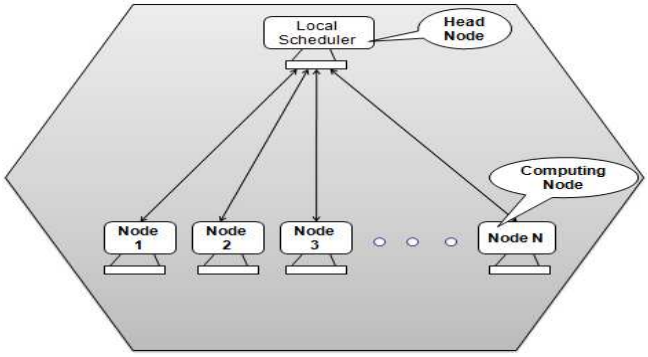

Fig. 2. Structure of the Cluster

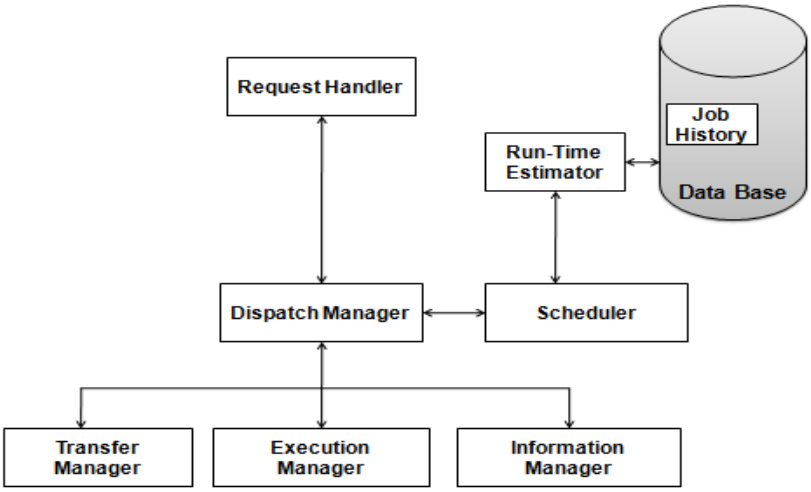

Fig. 3. Structure of Meta-Scheduler

The DM component maintains the time interval for pulling the jobs from the ready queue. For each time interval DM will periodically pulls the job from RH and give it to Scheduler component for appropriate scheduling. Now this scheduler component will exploit the proposed scheduling algorithm to map with the best possible resources available as shown in Algorithm 1. First the algorithm checks for the runtime of the job in the job specification. If it finds the runtime attribute in the job specification, then it will probably finds the appropriate matched resource for job submission and as a result it will migrates the job to the concern resource for execution. In-case if the scheduling algorithm doesn't find any runtime attribute in the job specification, then it will automatically calls the RE component to find the estimated runtime of the job from the Job History. This situation arises since many users submit their jobs with the incomplete job requirement (without specifying the runtime). These types of jobs with incomplete job requirement or with inaccurate runtime will leads to poor scheduling. Most of the cases user may not be aware of the jobs nature, so runtime may not be specified in the job requirement. To handle such a jobs, runtime or execution time is estimated by using the RE component which in turn query the database with some attributes of the jobs to get the runtime information of job from the Job History. The accurate runtime estimation helps to achieve better resource utilization, reduce waiting time, proper resource allocation and satisfying user level Quality of Service.

Finally the scheduler component will finds the matched resource-id and then invoke the DM for dispatching the job to matched resource. The IM will query the Monitoring and Discovery Service (MDS) and sends the resource or host information to the Scheduler. Based on the monitoring interval it keeps track of the host status and updates the host information such as if any new resources are added or removed in the Grid Environment that information are updated periodically. The TM is invoked by the DM with the job-id and the matching resource-id as input. Once it is invoked, the TM creates a remote directory for the given path name as one specified in user specification or jobs input. TM gives the permission rights for the execution of given job in the remote directory. Once this process is over, it informs the DM through messages. The EM is invoked by the DM when the TM completed the creation of directory in the remote host. The DM will dispatch the job for execution and the EM will keeps on monitoring and updating the job status to the scheduler. Finally EM reports the scheduler about the completion of job in case of successful execution and reports the failure in case of job incompletion.

5 Runtime Estimation Aware Scheduling Algorithm

Exploiting of this algorithm in the meta-scheduler will helps to predict the completion time of deadline based jobs in the available resource. The Completion Time of Job 'J_i' in the available 'k' number of resource is computed as shown in equation (1) and (2),

$$CT_{Ji}(R_k) = WT_{Ji}(R_k) + ET_{Ji}(R_k) \quad (\mathbf{1})$$

Where $WT_{Ji}(R_k)$ denotes the Waiting Time of Job 'J_i' in the resource R_k and $ET_{Ji}(R_k)$ represents the Execution Time of the Job 'J_i' in the resource R_k. Here the Waiting Time of the job in particular resource is computed as follows,

$$WT_{Ji}(R_k) = \sum [ET_{WJ1}(RQ_k), ET_{WJ2}(RQ_k), \ldots, ET_{WJm}(RQ_k)] \quad (2)$$

Where $ET_{WJ1}(RQ_k)$ denotes the Execution Time of Waiting Job 'WJ_1' in the Resource Queue RQ_k and the 'm' indicates the number of jobs waiting in Resource Queue of the resource R_k.

Algorithm 1. Runtime Estimation Aware Scheduling (REAS) Algorithm

```
Begin
Get information about list of jobs waiting in resource queue
for ( all job )
        if ( job requirement has the execution time )
            Compute Completion time of job in all resource
            Identify the resource having minimum Completion time
            Submit the job to selected resource
        else
            Identify the similar types of job from the job history
             Get the execution time of similar jobs present in the Database
            Add the execution time to the job requirement
            Compute Completion time of job in all resource
            Identify the resource having minimum Completion time
            Submit the job to selected resource
End
```

6 Experimental Results and Performance Evaluation

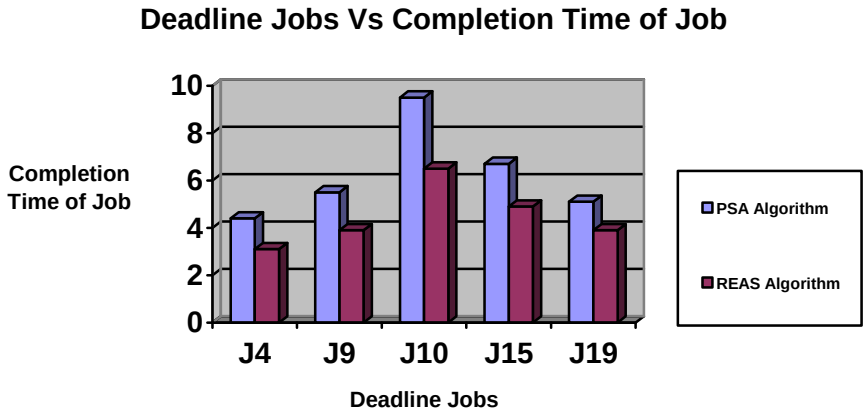

Fig. 5. Performance Evaluation of PSA Algorithm Vs REAS Algorithm

In the result phase we have evaluated the result by exploiting five resources with hundreds of dissimilar jobs (deadline and non deadline based). The result of the Proposed Scheduling Algorithm (PSA) in paper [3] is compared with the proposed REAS algorithm and its performance measure is also represented as a graph as shown in Fig. 5. In-order to show exact performance of the proposed algorithm, the graph is generated for deadline based jobs. In the job submission experimental model there is only five deadline based job such as J4, J9, J10, J15 and J19. From the graph it is clear that the performance of the proposed REAS algorithm works much better than the Proposed Scheduling Algorithm (PSA) in paper [3] with respect to completion time of deadline based jobs.

7 Conclusion and Future Work

In this paper a complete role of meta-scheduler using Runtime Estimation Aware Scheduling algorithm is evaluated with the simulation and traces from real time grid environment as well as using GridSim toolkit. The result obtained with performance evaluation can effectively schedule the job to the underlying resource by giving more priority to deadline based job (interactive job). Hence from the result it indicates that completion time of the priority jobs can be considerably optimized by using Runtime Estimation Aware Scheduling algorithm in the Meta-scheduler. In the future we will work towards the exploitation of Service Level Agreement in the meta-scheduler to provide the Quality of Service in the real Grid Environment.

References

1. Ashiq Anjum, Richard Mc Clatchey, Arshad Ali and Ian Willers, "Bulk Scheduling with the DIANA Scheduler", IEEE Transactions on Nuclear Science, Vol. 53, No. 6, December 2006.
2. Aysan Rasooli, Mohammad Mirza-Aghatabar and Siavash Khorsandi, "Introduction of Novel Rule Based Algorithm for Scheduling in Grid Computing Systems", Second Asia International Conference on Modeling and Simulation, 2008.
3. Hojjat Baghban and Amir Masoud Rahmani, "A Heuristic on Job Scheduling in Grid Computing Environment", Seventh International Conference on Grid and Cooperative Computing, 2009.
4. I. Foster and C. Kesselman, "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann, 1999.
5. Ivan Roder Francesc Guim and Julita Corbalan, "Evaluation of Coordinated Grid Scheduling Strategies", 11th IEEE International Conference on High Performance Computing and Communications, 2009.
6. Jie Lin, Bin Gong, Hui Liu, Chaoying Yang and Yuhui Tian, "An Application Demand aware Scheduling Algorithm in Heterogeneous Environment", 11th International Conference on Computer Supported Cooperative work in Design, 2007.
7. Keqin Li, "Job Scheduling for Grid Computing on Metacomputers", 19th IEEE International Parallel and Distributed Processing Symposium, 2005.
8. Keqin Li, "Job Scheduling for Grid Computing on Metacomputers", 19th IEEE International Parallel and Distributed Processing Symposium, 2005.

9. Kun-Ming Yu, Chen-Kwan Chen, “An Evolution-based Dynamic Scheduling Algorithm in Grid Computing Environment”, Eighth International Conference on Intelligent Systems Design and Applications, 2008.
10. Lina Ni, Jinquan Zhang, Chungang Yan, Changjun Jiang, “A Heuristic algorithm for Task Scheduling based on Mean Load”, First International Conference on Semantics, Knowledge and Grid, 2006.
11. Marcelo Massocco Cendron and Carlos Becker Westphall, “A price-based Scheduling for Grid Computing”, Seventh International Conference on Networking, 2008.
12. Rajkumar Rajavel, Thamarai Selvi and Kannan Govindarajan, “Dynamic Load Balancer Algorithm for the Computational Grid Environment”, Communication in Computer and Information Science, 1, Volume 101, Information and Communication Technologies, Part 1, Pages 223-227, LNCS Springer, 2010.
13. Yiqun Zhu, Minglu Li and Chuliang Weng, “Ant algorithm with Execution Quality Based Prediction in Grid Scheduling”, Fourth China Grid Annual Conference, 2009.