Framework for Job Scheduling in Grid Environment

Deepti Malhotra
Research Scholar
Department of Computer Science & IT
Jammu University, India

Devanand
HOD
Department of Computer Science & IT
Jammu University, India

ABSTRACT
Job scheduling is a fundamental issue in achieving a high performance on the Grids. In grid computing several applications require numerous resources for execution which are not often available for them, thus presence of a scheduling system to allocate resources to input jobs is vital. This paper introduces a model and a job scheduling algorithm in grid computing environments. Computational grids have the potential for solving Large-scale scientific problems using heterogeneous and geographically distributed resources. One problem that is critical to effective utilization of computational grids is the efficient scheduling of jobs. This work addresses this problem by describing and evaluating a grid scheduling architecture and a job-scheduling algorithm. The research work introduces NSA (node-selection algorithm) at the global scheduler and the PSA (processor selection algorithm) at the local scheduler. The NSA is based on the rule that the light-loaded processing node is selected for the job allocation. This technique fetches the jobs from the Global job queue that is ready to execute and assign these jobs to the best nodes of the grid. The PSA (processor selection algorithm) schedule the job to the processor of a selected node having maximum available CPU resource (ACR).The algorithm has been tested in a simulated grid environment.

Keyword
Grid Computing, Job Scheduling, Scheduler, load, ACR.

1. INTRODUCTION
Grid computing, adapted from Ian Foster [1], is concerned with "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations." A Grid is a decentralized heterogeneous system in which resources belong to multiple organizations. Grid does not enforce absolute control over these resources and resource management is subject to multiple and divergent organizational administrative policies. From user's perspective, a Grid is a collaborative problem-solving environment in which one or more user jobs can be submitted without knowing where the resources are or even who own the resources. A Grid must guarantee the quality of service of a job's execution. Since multiple applications may require numerous resources which often are not available for them so that in order to allocate resources to input jobs, having a scheduling system is essential. Because of the vastness and separation of resources in the computational grid, scheduling is one of the most important issues in grid environment [2]. *Grid scheduling is the process of scheduling jobs over grid resources.*

A scheduler is a component of the resource management system on a grid [3]. A scheduler system provides the interface between a user and the grid resources. A grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and usually owns the resource. A grid scheduler is in charge of resource discovery, grid scheduling (resource allocation and job scheduling) and job execution management over multiple administrative domains. Scheduling of jobs on a grid or a cluster is the task of mapping jobs to the available compute-nodes. The Complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively [4]. On a grid, scheduling is a NP-complete problem [5].According to Wright [6], a scheduler is designed to satisfy one or more of the following common objectives :(a) Maximizing system throughput. (b) Maximizing resource utilizations. (c) Maximizing economic gains (d) Minimizing the turn-around time for an application. Job scheduling is one of the most fundamental and important aspects of distributed and parallel computational systems. Vast investigations have been done in this scope, which have led to theories and practical results [7, 8, 9]. However new scheduling algorithms have been offered with emergence of grid computing. Objectives of scheduling algorithm are increasing system throughput [9], efficiency, and decreasing job completion time.

The rest of the paper is organized as follows. Section 2 presents the background and related work. Section 3 provides the system model for the proposed scheme. Algorithms related to the proposed scheme is also given in the figure3 in this section.Section3 also discusses the relationship between different entities in the form of a sequential diagram given in figure4.Section4 presents the experimental setup and results of experiment carried out in simulated grid environment. Finally Section 5 concludes the paper by summarizing our contributions and future works.

2. RELATED WORK
The different phases of Grid Scheduling Process have been discussed in [10]. In a grid, the abilities of the computing nodes vary, and tasks often arrive dynamically. Because of these, scheduling methods of parallel computing [11] may not be applicable in a grid. It is important to properly assign and schedule tasks to computing nodes [12]. Through good scheduling methods, the system can get better performance, as well as applications can avoid unnecessary delays. Grid scheduling algorithms in different occasions are discussed in paper [13], [14].

Various algorithms have been proposed which in recent years each one has particular features and capabilities. In this section we review several scheduling algorithms which have been proposed in grid environment. In [15] a scheduling algorithm which is based on HQ-GTSM is presented. This algorithm not only takes into account the input jobs but also considers the resource migration time in deciding on the scheduling. One of the most important features of this algorithm is that it guarantees the grid quality of service.

DIANA [16] and BLBD [17] are scheduling algorithms that have been proposed in grid computing environment too. Based on these methods, scheduler chooses the best resource for job allocation by considering the system load and cost of resource allocation. It is interesting to know that BLBD is an adaptive scheduling which is more focused on the guarantee of quality of service. Weighted meta-scheduling is presented in [18]. This algorithm takes into account both the system load and workload in order to enhancing the resource allocation.

Many of the Grid systems such as 2K [19], Darwin [20], and Legion [21] utilize a hierarchical scheduler. For simplicity of exposition, a two-level hierarchy job scheduling architecture of Grids is considered. At the first level, global schedulers schedule jobs to a site (e.g., a server or cluster). At the second level, the local scheduler of each site dispatches the incoming jobs to the service resources (e.g., processors). Both global schedulers and local schedulers can use various job scheduling algorithms.

3. PROPOSED GRID SCHEDULING ARCHITECTURE

A grid scheduling architecture is described in Fig 1. The main components used in this architecture are Job Collector, Resource Discovery Unit, dispatcher and Job Scheduler. The clients are users who are in need of resources. User applications are fed into the Grid Scheduling System. Resources provide the services requested by the user. Each resource may differ from the rest of the resources with respect to number of processors and local load factor.

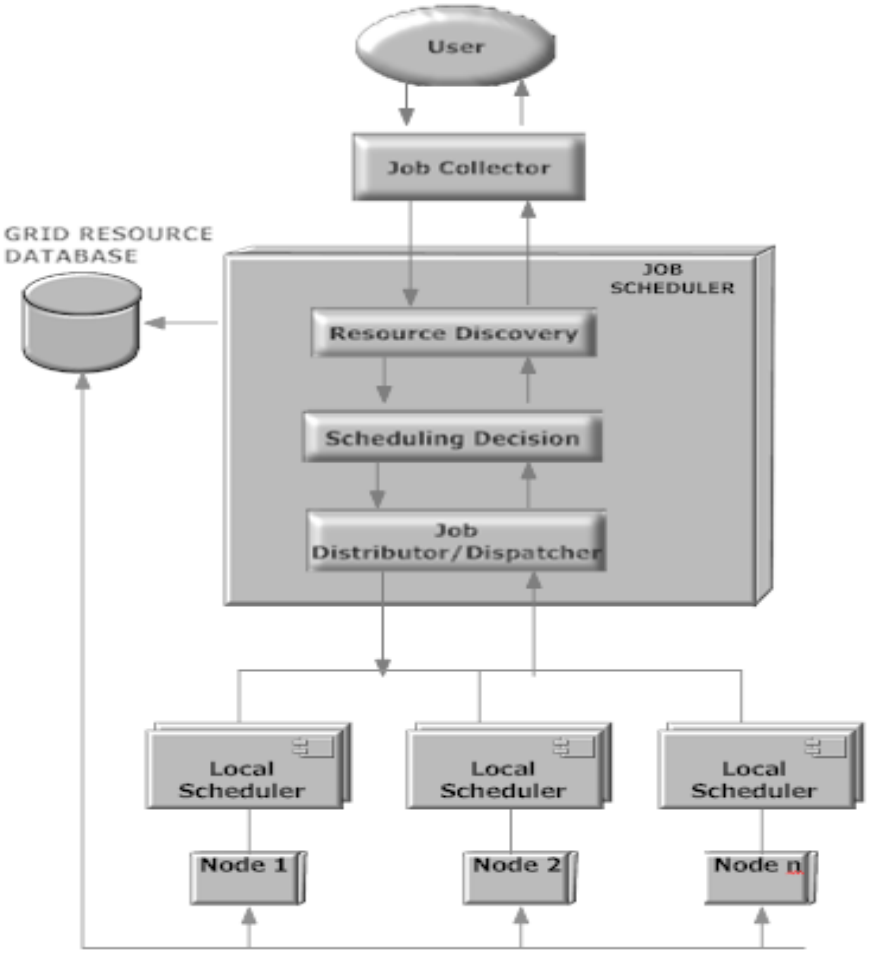

Fig1: The Proposed Grid Scheduling Architecture

- ✓ The system is composed of a number of independent non-dedicated sites with several heterogeneous computational resources of various organizations. As the sites are non-dedicated, no one has full control on all the available resources and applications (jobs). Each site has local users that submit jobs to its own local job scheduler and the local job scheduler is responsible for managing local jobs only.
- ✓ The independent users submitting their jobs in the grid environment need to register before submitting jobs to the Job Collector. The authentication of the registered users will be verified.
- ✓ The Job collector maintains the resource requirement of the jobs and the user's authentication. The job collector submits the collected jobs with their corresponding resource requirements to the Job scheduler.
- ✓ The GRD(Grid Resource Database) keeps the information of the grid resources such as IP Address, CPU Speed , CPU Frequency, Number of CPU in the resources as well as it also keep track of the ACR(Available CPU Resources) of all the computers on the Grid.

$$ACR = (Fequency\ Of\ CPU * CPU\ Idle)/100$$

- ✓ After the job being inserted, it will enter the "resource discovery" unit. The duty of this unit is to identify host in Grid system and retrieves the information of the grid resources such as IP Address, CPU Speed , CPU Frequency ,Number of CPU in the resource and ACR of all the computers on the Grid from the GRD.Therefore, the "resource discovery" unit is responsible for delivering the received information to the "scheduling decision" unit in order for the input jobs to be allocate to the most appropriate host for the execution of the input jobs, based on the policy of the proposed scheduling algorithm given in Fig3.
- ✓ After making decision and choosing the proper resource by "scheduling decision" unit "Dispatcher" or "Job Distributor" unit sends the job to the desired resource.
- ✓ A grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and usually owns the resource. A grid scheduler is in charge of resource discovery, grid scheduling (resource allocation and job scheduling) and job execution management over multiple administrative domains.

3.1 The Proposed Scheduling Model

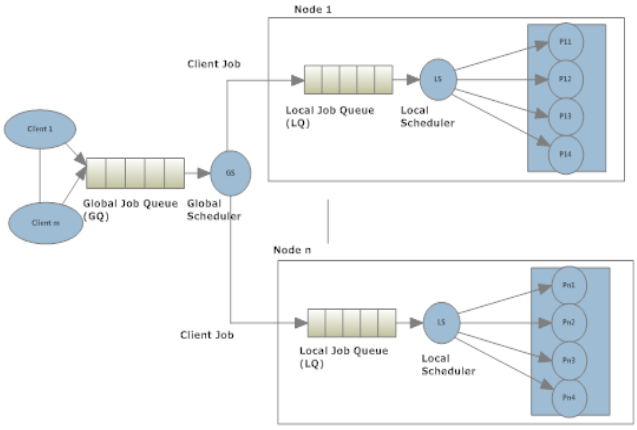

Fig2: Job Scheduling Model

1) Client: Each client submits job to scheduler. Then each job can be scheduled to any of the nodes and further dispatched to any of the processors of that node.

2) Node: Node is a set of cumulative resources (CPUs, memory,

storage space, list of specializations) with limited capacities. We have assumed that the capacity (load, ACR) of the machine can be a general value for the machine. Each Grid node comprises a number of computational resources (processors, denoted by P_{ni} $\{\sum_{n=1}^{x} n = node\ no. \mid \sum_{i=1}^{4} i = processors\ no.\ in\ node\}$) in Figure2. The resources on the grid are usually accessed via an executing "job". A resource is a basic device where jobs are scheduled/processed/assigned [22]. Each resource has a limited capacity (e.g., number of CPUs, amount of memory). Each resource also has a speed and a load. In our model we have assumed that each node consist of 4

- **Local scheduler**. When a job is delivered to a Grid node, it is managed by the local scheduler of that node. The LS determines how to dispatch the job to one processor according to PSA (processor selection algorithm) at the local scheduler). It only uses the local information, such as the ACR (Available CPU Resources) of each local processor for its decisions.

processors. The specification of each processor is Intel(R) Core(TM)2 Duo CPU with different computing power.

3) Client Job: A job (task, activity) is a basic entity which is scheduled over the resources. A job has specific requirements on the amounts and types of resources (including machines). In our model Job j has the following variables (properties):

- A ***release date*** (r_j) is the time when a job is available to start processing including i/o operations).
- The ***start time*** (S_j) is a time when the job actually begins its processing$(S_j \geq r_j)$.
- The ***completion time*** (C_j) is the time when a job completes its processing.

It is important to mention that r_j, C_j and S_j are often the wall-clock times.

4) Queue: A *queue* is an abstract structure that accepts, schedules, and/or sends jobs that have been submitted by users to resources that belong to the queue. The queue often has a set of jobs it has accepted that are waiting for scheduling. The length of the queue indicates the load of resources that belong to the queue or/and the number of jobs waiting for processing or scheduling in this queue. Here we assume that at one time only five jobs can reside in a queue.

Now consider two basic types of queues that are included in the proposed model shown in figure 2.

- **A *global job queue*** (GQ) schedules jobs to the selected node based on the proposed scheduling algorithm given in figure 3 and sends the jobs to a local job queue (LQ) belonging to the selected node. The parameters of the GQ indicate which local queues (LQs) may be used for scheduling.
- **A *local job queue*** (LQ) schedules jobs to resources that belong to it based on the proposed scheduling algorithm given in figure 3 and distributes the jobs to smaller set of resources(Processors of the selected node) than those available to the GQ.

A queue schedules jobs on the resources that belong to it. In the proposed model, we have limited the number of clients (client jobs) that can be in GQ and LQ belonging to the GQ to five. The LQ_n schedules jobs on $LQ_n.R = P_{n1} \cup P_{n2} \cup P_{n3} \cup P_{n4}$ where $P_{ni} = i^{th}$ machine/processor that belongs to the Queue Q_n. The GQ schedules jobs on the resources of all LQ_n that belong to the GQ, $GQ.R = \cup_n LQ_n.R$, where $\sum_{n=1}^{x} n =$ node no.

5) Scheduler: It is the core of the system and can be classified into two types: global scheduler (GS) and local scheduler (LS).

- **Global scheduler**. In the system, a client submits its jobs to the global GS. Once a GS receives a job, it immediately makes a decision on which Grid node the job should be assigned to, according to NSA (node selection algorithm). It may use the global information, such as the load of each Grid node, as input to its decisions. Each client is associated with a global scheduler, while each global scheduler is associated with all of the Grid nodes to assign jobs and acquire global information.

3.2 Proposed Job Scheduling Algorithm Description

In this section, we describe the routing strategies involved in the Grid model, which is integrated by the node-selection algorithm (NSA) of the global scheduler (GS), processor selection algorithm (PSA) at the local scheduler (LS).

In step 1 of Algorithm given in figure3, the user's request is processed and split into individual job requests. Step 2, consists of list of all nodes available in the grid. The actual scheduling process starts from step 3 that is repeated for each job request. In step 4, the scheduler discovers the available resources by contacting GRD (GridResource Database) .Here resources are evaluated according to the requirements in the job request and only the appropriate resources are kept for further processing. Step 5 calculate the load for each combination of processor *i* and node *n* .Step6 calculate the load of each node,which is the summation of loads of all the four processor exist in the node. Step 7, 8 & 9 predicts the performance of each resource by estimating the minimum load of each node and then maps the job with the resource/node having minimum load. Step10 calculate the ACR of each processor of selected Node n by using the frequency and idle time of each processor. In Step13 the job is assigned to the processor having maximum value of ACR.

3.2.1 Algorithm

Step 1: cList= list of all individual requests by validating the client specification(s);
Step2: nodeList =list of all nodes. Each Grid node comprises a number of computational resources (processors, denoted by
P_{ni} $\{\sum_{n=1}^{x}; n = \text{node no.} \mid \sum_{i=1}^{4} i = \text{processors no. in node}\}$);
Step 3: For each job do the following steps;
Step 4: Filter out the resources that do not fulfill the job requirements. Contact GRD (GridResource Database) to obtain a list of available resources;
/ NSA (node-selection algorithm)*/*
Step 5: Calculate Load for each combination of processor *i* and node *n*.
(P_{Lni}) = Load of i[th] processor no. of n[th] node. ($\forall\, n \leq x : i \leq 4$);
*/*calculate the load of each node which is summation of load of all the processors of a respective node*/*
Step6: $For\ n = 1\ to\ x$

$$N_{Ln=} P_{Ln1} + P_{Ln2} + P_{Ln3} + P_{Ln4};$$

*/*Declare the array named nLoad for storing the loads of all nodes*/*
Step7: $int\ ndLoad[200];$
Step 8: $minLOAD\ (N_{Ln_{min}}) = ndLoad[1];$
Step9: $For\ n = 1\ to\ x$ /*ndload[200]
{
$If(minLOAD\ (N_{Ln_{min}}) > ndLoad[n])$
{
$minLOAD\ (N_{Ln_{min}}) = ndLoad[n];$
$N_{Ln_{min}} = n;$
}/*End If */
} /*End For*/
Step10: $For\ i = 1\ to\ 4$ */* PSA (Processor selection algorithm) */*
{
*/*Calculate the idle time of each processor of selected Node n */*
$P_{MaxLn\,i} = 100;$
$CPU_{idle\,[i]} = P_{MaxLn\,i} - P_{Lni};$
Find frequency for each processor (P_{Fni}) ;
/*Calculate ACR for each processor */
$ACR[i] = \frac{(P_{Fni} * CPU_{idle\,[i]})}{100};$
Step11: } /*End for */
Step12: $maxACR(P_{ni_{max}}) = ACR[1];$
Step13: $For\ i = 1\ to\ 4$
{
$If\ maxACR(P_{ni_{max}}) < ACR[i]$
{
$maxACR(P_{ni_{max}}) = ACR[i];$
$P_{ni_{max}} = i;$
}/*End If */
}/*End For*/

Fig3: Scheduling Algorithm

3.3 Scheduling Interactions

Figure4 presents a UML sequence diagram depicting the interactions are between Job Collector, Scheduler, Grid Resource, Grid Resource database and Grid statistics. Initially the Grid resources register themselves with the GRD and Grid users register themselves with the Job Collector. A user submits a job request to the Job collector then job is transferred to the scheduler for computation by the Job collector. The scheduler then validates the request and interacts with GRD for requesting resource information. Then it estimates the choose the best resource as described in Algorithm given in fig3. Finally Submit Job request is sent to the selected resource only. When the job is finished, the resource entity sends back a completion event to scheduler. The report writer entity creates a report for entire simulation by getting the statistics from the Grid Statistics entity.

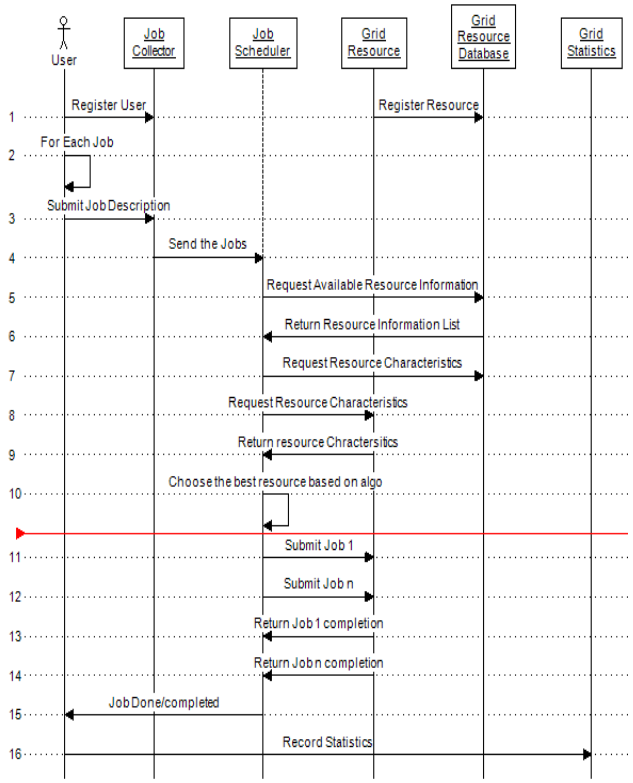

Fig4: Sequence Diagram for Grid Scheduling Process

4. EXPERIMENTAL SETUP

We carried out a simulation-based study. We created our own Grid simulation environment in Department of Computer Science & IT in University of Jammu .The simulated grid environment is carried out by using Turbo C. We simulate the grid computing environment by using, 10 nodes with different load and computing power. Each node consists of 4 processors. The specification of each processor is Intel(R) Core(TM)2 Duo CPU with different computing power. The load and frequency of each processor is used by calling the random function. Table1 and Table2 give the analysis for the minimum load of a node and maximum ACR (available CPU Resources) of a processor respectively.

Table 1.Analysis of Minimum Load of a Node

Node No	Load of Node				
	Load of P1	Load of P2	Load of P3	Load of P4	Total load of Node
1	34	59	33	91	217
2	46	51	31	28	156

Table2: Analysis of maximum ACR

Node No	Processor No	Load (no of jobs)	Frequency (GHZ)	Idle Time (in terms of no of jobs)	ACR
2	P_{21}	46	1.6	110	1.76
	P_{22}	51	1.3	105	1.36
	P_{23}	31	1.5	125	1.87
	P_{24}	28	1.5	128	1.92

Job is allocated to the Processor4 of Node 2

4.1Experimental Results

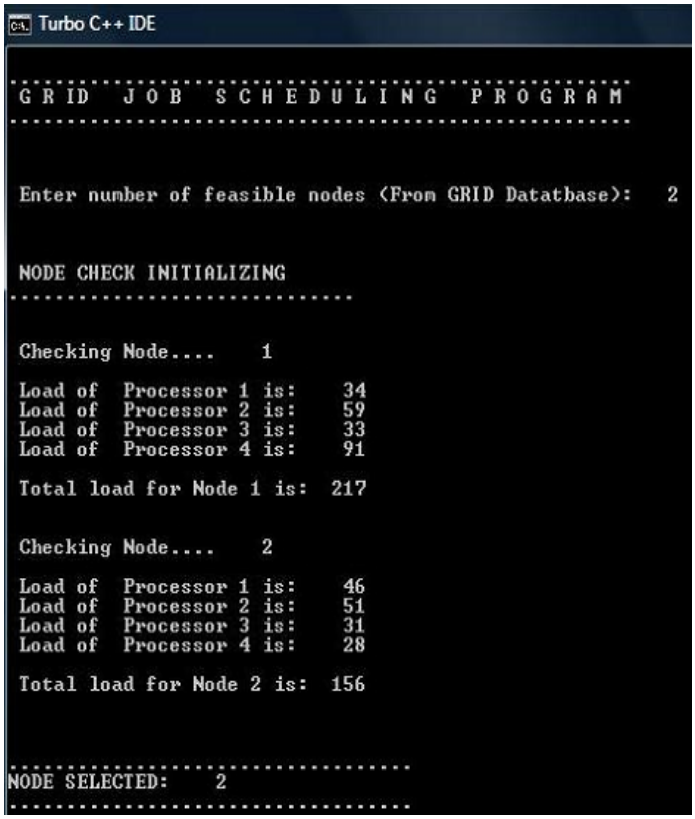

Fig5: Node Selection

Fig6: ProcessorSelection

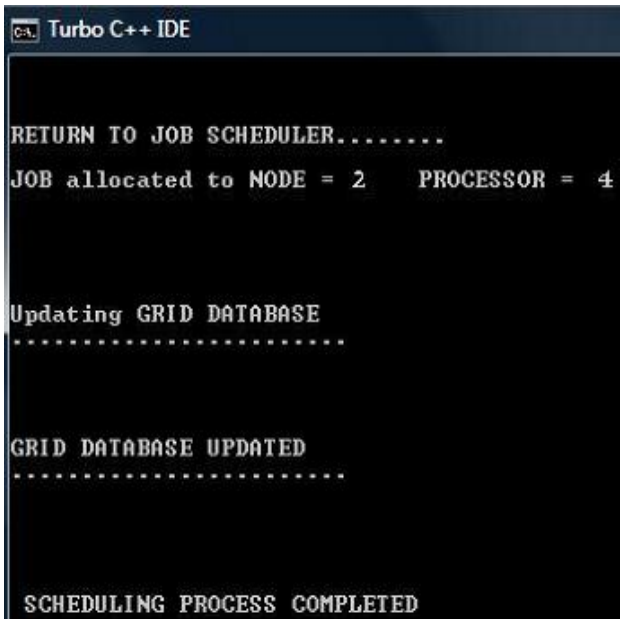

Fig7: Job Allocation

5. CONCLUSION AND FUTURE WORK

The success of grid computing will depend on the effective utilization of the system for various computationally intensive jobs. Given a vast number of resources that are available on a Grid, an important problem is the scheduling of jobs on the grid with various objectives.

In this paper a model and a job scheduling algorithm in grid environment have been presented. This algorithm has a key role in increasing the speed of executing input jobs by considering the parameters of input jobs, load and computational capability of resources. We created our own Grid simulation environment to carry out the simulation of our scheduling algorithm. The result of simulation and proposed algorithm analysis shows that it plays a key role in reducing the average job completion time.

Now we discuss some of the limitations of this work and present some possible directions for future research. In this work, we assume that there is no precedence constraint among different jobs or different tasks of a job. Usually, the jobs are independent of each other in the grid, but different tasks of a job may have some precedence constraints. Hence, it is an interesting direction for future research. Such dependencies will not only make the problem extremely difficult to solve, but would also require estimating a very large number of parameters. In the future we should also consider some fault tolerant measures to increase the reliability of our algorithm.

6. REFERENCES

[1] I. Foster, C. Kesselman, S. Tuecke. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations.International Journal Supercomputer Applications, **15**(3).

[2] Rajkummar Buyya.2002.Economic-based Distributed Resource Management and Scheduling for grid computing.PhD thesis, Monash university, Melborn, Australia, (April 2002)

[3] M. Aggarwal, R.D. Kent and A. Ngom.2005. Genetic Algorithm Based Scheduler for Computational Grids.In Proc. of the 19th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'05), pp.209- 215(May 2005.), Guelph, Ontario Canada.

[4] Jamshid Bagherzadeh,Mojtaba MadadyarAdeh.2009. An Improved Ant Algorithm for Grid Scheduling Problem.In Proceedings of the 14th International CSI Computer Conference .

[5] Fangpeng Dong and Selim G. Akl ,School of Computing, Queen's University Kingston, Ontario. 2006 .Technical Report No. 2006-504. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems,

[6] D Wright.2001. Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor.In Proceeding of HPC Revolution 01, Illinois.

[7] K. Al-Saqabi, S. Sarwar, and K. Saleh.1997. Distributed gang scheduling in networks of heterogeneous workstations, Computer Communications Journal, pp.338-348.

[8] Maheswaran M, Ali S, Siegel H J, et al.1999.Dynamic mapping of a class of independent tasks on to heterogeneous computing systems. In the 8th IEEE Heterogeneous Computing Workshop (HCW '99),San Juan, Puerto Rico,(Apr. 1999), pp.30-44.

[9] XiaoShan He, XianHe Sun, and Gregor von Laszewski.2003.QoS Guided Min-Min Heuristic for Grid Task Scheduling, Computer Science and Technology, 18(4):442-451.

[10] Malarvizhi.N, Rhymend Uthariaraj. 2008. A Broker-Based Approach to Resource Discovery and Selection in Grid Environments. In IEEE International Conference on Computer and Electrical Engineering, pp. 322-326.

[11] Feitelson, D.G.Rudolph R. 1995. Parallel Job Scheduling: Issues and Approaches, Springer Lecture Notes in Computer Science, vol 949, pp. 1- 18.

[12] M. Harchol-Balter, T. Leighton, D. Lewin.1999.Resource Discovery in Distributed Networks. In18th ACM Symposium on Principles of Distributed Computing.

[13] Anne Benoit, Murray Cole, Stephen Gilmore and Jane Hillston.2005. Enhancing the effective utilization of Grid clusters by exploiting on-line performability analysis.In IEEE International symposium on Cluster Computing and the Grid, pp.317-324.

[14] Shun-Li Ding, Jing-Bo Yuan and Ji-U-Bin Ju.2004.An algorithm for agent based task scheduling in grid environments, proceedings of International Conference on Machine Learning and Cybernetics, pp2809- 2814.

[15] Huyn zhang, chanle wu, Q.xiong, and L.Wu,G. Ye. 2006. Research on an Effective Mechanism of Task Scheduling in Grid Environment.In IEEE, Fifth International Conference on Grid and Cooperative Computing (GCC'06).

[16] A.Anjum, R. McClatchey, H. Stockinger, A. Ali, I. Willers, M. Thomas, M. Sagheer, K. Hasham, and O.Alvi.2006. DIANA Scheduling Hierarchies for Optimizing Bulk Job Scheduling. In Proc. Second IEEE International Conference On e-Science and grid computing (e- Science'06).

[17] T.Wang, X. zhou, Q. Liu, Z. Yang, and Y. Wang.2006. An adaptive Resource Scheduling Algorithm for Computational Grid.In IEEE Asiapacific Conference on Services Computing (APSCC'06).

[18] Jie Song, Chee-Kian Koh, Simon See, and Gay Kheng.2005. Performance Investigation of Weighted Meta-scheduling Algorithm for Scientific Grid.The 4th International Conference on Grid and Cooperative Computing (GCC 2005) LNCS 3795,pp.1021-1030.

[19] F. Kon, R. Campbell, M. Mickunas, and K. Nahrstedt.2000. 2K: A distributed operation system for dynamic heterogeneous environments. In: Proc. 9th IEEE Int'l Symposium on High Performance Distributed Computing (HPDC '00).

[20] P. Chandra, A. Fisher, C. Kosak.1998. Darwin: Customizable resource management for value-added network services. In: Proc. 6th IEEE Int'l Conference on Network Protocols.

[21] S. Chapin, J. Karpovich and A. Grimshaw. 1999. The Legion resource management system. In: Proc. 5th Workshop on Job Scheduling Strategies for Parallel Processing.

[22] Blazewicz, J. Brauner, and Finke. 2004. Scheduling with Discrete Resource Constraints, In Lueng (2004). chapter 23.