RR Based Grid Scheduling Algorithm

Manjot Bhatia
Research Scholar, Delhi University
New Delhi, India
+91-9810555592
manjot_bhatia@hotmail.com

ABSTRACT

Grid computing is a high performance computing environment to solve larger scale computational demands. Grid computing contains resource management, task scheduling, security problems, information management and so on. Task scheduling in an important aspect of distributed computing. As grid computing is a form of distributed computing with heterogeneous resources working in a shared environment with no central control. The main aim of Grid scheduling is to increase the system throughput and to satisfy the job requirements from the available resources. In this paper we introduce a new resource requirement (RR) based task-scheduling algorithm for grid computing. The new algorithm is based on the resource requirement of the jobs. It has been tested in simulated grid environment. The experimental results showed a significant improvement in terms of a smaller makespan time in various applications.

Categories and Subject Descriptors

D.4 [**Operating Systems**]: D.4.1 [**Process Management**]: Scheduling

General Terms

Algorithms, Management, Performance

Keywords

Task Scheduling, Grid computing, requirement based scheduling, Distributed environment

1. INTRODUCTION

Grid computing is a form of distributed computing in which resources are geographically distributed and owned by different individuals with different technologies. This distributed environment allows sharing of geographically distributed heterogeneous computers and resources. Users can access and utilize the resources of multiple domains participating in the grid network. It's a new technology that easier access to remote computational resources to tackles complex computations. Grids computing aims to maximize the utilization of an organization's computing resources by making them shareable across applications (sometimes called virtualization) and, potentially, provide computing on demand to third parties as a utility service. These resources can be shared by various applications depending upon their availability and QOS requirement of the applications. It scheduled the independent jobs submitted by different users on dynamically distributed resources that increases the overall throughput & also utilizes the unused processors. As grid computing allows user to access remote resources & provide cooperative distributed computing environment, so user jobs can be executed either on local or remote computer systems. The job of the grid scheduler is to automatically assign the suitable resources to the independent jobs to maximize the system utilization. It also reduces the average response time of the jobs. The efficient utilization of grid computing resources can improve the overall job-throughput due to load balancing of the tasks between the grid resources.

The grid scheduling is divided into three phases [7]. These are Resource exploring, Machine selection and Executing. The first phase recovers all the available resources, the second phase deals with finding the best match between the set of jobs and available resources. The phase two is a NP-hard Problem [13]. The behavior of computational grid is dynamic and unpredictable as it depends upon various factors:

(i) network connection (ii) availability & non-availability of resources at the required time (iii) the no. of resources joining & leaving the grid (iv) performance of grid resources can vary from time to time.

The jobs will take different execution time on different machines. So, task scheduling in grid environment is a problem to schedule a stream of applications from different users to a set of computing resources to minimize the total completion time. This scheduling requires the matching of different jobs with the machines that satisfy their resource requirement. There are two different goals for task scheduling: (i) Increasing computing performance, its aim is to minimize the execution time of each application that is considered in parallel processing. (ii) Increasing overall throughput, its purpose is to schedule a set of independent tasks in such a way that it increases the processing capacity of the systems for long period of time.

We focused on the second goal and proposed a new task-scheduling algorithm for grid computing that provides high throughput and efficient utilization of resources. This grid-scheduling algorithm tries to minimize the total turnaround time of the jobs. The resources in grid environment are not only dedicated to grid applications, as they have to handle their own local jobs also. So, the grid jobs need to compete for the resources according to their resource requirement. In this scheduling algorithm we are considering the main-memory as the resource requirement of the grid jobs. The scheduler considers the resource requirement of the grid jobs and assigns the jobs to the resources that satisfy their resource requirement. Our Scheduling algorithm increases the efficiency and utilization of grid resources by scheduling the jobs based on their resource requirement.

ACAI '11, July 21 - July 22 2011, Rajpura/Punjab, India

2. LITERATURE SURVEY

In the past few years, researchers have proposed scheduling algorithms for parallel system [2, 7, 15]. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, large number of researchers [1, 9, 11, 13, 14] is showing interest in it. Current systems [8] of grid resource management was surveyed and analyzed based on classification of scheduler organization, system status, scheduling and rescheduling policies. However, the characteristics and various techniques of the existing grid scheduling algorithms are still complex particularly with extra-added components.

At the present time, job scheduling on grid computing is not only aims to find an optimal resource to improve the overall system performance but also to utilize the existing resources more efficiently. Recently, many researchers have been studied several works on job scheduling on grid environment. Some of those are the popular heuristic algorithms, which have been developed, are UDA [12], OLB [12], min-min [12], the fast greedy [12], GA [12], SA [12], tabu search [12] and an Ant System [4]. These algorithms have several advantages and have some drawbacks also. UDA assign too many jobs to a single grid node. This leads to overloading and the increases the response time of the jobs. The drawback of OLB is that it does not achieve the load balance and leads to hard calculation of minimum completion time for a job. The algorithms GA, SA and GSA are difficult to implement. The heuristic algorithms proposed for job scheduling in [12] and [4] depend on static environment of system load and the expected value of execution times. In paper [10], the job will be moved from one machine to another machine. So the traffic in the grid system will be automatically increased. In paper [6] they considered communication cost and different ant agents.

Currently available Grid Resource Management system like: Condor, Globus, NetSolve, Nimrod/G, AppLeS uses different Grid scheduling approaches. The Condor uses centralized scheduler and designed to improve overall throughput of the system in a controlled network environment. Its scheduling algorithm does not consider any QoS requirement of jobs. The AppLeS scheduling algorithm focuses on efficient co-location of data and experiments as well as adaptive scheduling. In our algorithm we consider QoS in scheduling. The Nimrod uses decentralized scheduler and its scheduling approach is based on predictive pricing model and Grid economy model. The Netsolve has decentralized scheduler and scheduling approach focuses on fixed application oriented policy considering soft QoS. Our proposed algorithm is different from the above given algorithms; it considers the memory requirement of the job and assigns the jobs to the available resource accordingly.

3. RESOURCE REQUIREMENT BASED GRID SCHEDULING MODEL

3.1 Proposed Grid Model Architecture

In this we assumed a computing grid 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. For this reason, the job scheduling in this environment is complicated.

The independent users submit their jobs to Job collector. 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 (memory requirement) of the jobs and the user's authentication. The job collector submits the collected jobs with their corresponding resource requirements to the Grid scheduler. The scheduler retrieves the information of the grid resources such as CPU speed, Size of memory, number of CPUs in the resource from the Grid Resource Database (GRD). The information retrieved from the GRD is used to find the optimal resources according to job resource requirement for processing of jobs. GRD is a database that maintains records of the resources and its corresponding parameters.

The purpose for a Job Distributor is to send jobs to the selected resources .The function of the Grid Resource Database (GIS) is to maintain and update the status of the grid resources.

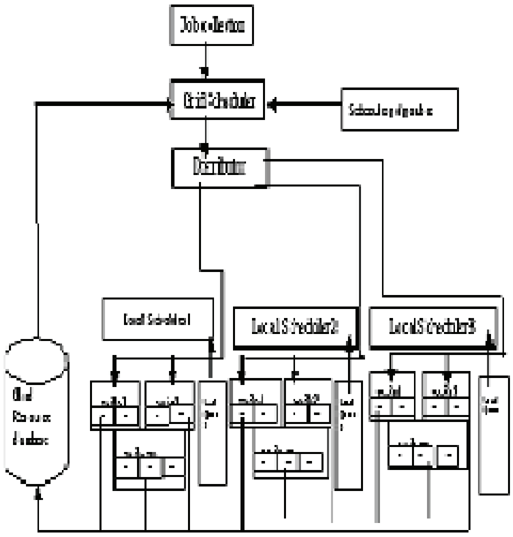

Figure1: Grid Model Architecture

3.2 Proposed Grid Scheduling Algorithm

The major objective of our algorithm is to allocate the best suitable machine to the tasks, arriving at the Grid Scheduler. As our task scheduling algorithm is based on the availability of memory required by the task. The Grid Resource Database needs to maintain the data of the resources for each and every site in the grid environment.

Some terms and its definitions, used in this paper:

The time spent by the i^{th} process in the waiting queue(Grid Scheduler) is

$$T_w(i) = \text{Max}(\min \boldsymbol{tp_j}\,(\text{avail})\,,\ \min \boldsymbol{tme_j}\,(\text{avail}))$$

where $\boldsymbol{tp_j}$ is time that shows after how much time the i^{th} process has got the j^{th} processor, $\boldsymbol{tme_j}$ is time i.e after how long time the i^{th} process has got the j^{th} processor that satisfy its memory requirement($\boldsymbol{me_i}$). Let ET_{ij} is defined as the amount of time taken by machine $\boldsymbol{m_j}$, to execute task t_i, given that m_j has no load when ti is assigned.

$$ET_{ij} = outTime_{ij} - inTime_{ij}$$

Where $\boldsymbol{outTime_{ij}}$ is the at which i^{th} process is completed in the jth processor.

$\boldsymbol{inTime_{ij}}$ is the time at which i^{th} process is submitted to the j^{th} processor

The completion time C_{ij} of the i^{th} process at j^{th} machine is :

$$C_{ij} = T_w(i) + ET_{ij}$$

Make span is a measure of the throughput of the heterogeneous system. The aim of our grid scheduling algorithm is to minimize the makespan.

The heuristic can be divided into two categories online mode and the other batch mode. In online mode, whenever a job arrives to the scheduler it is allocated to the first free machine. In this method, the arrival order of the job is important. Each task is considered only once for matching and scheduling. In case of batch mode, the jobs are collected in a set and are examined for mapping at prescheduled times called mapping events. This independent event uses heuristic approach to make better decision. This mapping heuristics do better task/host mapping because the heuristics have the resource requirement information for the meta-task, and know the actual execution time of a larger number of tasks. Several heuristics approaches like Min-min, Max-min, UDA and GA are proposed for scheduling independent tasks. Most of these algorithms consider the expected execution time of each task as the criteria to make better decision. The general scheduling algorithms does not consider the resource requirement, which affects scheduling process in a Grid. Regardless of their computing power request, some tasks may require more memory whereas others can be satisfied with less memory. For e.g. If scheduler assigns a task that require less memory for execution on the processor with high memory, tasks requiring high memory will then have to wait. Considering memory as the resource requirement in scheduling should lead to a better scheduling algorithm. Based on this requirement, a new scheduling algorithm based on 'memory as resource requirement' is proposed.

3.3 Scheduling Algorithm

for all hosts p_j
read me_j(available memory of host)
enter it in GRD
end for
for all jobs t_i in the waiting queue(of Grid Scheduler)
for all hosts p_j
for each job t_i , find a host p_j from the GRD that fulfill the job's memory requirement
calculate
$T_w(i) = \text{Max}(\min \mathbf{tp}_j \text{(avail)}, \min \mathbf{tme}_j \text{(avail)})$
assign job t_i to the specified host pj
calculate
ET_{ij} = outTime_{ij}-inTime_{ij}
$C_{ij} = T_w(i) + ET_{ij}$
delete job t_i from the list
end for

Figure 2. RR based scheduling algorithm

4. EXPERIMENTAL TESTING

We have developed our own simulation for evaluation of newly proposed RR based scheduling algorithm in Grid environment. In the experimental testing we used heterogeneous machines and tested it for different number of tasks. We compared the results of our RR based scheduling algorithm with the commonly used FCFS algorithm. The results are shown in Table 1.

It is observed that makespan for the given number of tasks is shorter in case of RR based scheduling algorithm as compare to FCFS algorithm. Our algorithm has shown an improvement of approximately above 10%.

Table 1: Makespan(in secs) for RR based algorithm and FCFS

No. of tasks	FCFS	RR scheduling	Improvement
3	0.000515556	0.000416752	19.16%
5	0.001216293	0.000786133	35.37%
7	0.00165708	0.001437424	15.28 %

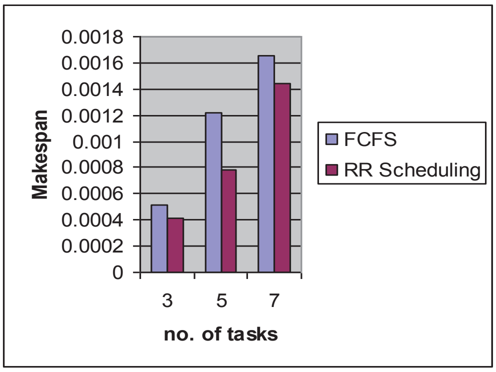

Figure 3. makespan of FCFS and RR based scheduling

5. CONCLUSION

In this paper we have proposed a scheduling algorithm that considers the resource requirement of the jobs in the Grid environment. This newly proposed scheduling algorithm achieve high throughput in the Grid computing. A simulation system was developed to test the Resource requirement based scheduling algorithm in a simulated Grid environment. We used the makespan time of batch jobs as the comparison criteria. When Compared with First come first served scheduling algorithm, the experimental results show that RR based scheduling algorithm show a noticeable increase in performance than FCFS algorithm. As memory is an important resource, this research work considers memory as a resource requirement of the job. In future research can be done considering others factors as the resource requirement in task scheduling algorithm.

6. REFERENCES

[1] Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour "Economic Scheduling in Grid Computing".

[2] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, K.C. Sevcik, and P. Wong. "Theory and practice in parallel job scheduling", In 3rd *Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1291, pages 1–34, 1997.

[3] Fernandez-Bacad " Allocating modules to processors in a distributed System", IEEE transaction on Software Engineering 15-11 1989 1427-1436

[4] G. Ritchie and J. Levine, "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments".

[5] Gilbert C. Sih and Edward A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures", IEEE Trans. Parallel and Distributed Systems, vol.4, pp175-187, Feb.1993.

[6] HUI YAN, XUE-QIN SHEN, XING LI, MING-HU MU, "An Improved ant algorithm for Job Scheduling in Grid Computing" IEEE Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

[7] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. "On the design and evaluation of job scheduling algorithms", In *5th Workshop on Job Scheduling Strategies for Parallel Processing*, volume LNCS 1659,pages 17–42, 1999.

[8] K. Krauter, R. Buyya and M. Maheswaran, "A taxonomy and survey of Grid resource management systems for distributed computing", Software Pract. Exp. 2 (2002) 135–164.

[9] K. Li, "Job scheduling and processor allocation for grid computing on Metacomputers ", *Journal of Parallel and Distributed Computing*, *Elsevier,* 2005

[10] Li Liu, Yi Yang, Lian Li and Wanbin Shi, " Using Ant Optimization for super scheduling in Computational Grid,

IEEE proceedings of the 2006 IEEE Asia-pasific Conference on Services Computing (APSCC' 06)

[11] Schopf J.M: "A General architecture for scheduling on the Grid" Special issue of JPDC on Grid Computing 2002.

[12] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F.Freund (2001), "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", *Journal of Parallel and Distributed Computing*. Vol.61(6): Pages 810-837.

[13] V. Hamscher, U. Schwiegelshohn, A. Streit, R.Yahyapour, "Evaluation of job-scheduling strategies for grid computing", *Proceedings of First IEEE/ACM International Workshop on Grid Computing, Lecture Notes in Computer Science*, vol. 1971, Springer, Berlin, 2000, pp. 191–202.

[14] Xiaoshan He1 Xian-He Sun1 Gregor Von Laszewski2 " A QoS Guided Scheduling algorithm for Grid Computing ", USA.

[15] Zhou, Brent and Qu X. "An efficient scheduling Algorithm for Multiprogramming on parallel Computing System" , In Proceedings of the 20th Australasian Computer Science Conference , Sydney, Australia ,Feb 1997