International Journal of Engineering, Science and Metallurgy

(Research article)

Min-Min Approach for Scheduling in Grid Environment

Preeti Bansal[a] and Gaurav Sharma[a]

[a]*CSE Deptt.JMIT Radaur.*

Accepted 4 July 2012, Available online 1Sept 2012

Abstract

Scheduling jobs on computational grids is identified as NP-complete problem due to the heterogeneity of resources; the resources belong to different administrative domains and apply different management policies. This paper presents a novel metaheuristics method based on min min approach for scheduling of jobs in the grid environment. The proposed method schedules the jobs providing QOS to the jobs. Jobs are classified on the basis of their communication and computational requirement. Depending on this it is scheduled to the required type of processor providing the desired QOS. The algorithm is better than the typical scheduling algorithms as it not just schedules the jobs based on processor speed but also considers the bandwidth requirements.

Keywords: *QOS, Metaheuristics*

1. Introduction

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components make it possible to construct large-scale high-performance Grid computing systems. These technical opportunities enable the sharing, selection, and aggregation of geographically distributed heterogeneous resources for solving large-scale problems in science, engineering, and commerce [1,2]. To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. The scheduling problem deals with the coordination and allocation of resources so as to efficiently execute the users' applications.

A major issue in Grid Environment is how to distribute the tasks among processors to improve performance so that some jobs do not suffer unbounded delays. Dandamudi [3] defines task routing as the method that tasks are assigned to processors and task scheduling as how tasks are scheduled on the assigned processor. Task routing policies may be either static or adaptive. The former uses only information about the average behaviour of the system, ignoring the current state, while the latter reacts to system state. Adaptive policies are more complex but produce significantly better performance results than static policies. Static policies are separated into deterministic and probabilistic. Probabilistic policies adjust the routing decision to a probability distribution. In deterministic policies once the set of currently ready tasks has been specified, the routing discipline is applied [3]. Task routing algorithms can also be separated into immediate mode and batch mode routing algorithms. Immediate mode algorithms forward the tasks of each job as soon as they arrive in the system [4]. On the contrary, batch mode algorithms allocate a batch of tasks of many jobs which are in the queue of the scheduler [5].

While considering the scheduling of the resources many factors such as CPU utilization rate, throughput, turnaround time, waiting time, response time should be focused for all the processors when assigned with the jobs [6]. The jobs are assigned to the resources considering the system's performance. Thus the scheduling plays an important role in achieving the best utilization of resources and the better completion of the submitted jobs. The scheduling problem is a NP hard problem and the solutions for these problems need heuristics [7]. Many heuristic scheduling algorithms have been designed for this purpose.

2. Need for Grid Technology

Computers have been proven to be very efficient to solve complex scientific problems. They are used to model and simulate problems of a wide range of domains; for instance medicine, engineering, security control and many more. Although their computational capacity has shown greater capabilities than the human brain to solve such problems, computers are still used less than they could be. One of the most important reasons to this lack

of use of computational power is that, despite the relatively powerful computing environment one can have, it is not adapted to such complicated computational purposes. The following are given the reasons for why we need grid computing.

2.1 Exploit Unused Resources

In most organizations, computing resources are underutilized. Most desktop machines are busy less than 25% of the time (if we consider that a normal employee works 7 hours a day and that 42 hours a week and that there are 168 hours per week) and even the server machines can often be fairly idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usages. The easiest use of Grid computing would be to run an existing application on several machines. The machine on which the application is normally run might be unusually busy, the execution of the task would be delayed. Grid Computing should enable the job in question to be run on an idle machine elsewhere on the network.

2.2 Increase Computation

To provide users with more computational power, some crucial areas have to be considered. These areas are: hardware improvement, periodic computational needs capacity of idle machines sharing of computational results

3. Problem definition

In grid environments, the shared resources are dynamic in nature, which in turn affects application performance. Workload and resource management are two essential functions provided at the service level of the Grid software infrastructure. To achieve the desirable goals i.e. increase throughput, maximum system utilization, and fulfil economical system and user constraints of these environments, effective and efficient task scheduling algorithms are fundamentally important. The focus of our study is to consider factors which can be used as characteristics for decision making to initiate task scheduling.

At first, we discover the set of potential resources suitable for the proposed tasks. Second, we select scheduling constraints such as minimization of the run time. After the matching problem is resolved we conduct the final job execution and data transfers with the selected environment.

The main objective of this paper is to provide a good scheduling algorithm for Grid environment. In the Grid environment, it is desirable to compete for the best QOS provided by and for remote resources to fulfil application constraints. The scheduler in the Grid environment needs to consider application and QOS constraints to get a better match between applications and resources.

4. Proposed algoritm

The term quality of service (QOS) is used differently based on the different context while applying it to grid resources. For example when QoS is used in the network context it means the desirable bandwidth for the application. Similarly when it use for CPU it means the requested speed like FLOPS or the utilization of the underlying CPU. In our proposed approach both QoS of a network and QoS for CPU is considered. In current Grid job scheduling, the job with different levels of QoS request compete for resources. It may be possible that a job with low QoS requested can be executed on a resource providing high quality of service. So when the job with high QoS requested is come for execution it has to wait till the low QoS requested job completed, this will increase the jobs completion time, makespans time and decrease the overall performance of the Grid. To overcome this shortcoming we modify the Min-Min algorithm to take the QOS matching into consideration while scheduling.

There are two types of jobs in grid, computation based and communicational based. The communication based jobs like transfer a file from one node to another node require high bandwidth for its operation. The computational based jobs like solving scientific computation based problems which require high speed CPU to complete the assigned task in minimum delay of time. If the computational jobs submitted to high bandwidth resource then it will not utilize its bandwidth effectively, similarly if a communication based jobs submitted to a resource having high speed CPU and low bandwidth then it does not fully utilize the resource and also increase its job completion time. So we proposed an algorithm which considered the QoS in scheduling should lead to a better scheduling algorithm.

Step1. We divide the resources in the four different classes. Class1 Low QOS and Class2 high QoS in term of bandwith. Class3 Low QOS and Class4 high QoS in term of CPU speed.

Step2. Calculate the job is communication based or computational based by computing CCR value(Communication to computation ratio).
Step3. If the job is communication based and requested for high QoS then it is submitted to the class2 resources.
Else If the job is communication based and requested for low QoS then it is submitted to the class1 resources .
Else if the job is computational based and requested high QoS then it is submitted to the class4 resources.
Else Job is submitted to class3 resources.
End if

Step 4. We apply Min-Min algorithm and computes the completion time of all the jobs on all the host of that class.
Step 5. Submit the job to resource which requires least completion time to execute that job.

The proposed QoS based Min-Min algorithm significantly decreases the completion time and makespans time of the jobs as compare to Min-Min algorithm.

5. Experimental setup

We have setup a simulated Grid environment to evaluate the proposed QoS guided Min-Min scheduling algorithm. In our experiment, we fixed the parameter for the hosts and used six task submission scenarios three for communicational based jobs and three for computational based jobs. The QoS guided Min-Min and the conventional Min-Min are compared by their makespans on the same set of tasks.

First scenario:- Most of the jobs(80% jobs) require high QoS.
Second scenario:- half of the jobs(50% jobs) require high QoS.
Third scenario:- very few of the jobs(20% jobs) require high QoS.

TABLE 1. Makespan for Three Scenario for Two Heuristics

Scenario	Min-Min	QoS Guided Min-Min	Improvements
First	125.40	90.78	27.61%
Second	172.55	110.28	35.77%
Third	258.87	240.77	6.99%

For each of the scenarios, we compare the performance of the conventional Min-Min heuristics and the QoS guided Min-Min. For each scenario and each heuristic we create 100 tasks 100 times independently and get the average makespan of the 100 times. Table 1 and Fig. 1 shows the comparison. The data is in seconds.

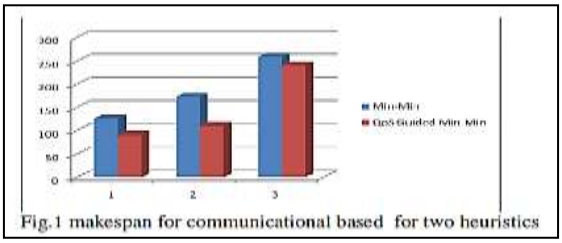

Fig.1 makespan for communicational based for two heuristics

Fig. 1 shows the comparison. The data is in seconds.

As shown in Fig. 1, for all three scenarios the QoS guided Min-Min outperforms the traditional Min-Min heuristics by 23.46% shorter makespam. For scenario (a), where the tasks that require high QoS are in higher density (80%), a satisfactory performance gain 27.61% is acquired. For scenario (b) where the tasks that require high QoS and the tasks that require low QoS are evenly distributed, the performance gain reaches as high as 35.77%. For scenario (c), where the tasks that require high QoS is only 20%, the performance gain of the QoS guided Min-Min is relatively small, i.e., 6.9% better than the conventional Min-Min.

6. Conclusion

Job scheduling is one of the well-known problems in distributed computing systems such as grid environments. In this paper we propose an adaptive scheduling algorithm which considers QOS for network as well as CPU based upon the types of jobs. In proposed method we enhance the Min-Min algorithm by classifying it according to the QOS parameters. The experimental results shows that QOS guided Min-Min scheduling algorithm outperform the traditional Min-Min heuristic on the same set of task.

References

1. M. Wieczoreka, A. Hoheiselb, R. Prodana, Towards a general model of the multi-criteria workflow scheduling on the grid, Future Gener. Comput. Syst. 25 (3) (2009) 237–256.
2. I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, 1999.
3. S.P. Dandamudi, Performance implications of task routing and task scheduling strategies for multiprocessor systems, in: Proceedings of the IEEE Euromicro Conference on Massively Parallel Computing Systems, Ischia, Italy, May 1994, pp. 348–353.
4. F. Xhafa, L. Barolli, A. Durresi, Immediate mode scheduling of independent jobs in computational grids, in: Proceedings of the 21st International Conference on Advanced Networking and Applications (AINA'07), May 2007, IEEE, 2007, pp. 970–977.
5. F. Xhafa, L. Barolli, A. Durresi, Batch mode scheduling in grid systems, International Journal of Web and Grid Services 3 (1) (2007) 19–37.
6. Fangpeng Dong and Selim G. Akl, 2006. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report, School of Computing, Queen's University, Canada.
7. He Xiaoshan, Xia-He Sun, Gregor Von Laszewski, 2003. QoS Guided Min-min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology, pp. 442-451, DOI:10.1007/BF02948918.
8. Sheng-De Wang,I-Tar Hsu, Zheng Yi Huang: Dynamic Scheduling Methods for computational grid environments ,international conference on parallel and distributed systems 1(2005) 22-28.
9. Zhihong XU. Xiangdan HOU. Jizhou SUN , Ant algorithm-based task scheduling in grid computing ,Canadian conference on Electrical and computer engineering 2May(2003)1107-1110.