A Survey on QoS Based Task Scheduling Approach in Grid Computing

Mr. Jujhar Singh[1], Mr. Gaurav Sharma[2]

[1](M.Tech, Computer Science and Engineering Department, JMIT, Radaur/ Kurukshetra University, India)

[2](Assistant Professor, Computer Science and Engineering Department, JMIT, Radaur/ Kurukshetra University, India)

ABSTRACT : *Science and engineering are the fields which deal with the complex computational problems. So, dealing with these problems requires a very robust platform and Grid Computing provides the required functionality. Geographically distributed heterogeneous resources define the Grid computing scenario. The major problem which arises in the area of Grid Computing is the task scheduling which tends to select the best resource for a given job. There are many scheduling heuristics (online and batch mode) are available but they are not able to achieve the maximum objectives. So, Quality of Service (QoS) based approach is required to achieve the maximum objectives. The QoS based approach keeps into account the QoS based characteristics for both the tasks and the resources. In this paper we study the different heuristics techniques for resource selection in Grid Computing and also we go through the need and significance of this QoS based approach in Grid Computing.*

Keywords *- Grid Computing, Task Scheduling, Grid Scheduler, Online mode heuristics, Batch mode heuristics.*

1. INTRODUCTION

Within the last few years we have witnessed that the computing-intensive fields like meteorology, physics, medicine and others are making use of Grid computing for satisfying the increasing demand of the computing power. Grid resource management involves dealing with three classes of stakeholders - end users making use of grid resources, owners of resources, and grid administrators. Each class of stakeholders has their own perspective and preferences, which result in different, often contradictory, criteria for scheduling (main step of Grid Resource Management) [1]. To increase the level of satisfaction of these stakeholders, grid management system must use the scheduling heuristic, which provides compromise solution (i.e. a compromise schedule) using the many conflicting objectives. Geographically distributed computers, linked through the Internet in a Grid-like manner, are used to create virtual supercomputers of vast amount of computing capacity able to solve complex problems from e- Science in less time than known before.

Also resources in grid system are heterogeneous, geographically distributed, belong to different administrative domains and apply different management policies [2]. Furthermore grid management systems do not have full control over the resources that belong to the grid. The term resource is defined in grid context to denote any capability that may be shared and exploited in a networked environment. The resources and services may differ in form of functionalities that they offer to the user, but both are similar in the way that they provide those functionalities to the users; hence we can use the term resource more generally to refer to all types of ancient resources, as well as services.

The key goal of grid computing is to design a system that can provide improved efficiency and a platform for proper utilization of all computing resources within an enterprise or extended enterprise to meet end user demands [3]. But the heterogeneous and decentralized nature of Grid makes it very complex and selecting appropriate resources for jobs has become a critical issue due to rapid increase of resources in the grid. Grid environments ought to provide effective service and mechanisms to select the most adequate resources for satisfying application requirements. In large grid computing system it is unwieldy for an individual to select the resources manually. So resource selection mechanism and scheduling of tasks into machines are required for better performance. Resource selection usually occurs after resource discovery phase. While the first phase filters out unwanted resources, this phase should determine from this large list the best set of resource(s) chosen to map the application.

A Grid scheduler (or broker) must make resource selection decisions in an environment where it has no control over the local resources, the resources are distributed, and information about the systems is often limited or dated. Here, schedulers are responsible for job management like allocating

resources needed by particular job, partitioning of the job to run the job in the parallel manner in parallel processing environment, management of data, event correlation and service-level management capabilities. 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. As heuristic techniques are increasingly being used for solving optimization problems, they have proven themselves as a good candidate in this area. This can be inferred by recent research in the area.

Scheduling of task on heterogeneous grid resources is known to be a NP-complete problem; therefore, to get a near optimal solution within finite duration, heuristics/meta heuristics are used instead of exact optimization methods. For the majority of Grid systems, scheduling is a very important mechanism [4]. In the simplest of cases, scheduling of jobs can be done in a blind way by simply assigning the incoming tasks to the available compatible resources. Nevertheless, it is a lot more profitable to use more advanced and sophisticated schedulers. Moreover, the schedulers would generally be expected to reactto the dynamics of the Grid system, typically by evaluating the present load of the resources, and notifying when new resources join or drop from the system. Additionally, schedulers can be organized in a hierarchical form or can be distributed in order to deal with the large scale of the Grid. An important issue here is how to formally define the task scheduling problem.

A large number of heuristic algorithms have been designed to schedule tasks to machines on grid computing systems. The commonly used algorithms are Opportunistic Load Balancing (OLB), Max-min, Min-min, Minimum Execution Time (MET), Minimum Completion Time (MCT), User Directed Assignment (UDA), Genetic simulated annealing (GSA), Tabu search, simulated annealing (SA), genetic algorithm (GA), ant colony optimization (ACO) and particle swarm optimization (PSO).

In this paper mainly we will focus on online mode and batch mode heuristics which are prevalent in their use to solve the task scheduling problem. We will go through their limitations for being unable to achieve the maximum objectives and also we will see that how QoS based task scheduling approach is better than the existing heuristics. This paper is organized in such a way that section 1 which is the basic introduction of the Grid computing and the task scheduling approach is followed by the section 2 which discusses an analysis of already existing batch mode and online mode heuristics. Section 3 deals with the need of QoS based task scheduling approach which arises due to the limitations of already existing heuristics. Section 4 is related to the significance of this QoS based task scheduling approach and the last section 5 ends up the paper with the conclusion.

2. AN ANALYSIS OF EXISTING TASK SCHEDULING HEURISTICS

In this section we will study the already available online mode and batch mode heuristics.

2.1 Selection Heuristics

In the process of resource selection in heterogeneous environment, it is difficult to find rules that determine the right selection decisions [2]. However researchers use heuristics to help obtaining better selection decisions. Heuristics are approaches that help in making the right decisions; but they do not always produce the correct selection decisions. There are two different types of heuristics as shown in the Fig. 1 that are commonly used in the process of resource selection; probabilistic or stochastic and deterministic. In probabilistic heuristics variables states are not determined uniquely, but they are determined using probability distributions. However, in deterministic heuristics, all resources states are uniquely determined by parameters in the heuristics model. Hence deterministic heuristics achieve the same result for a given initial conditions which is not necessarily occur when using probabilistic heuristics.

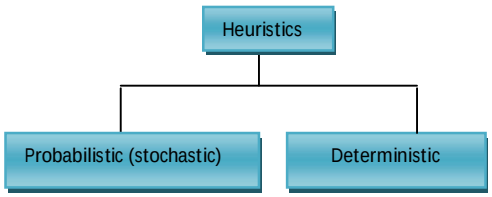

Fig. 1 showing the types of heuristics

2.2 Mapping Methods

The process of mapping jobs to resources and figuring out the execution order of the jobs allocated to each resource is classified into dynamic and static, Fig. 2. Dynamic techniques do the mapping immediately when jobs arrive. Whereas in static techniques, the total set of jobs is identified a priori and the mapping process is done

previous to the execution of any of the jobs. There are two types of dynamic mapping: batch mode and online mode [2]. In batch mode tasks are gathered into a set, and this set is analysed for resources to be assigned before scheduling whereas in the online mode the task is assigned immediately when it arrives the mapper.

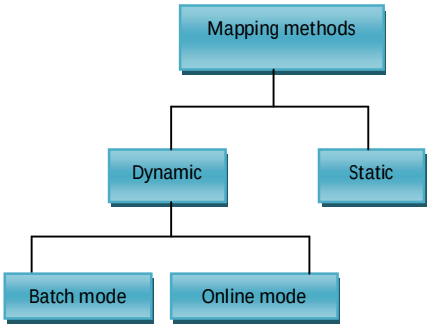

Fig. 2 showing the hierarchy of mapping methods

2.3 Online and Batch mode Heuristics

Online and batch scheduling are well-known methods, largely explored in distributed computing [4]. They are also useful for task scheduling. In online mode, jobs are scheduled as soon as they enter the system, without waiting for the next time interval when the scheduler will get activated or the job arrival rate is small having thus available resources to execute jobs immediately. In batch mode, tasks are grouped in batches and scheduled as a group. In contrast to online mode scheduling, batch mode scheduling could take better advantage of job and resource characteristics in deciding which job to map to which resource since they dispose of the time interval between two successive activations of the scheduler.

2.4 Objectives of Grid Scheduling

The task scheduling problem is multi-objective in nature. Several performance measures and optimization criteria can be considered to evaluate the quality of a given schedule and overall grid system performance [1]. Following are the important objectives.

2.4.1 Makespan

Most popular optimization criterion is minimization of makespan i.e. the finishing time of the latest job. Makespan measures the throughput of grid system [1].

2.4.2 Flow time

Flow-time is the sum of the finishing times of jobs. Flow time measures the Quality of Service of the grid system. Flow time is minimum when jobs are processed in ascending order of processing time on a particular grid resource [1].

2.4.3 Resource utilization

Due to economic aspect resource providers and grid managers are interested in the maximum utilization of resource. Resource utilization is defined as the degree of utilization of resources with respect to the schedule. The resource utilization is defined using the completion time of a machine, which indicates the time at which machine m will finalise the processing of the previous assigned jobs as well as those already planned for the machine [1].

2.4.4 Matching Proximity

In grid computing effort is made to process the task on the best possible machine i.e. the machine which takes minimum execution time. Matching Proximity indicates the degree of proximity of a given schedule to the schedule produced by the Minimum Execution Time (MET) method, which assigns a job to the machine having the smallest execution time for that job. Matching proximity is an additional performance parameter of batch mode methods. A large value for matching proximity means that a large number of jobs are assigned to the machine that executes them faster [1].

2.4.5 Computation Time

Due to dynamic nature of grid, computation time needed to generate schedule is also an important criterion for selecting a suitable scheduling method. In task scheduling problem there is no need to get the optimal solutions. In the highly dynamic environment it is essential to get high quality feasible solution in short time. Computation time of the order of 1 micro second is measured for different heuristics [1].

2.4.6 Load Balancing

This objective is related to balance the load on the machines which are used for the task scheduling. Because if the load imbalance occurs at the machines then task scheduling process will not be appropriate to obtain the optimum solution.

2.5 Batch mode heuristics

The following are the important batch mode heuristics which are used for the task scheduling problem.

2.5.1 Min-Min Heuristic

Min-Min algorithm finds the task which has minimum execution time and assigns the task to the resource that produces minimum completion time. The ready time of resource is updated [1]. This procedure is repeatedly executed until all tasks are scheduled. The Min–Min scheduling algorithm chooses the smaller task first. There are some drawbacks if the smaller number of tasks exceeds the larger one.

Note: Problem is solvable using Min-min in O (n^2m) time.

2.5.2 Max-Min Heuristic

The Max-min scheduling algorithm is similar to Min-min scheduling algorithm but it schedules the larger task first. The ready time of resource is updated. This process is repeated until all unmapped tasks are assigned. Min-min and Max-min are used for small scale distributed system [1]. This method is appropriate when most of the jobs arriving in the grid system are short ones. Thus, Max-Min would try to schedule at the same time all the short jobs and the longest ones while Min-Min would schedule first the shortest jobs and after that the longest ones, implying thus a larger makespan.

Note: Heuristic Max-min can solve problem in O (n^2m) time.

2.5.3 Suffrage

Suffrage for a task is the difference between second minimum completion time and first minimum completion time for that task. Suffrage method tries to allocate most suffered tasks in terms of expected completion time first. In this method, suffrage is calculated for all unassigned tasks and the task which has maximum suffrage value is assigned to the resource which gives first minimum completion time [1]. Then task is removed from unassigned task list, resource workload is updated and above cycle of task allocation is repeated till list of unassigned tasks gets exhausted. Suffrage heuristic is based on the idea that better mappings can be generated by assigning a machine to a task that would "suffer" most in terms of expected completion time if that particular machine is not assigned to it.

Note: Problem is solvable using Suffrage in O (n^2m) time.

2.5.4 LJFR-SJFR Heuristic

Largest Job on Fastest Resource – Shortest Job on Fastest Resource (LJFR-SJFR) method allocates largest job on fastest resource to reduce the makespan and allocates smallest job to fastest resource to reduce the flow time of the schedule. In first stage the algorithm allocates n number of jobs to m number of machines similar to Max-min i.e. completion time of all unassigned tasks on all the available machines is used to calculate the minimum completion time (MCT) of a task on a particular machine. Then task which gives maximum of (MCT) is identified and is assigned to the corresponding machine [1]. Subsequently the task is removed from the list of unassigned tasks and workload of that machine is updated. In this manner n numbers of jobs are assigned to m number of unallocated machines. In second stage remaining unassigned jobs are assigned alternatively using min-min and max-min method i.e. smallest job on fastest resource followed by largest job on fastest resource till the list of unassigned jobs gets exhausted.

Note: Heuristic LJFR-SJFR can solve a problem in O (n^2m) time.

2.5.5 Relative Cost (RC) Heuristic

While assigning jobs relative cost method considers both the load balancing of machines and the execution time of jobs on machines. Method calculates two parameters static relative cost and dynamic relative cost [1]. Static relative cost is computed only once at the start of the method, on the other hand, the dynamic relative cost is computed at the start of each iteration k.

2.5.6 Min-Max Heuristic

Min-max heuristic has two steps for assigning jobs to machines. In the first step, similar to min-min method, completion time of all unassigned tasks on all the available machines is used to calculate the minimum completion time (MCT) of a task on a particular machine. In the second step, for all tasks, ratio (pf) of minimum execution time (time to execute on fastest machine) to its execution time on selected machine is computed and the task which has maximum value of pf is selected for assignment [1]. Then job is removed

from unassigned job list, machine workload is updated and above cycle of job allocation is repeated till list of unassigned jobs gets exhausted.

Note: The idea behind this heuristic is that we select pair of machines and tasks from the first step so that the machine can execute its corresponding task effectively with a lower execution time in comparison with the other machines.

2.6 Online mode heuristics

Following are some of the important online mode heuristics. As discussed above in online mode, jobs are scheduled as soon as they enter the system.

2.6.1 Opportunistic Load Balancing (OLB) Heuristic

In this method earliest idle machine is selected without considering the job's execution time on the selected machine. If two or more machines are idle then machine is selected arbitrarily. In this method time required for scheduling is less and it keeps almost all the machines busy at all possible time. Resulting schedule is not optimal [1]. One advantage of this method is that it tries to keep the machines as loaded as possible; however, the method is not aware of the execution times of jobs in machines, which is, certainly, a disadvantage regarding the makespan, flow time and matching proximity parameters. The opportunistic algorithm takes into account the dynamic characteristics of Grid environments without the need to probe the remote sites. The Opportunistic algorithm benefits from the dynamic aspects of the Grid environment. If a site happens to perform poorly, then the number of jobs assigned to this site decreases. Similarly, if a site processes the jobs quickly, then more jobs are scheduled to that site.

Note: Problem is solvable using OLB in O (nm) time.

2.6.2 Minimum Completion Time (MCT) Heuristic

This algorithm finds the machine which has Minimum Completion Time for the particular task. It assigns the task to resources based on completion time. Completion time is calculated by adding the execution time and the ready time of the resource. Allocating task in this manner may result in execution of task on less fast grid resources [1]. The idea behind MCT is to combine the benefits of opportunistic load balancing (OLB) and MET, while avoiding the circumstances in which OLB and MET perform poorly.

Note: Problem is solvable using MCT in O (nm) time.

2.6.3 Minimum Execution Time (MET) Heuristic

This algorithm finds the task which has minimum execution time i.e. the task is assigned to the resource (on FCFS basis) on which it can be executed in minimum time. Unlike MCT method, MET does not take into account the ready times of machines. One of the main drawbacks of this algorithm is load imbalance [1]. It does not consider the availability of the resource and its load.

The motivation behind MET is to give each task its best machine. This can cause a severe load imbalance among machines. Even worse, this heuristic is not applicable to heterogeneous computing environments where resources and tasks are characterized as consistent, which means a machine that can run a task faster will run all the other tasks faster.

Note: Heuristic MET can solve a problem in O (nm) time.

2.6.4 Switching Algorithm (SA) Heuristic

This method of scheduling combines the best features of MCT and MET methods of scheduling. The method tries to use better load balancing of MCT and execution on fastest machine of MET. Here the idea is to first use the MCT till a threshold of balance is reached followed by MET which creates the load unbalance by assigning jobs on faster machines [1]. Here MCT and MET are used in cyclic manner.

Note: Problem is solvable using this algorithm in O (nm) time.

2.6.5 k-Percent Best (kPB)

This method also attempts to combine the best features of MCT and MET simultaneously instead of cyclic manner. In this method only k percentage of best resources, on the basis of execution time, are considered while assigning the jobs [1]. For a particular job a resource which gives minimum completion time is selected out of the k percent best resources instead of all possible resources. For k=100 this method acts similar to MCT while for

k=100/total number of machines this method acts similar to MET.

Note: Here kPB has serious drawback that in a situation when k percentage resources are busy but other resources are free, even in such situation kPB allocates job only to one of the busy k percent best resource. As a result there is large idle time of resources in the generated schedule.

2.6.6 Work Queue (WQ)

It is a very simple method of job allocation. Jobs are randomly selected from the list of unassigned jobs and assigned to the machine which has minimum workload [1]. Job assignment is repeated in similar manner till list of unassigned jobs gets exhausted.

3. NEED FOR QOS BASED TASK SCHEDULING APPROACH

3.1 Scenario 1

QoS is an extensive concept. It means different to different applications [7]. In computing-intensive tasks, QoS usually indicates the operation speed, while in data-intensive tasks, QoS often represents bandwidth, quality of node data. But in most of the cases, QoS parameters are detailed as network bandwidth. Since tasks also require QoS of hosts, tasks with high QoS require the hosts that are able to provide high QoS guarantee. Under such circumstances, traditional algorithms may lead to the phenomenon i.e. the certain tasks which are with no special requirement of host QoS may be allocated to the host with high QoS guarantee. This phenomenon delays makespan, wastes resources and unbalances payloads, thus reducing overall performance of Grid system.

3.2 Scenario 2

There are two types of jobs in Grid, computation based and communicational based [3]. The communication based jobs for example 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 are submitted to high bandwidth resource then it will not utilize its bandwidth effectively, similarly if the communication based jobs are submitted to a resource having high speed CPU and low bandwidth then it does not fully utilize the resource and also increases its job completion time.

3.3 Scenario 3

Classical Grid scheduling algorithms such as Max-Min, Min-Min and Suffrage do not consider the influence brought by QoS of tasks [7]. And in some systems that involve QoS, most of them adopt QoS in the level of resource management and network, without considering QoS in task and computer level.

3.4 Scenario 4

Now a day's user needs guaranteed quality of service [5]. User's demand and evaluation performance are not being satisfied with the traditional type of grid scheduler. Lot of research was carried out in this field and the result is QoS based task scheduling. It is a new issue in grid scheduling algorithm. It is a quite complicated QoS based grid scheduling which considers the QoS parameter of both the tasks and the resources before scheduling. CPU speed, memory, bandwidth are some of the QoS parameter in QoS based task scheduling. QoS aware Grid scheduler will allocate a task to the resource only if both task and resource QoS requirements are satisfied [6]. QoS aware scheduler must consider the QoS requirements of the resources, QoS requirements of tasks, and minimize the makespan simultaneously.

Note: Thus from the above mentioned four scenarios the need of the QoS based approach is an obvious choice.

4. SIGNIFICANCE OF QOS BASED TASK SCHEDULING APPROACH

Significance of the QoS based task scheduling approach lies in the fact that there are many QOS based heuristics are available which are capable to achieve the maximum objectives as compared to the existing batch mode and online mode heuristics. In this paper some important QoS based heuristics are discussed which are given below.

4.1 QoS priority grouping heuristic

This heuristic is based on the number of computing resources with which the task can run. There are n hosts in the Grid environment therefore all tasks in Grid can be grouped into n groups [7]. With the descendent order from high to low of QoS priority, the tasks in different QoS level are

scheduled by suffrage algorithm independently. Thus this algorithm yields better results in makespan and task accepted performance.

4.2 QOS Suffrage heuristic for independent task scheduling in Grid

Reduction in makespan is a fundamental objective of optimizing task scheduling problems in distributed heterogeneous computing systems. So, QoS Suffrage heuristic is adapted to include network bandwidth as Quality of Service (QoS) parameter [6]. This algorithm is based on single-tier case for QoS and gives better performance gain in a variety of applications.

4.3 QoS Guided Min-Min task scheduling heuristic

Traditional Min-Min heuristic does not consider QoS, which effects its effectiveness in a grid. QoS Guided Min-Min adds a QoS constraint (QoS for a network by its bandwidth) to basic Min-Min heuristic [5]. Its basic idea is that some tasks may require high network bandwidth, whereas others can be satisfied with low network bandwidth, so it assigns tasks with high QoS request first according to Min-Min heuristic. In the worst cases, where all tasks need either low QoS or high QoS, this heuristic will take O (n^2m) time

4.4 QoS guided weighted mean time min (QWMTM) heuristic

The algorithm first divides the tasks into two groups: high and low QoS. In high QoS group the task with high QoS demands are taken. In low QoS group, tasks with low or no QoS demands are taken. This heuristic first schedules tasks from the high QoS group and then the tasks from low QoS group. For each task in high QoS group, the heuristic calculates the weighted mean time. It selects the task t_i with maximum weighted mean time for mapping. Then it finds the resource r_j, from QoS qualified resource set that gives the earliest completion time [5] for the task t_i. It maps the task t_i on resource r_j. It updatesthe ready time of resource r_j. and the task t_i is deleted from the task set. The process will continue till all the tasks are mapped. After mapping all high QoS tasks, the algorithm maps the tasks from low QoS group.

4.5 QoS priority based scheduling heuristic (QSPBS)

This algorithm assigns the high priority jobs to resources first and then it assigns the low priority jobs for scheduling. It assigns the jobs to the resources based on their complexity. High complexity jobs are allocated to high speed system and low complexity jobs are allocated to hybrid system. So the makespan and resource utilization for all the tasks is improved. It assigns tasks to all resources equally, so the load balancing problem is solved in this algorithm. Thus this algorithm gives optimum solution for task scheduling in Grid. This algorithm is being widely used in the present scenario because it tends to take into account all the important characteristics of both the tasks and the resources before scheduling. Also the complexity parameter which has been added to this heuristic helps it in achieving the best possible solution for the given application [5]. Apart from satisfying these three objectives this heuristic can be improved further to satisfy even more objectives.

TABLE 1

Summary of Heuristics With Their Objectives

Heuristics	Objectives		
	Makespan	Resource Utilization	Load Balancing
MET	Yes	No	No
MCT	Yes	No	Yes
Min-Min	Yes	No	No
Max-Min	Yes	Yes	No
QoS priority based scheduling heuristic	Yes	Yes	Yes
QoS Guided Min-Min heuristic	Yes	Yes	Yes
QoS priority grouping scheduling heuristic	Yes	Yes	Yes
QoS guided weighted mean heuristic	Yes	Yes	Yes
Qos Suffrage heuristic	Yes	Yes	Yes

Note: In the above Table the terms “Yes” and “No” are used to indicate whether a particular heuristic satisfies an objective or not.

5. CONCLUSION

In this paper first of all we discussed the existing online mode and batch mode heuristics and we found that the existing heuristics are not able to achieve the maximum objectives and then we discussed some of the important QoS based heuristics. From the discussion of both types of heuristics we concluded that QoS based heuristics are indeed better than the existing heuristics in achieving the maximum objectives. Also, we can see that so far most of QoS based heuristics outperforms the existing heuristics in terms of objectives i.e. makespan, resource utilization and load balancing only. In the future we can explore other important objectives and QoS based heuristics and can show that QoS based heuristics always perform better in achieving those objectives as compared to existing task scheduling heuristics.

REFERENCES

Journals Papers:

[1] Rajendra Sahu and Anand K Chaturvedi, ABV-IIITM Gwalior, India, *"Many-Objective Comparison of Twelve Grid Scheduling Heuristics", International Journal of Computer Applications (0975 – 8887),Volume 13– No.6,* January 2011.

[2] Adil Yousif, Abdul Hanan Abdullah, Muhammad Shafie Abd Latiff and Mohammed Bakri Bashir, *"A Taxonomy of Grid Resource Selection Mechanism,"* Faculty of Computer Science &Information System, Universiti Teknologi Malaysia, *International Journal of Grid and Distributed Computing, Vol. 4, No. 3, September, 2011*

[3] Dr. Rajesh Kumar Bawa, Deptt. of Computer Science, Punjabi University Patiala, Punjab, India, Mr. Gaurav Sharma, Deptt. of Computer Engineering, JMIT Radaur, Kurukshetra University ,Haryana, India, *"Various Resource Selection Heuristics in Grid"*.

[4] Fatos Xhafa, Department of Computer Science and Information Systems, Birkbeck, University of London, UK, Ajith Abrahamb, Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, USA, *"Computational models and heuristic methods for Grid scheduling problems ",Future Generation Computer Systems 26 (2010) 608-621,journal homepage: www.elsevier.com/locate/fgcs.*

[5] T Amudha (Assistant Professor), T T Dhivyaprabha (MPhil Research Scholar),*"QoS Priority Based Scheduling Algorithm and Proposed Framework for Task Scheduling in a Grid Environment"*, Department of Computer Application ,School of Computer Science & Engg, Bharathiar University, Coimbatore – 46,*IEEEs-International Conference on Recent Trends in Information Technology, ICRTIT 2011*, MIT, Anna University, Chennai. June 3-5, 2011

[6] Ehsan Ullah Munir, Jianzhong Li and Shengfei Shi, School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China, *"QoS Suffrage Heuristic for Independent Task Scheduling in Grid"*.

Proceeding Paper:

[7] Fang Dong, Junzhou Luo, Lisha Gao, Liang Ge, School of Computer Science and Engineering, Southeast University, Nanjing 210096, P.R.China,*"A Grid Task Scheduling Algorithm Based on QoS Priority Grouping", Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06),0-7695-2694-2/06 $20.00 © 2006*