Modeling and Simulation of Grid Resource Brokering Algorithms

Aditya B. Patel,
AES Institute of Computer Studies,
Ahmedabad, India

ABSTRACT

Grid Computing is concerned with applying the heteregenous resources of many computers to solve a single problem and involves managing the diverse resources towards a common objective. Successful utilization of grid infrastructure to solve resource intensive and computing problems requires performance modeling and evaluation to meet the QoS requirements of end users. Resource management and scheduling is the most important component of grid systems. A Grid scheduler 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. Grid Resource broker or meta-scheduler uses local schedulers of the different grid middleware and local schedulers of clusters to allocate jobs to distributed resources. To address different issues in grid scheduling, different scheduling approaches and algorithms have been proposed in the literature. However, evaluation and comparative analysis of these algorithms and research experiments are often difficult to perform due to problems like large number of heterogeneous resources, dynamic nature of grid, not able to create different types of realistic workloads and jobs with different parameters and lack of certain functionalities in available resource management systems like advance reservation (AR) and grid usage accounting. This paper focuses on modeling and simulation of grid towards achieving various grid performance metrics and QoS requirements. This paper also presents the experimental results on grid simulation and performance evaluation different grid resource brokering approaches using synthetic workloads. Different simulation experiments are used to compare different aspects of scheduling using different types of grid job, input resources and workloads. The simulation results indicate the effect of resource brokering approach used on efficient execution of grid jobs, success ratio of jobs with QoS requirement, resource utilization and load balancing of grid system.

General Terms

Grid Resource Brokering, Quality of Service, Grid Simulation and Modeling

Keywords

Grid Resource Brokering; Grid Scheduling Evaluation; Grid Simulation; Synthetic Workload;

1. INTRODUCTION

With the increase in complexity and resource requirements of scientific and business applications, the collective usage of distributed resources has seen an increasing trend over the last decade. As a result, the focus for distributed computing has shifted from using homogeneous resources (clusters) to using remote heterogeneous resources using grid and cloud computing paradigm. Grid and cloud computing have become a widely used platforms for solving large scale resource intensive and computing tasks. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1]. Resource management and scheduling is the most crucial, complex and central part of any grid implementation. A Grid scheduler or resource 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. Grid resource broker must communicate with local resource managers for solving computational problems submitted by end users to utilize collective resources available in the grid system. Besides that, performance evaluation and comparison require the existence of workload traces within a grid, which at the moment simply do not exist. We assume that all three approaches, simulation, emulation, and real system testing, are of significance in their domain. Thus, a performance evaluation system should ideally (a) support all of them and (b) allow a comparison of results on a technical level.

2. GRID RESOURCE BROKERING

Grid resource brokering involves converting application requirements into resource requirement, resource discovery, job submission, resource matchmaking with application requirements, resource negotiation and allocation based on QoS requirement, scheduling and co-allocation, resource monitoring, resource reservation capability and performance prediction capability. The scheduling algorithm used by the grid system is very important to achieve key performance metrics of grid system like job execution time, maximize system performance and improve load balance of the grid system. At the same time, various issues and challenges to be addressed by grid scheduler are heterogeneity, scalability, dynamicity and autonomy of distributed grid resources. To address different issues in grid scheduling, different scheduling algorithms have been proposed in the literature. However, evaluation and comparative analysis of these algorithms and research experiments are often difficult to perform due to problems like not able to access large number

of heterogeneous resources, not able to create different types of realistic workloads and jobs with different parameters and lack of certain functionalities in available resource management systems like advance reservation (AR) and grid usage accounting.

3. GRID RESOURCE BROKERING ARCHITECTURE

The Resource Broker provides an abstraction to the complexity of Grids by ensuring transparent access to computational resources for executing a computational job on a grid. The resource broker enables the selection of best available resources from the providers for the execution of a specific task. These resource brokers collect information (e.g., resource availability, usage models, capabilities and pricing information) from the respective resources and use this information source in the scheduling process. Most distributed resource clusters have a local resource management system to manage and allocate resources (e.g. SGE, Condor, PBS/Torque, LSF). It uses user requirements to create a set of jobs, discover resources, schedule, execute and monitor the jobs and retrieve their output once they are finished. The broker interfaces with various grid middleware and local resource management systems and select suitable resources in order to meet the application requirements.

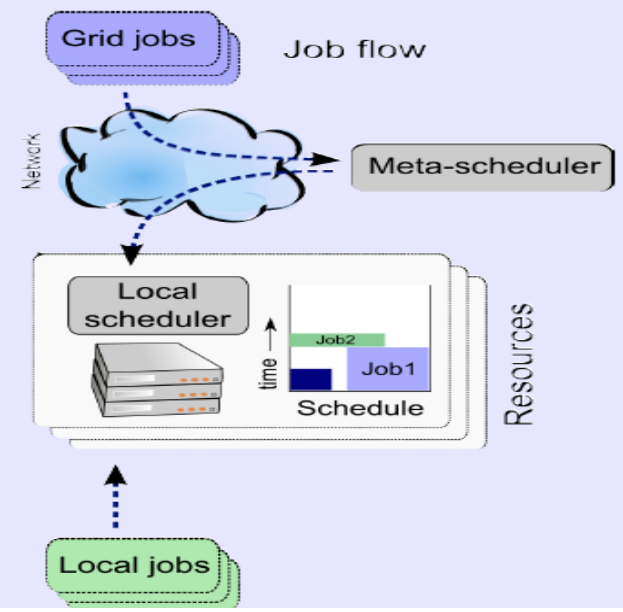

Figure 1: Grid Resource Brokering architecture

4. GRID RESOURCE BROKERING ALGORITHMS

This paper evaluates different widely used grid scheduling approaches [2] used in various grid implementations. These algorithms are implemented in simulated grid environment using GSSIM grid simulator [3].

4.1 Backfilling

Backfilling is conventional and widely used approach for scheduling parallel tasks efficiently in distributed systems. In systems of parallel tasks, the job at the head of the queue is often delayed because of a lack of available resources. However, there are often less demanding jobs in the queue which could be executed earlier without delaying the job at the head. EASY backfilling follows this strategy**.** If the resources for the job at the head of the queue are expected to become available at time t, then the easy backfilling strategy allows a job with expected length l, where l < t, to be executed immediately, rather than waiting, assuming its required resources are available. This effectively fills the idle bubbles in the schedule and results in a shorter overall completion time.

4.2 Heuristic Algorithms for NP-hard problem (Non-deterministic polynomial time)

In general, the problem of mapping tasks on distributed resources belongs to a class of problems known as NP-hard problems. For such problems, no known algorithms are able to generate the optimal solution within polynomial time. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. Many heuristic algorithms have been proposed to solve the problem like Min-min, Max-min and suffrage. We consider Min-Min heuristic in this study.

4.2.1 Heuristic Algorithm

The expected execution time ETij of task ti on machine mj is defined as the amount of time taken by mj to execute ti given that mj has no load when ti is assigned.

The expected completion time CTij of task ti on machine mj is defined as the wall-clock time at which mj completes ti (after having finished any previously assigned tasks).

dj denotes the delay time or next available time of host mj , which is also the beginning time of next ti that is to be executed on mj.

The completion time of task ti to be executed on machine mj is defined as

$$CTij = dtij + ETij \quad (1)$$

Where dtij is the delay time of task ti on machine j and machine j is assigned to execute task i. Makespan is a measure of the throughput of the heterogeneous computing system.
The throughput (makespan) for the complete schedule is then defined as Max (CTij).

The MCT (minimum completion time) heuristic assigns each task to the machine so that the task will have the earliest completion time. The MET (minimum execution time) heuristic assigns each task to the machine that performs that task's computation in the least amount of execution time. The Min-Min algorithm for scheduling m jobs on n machines is as below [4]. The algorithm iteratively assigns jobs to machines by considering jobs not yet scheduled and computing their expected Minimum Completion Time (MCTs). In the first phase, the set of minimum expected completion time (such that job has the earliest expected completion time on the corresponding machine) for each job is found. In the second phase, the job with the overall minimum expected completion time from set is chosen and assigned to the corresponding resource. Then this job is removed from set and the process is repeated until all jobs in the set are mapped.

```
while there are jobs to schedule
        for all job i to schedule
                for all machine j
```

```
            Compute CTij = CT (job i, machine j)
            end for
            Compute minimum CTij
        end for
        Select best metric match m
        Compute minimum CTmn
        Schedule job m on machine n
end while
```

Figure 2: Heuristic Algorithm (Min-Min)

For each task, a metric is computed using their MCTs and the task with the best metric is assigned to the host that let it achieve its MCT. If there are *m* jobs to be scheduled in *n* machines, the time complexity of Min-Min algorithm is $O(m^2n)$. For computation time prediction, linear model of estimation based on host processing power, current load and historical information available in Grid Information Service (MDS) is used.

4.2.2 Drawbacks of Heuristic algorithm (Min-Min)

The Min-Min algorithm does not take into account the QoS issue in the scheduling. In some situation, it is possible that normal tasks occupies machine that has high QoS resources. This may increase the delay of QoS tasks or result idle of normal machines and uneven load balancing in the system.

4.3 QoS Oriented Scheduling

To solve the problems with heuristic algorithms and provide differentiated services to different end users, QoS oriented scheduling approach [5] is used. The objective of the QoS oriented scheduling is to fulfill the user QoS constraints and maximize the system utilization. The scheduling algorithm depends on the prediction system which calculates estimations time of jobs on resources based on historical information that contain traces of previous job submissions.

Input: set of jobs in JobQueue with QoS attributes like resource requirement, budget and deadline

Output:schedule and CT of all jobs

Contact Grid Information Service (GIS) to obtain a list of available resources with associated QoS parameters and create ResourceList

```
for all jobs i in JobQueue
        for all hosts mj in ResourceList
        CTij = ETij + dj
        end for
end for

Sort the Jobs in JobQueue using the earliest due date (EDD) policy
do until all jobs with QoS request in JobQueue are mapped
        for each job with QoS in JobQueue, find a host in the QoS qualified
        host set that matches the QoS attributes and obtains the earliest completion time
        (QoS attribute can include request for advance resource reservation)
        end for
        find the job Ji with the minimum earliest completion time (min(CTij))
        if(min(CTij) < DeadlineJobi)
                Reschedule:
                if host mj is having a Job with no QoS , reschedule it and calculate revised CTij
                if (revised CTij < DeadlineJobi)
                        assign jobi to host mj with next min(CTij) and perform reschedule
                else
                        assign jobi to host with next min(CTij) and perform reschedule

        assign job Ji to the host mj that gives it the earliest completion time and satisfies deadline
        delete job Ji from JobQueue
        update dj = dj+CTij
        update CTij for all i
end do

do until all jobs with non-QoS request in JobQueue are mapped
        for each job in JobQueue
        find the earliest completion time and the corresponding host
        find the jobi with the minimum earliest completion time
        end for
        assign jobi to the host mj that gives it the earliest completion time
        delete jobi from JobQueue
        update dj
        update CTij for all i
end do


MoniterJob
Input: Job Ji, Mj, JobQueue
Output: Updated JobQueue and ResourceList
do
        Monitor the status of job execution
        If (JobStatus = = completed)
                Delete the job Ji from JobQueue
                return
        Else if (JobStatus = failed)
                Reschedule the job
                return
end do
```

Figure 3: QoS oriented Algorithm

The QoS oriented approach computes the expected completion time of all the jobs on all the hosts at the start using linear prediction model of estimation based on host processing power, current load and historical information available in Grid Information Service (MDS) is used. Then, instead of mapping the whole JobQueue to the hosts, it maps the tasks with high QoS request first. In the first "do" loop, initially, for each task with high QoS request in the TaskQueue, the algorithm finds the earliest completion time and the host that obtains it, in all the QoS qualified host sets. The algorithm finalizes the loop by deleting the scheduled job from the JobQueue, and updating dj and CTij for all i (hosts). If there are m jobs to be scheduled on n machines, the time complexity of the algorithm is $O(m^2n)$.

5. GRID SCHEDULING SIMULATION

The Grid Scheduling Simulator (GSSIM) provides an easy-to-use Grid scheduling framework for enabling simulations of a wide range of scheduling algorithms in multi-level, heterogeneous grid infrastructures. GSSIM is built on top of GridSim. GridSim [6] is a Java based grid simulation toolkit for modeling and simulation of distributed grid applications. It supports modeling and simulation of heterogeneous time and space-shared Grid resources, application models and QoS in Grid environment.

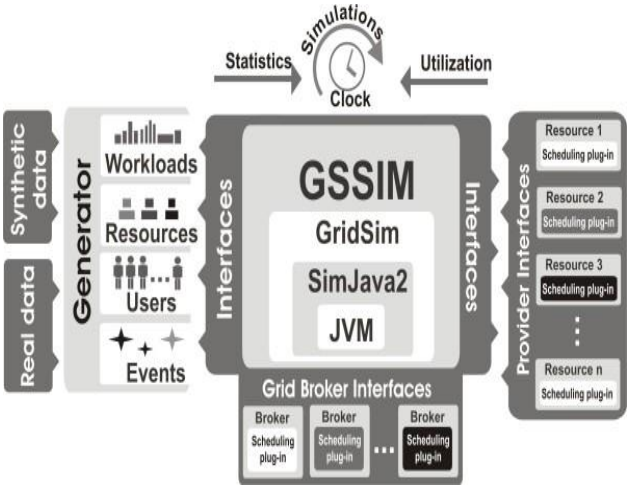

Figure 4: GSSIM architecture

It also enables both reading existing real workloads and generating synthetic Grid workloads based on given probabilistic distributions and constraints. These workloads are compliant with workload formats such as Standard Workload Format (SWF) [7] and Grid Workload Format (GWF) [8]. Different scheduling approach algorithms are implemented in simulated grid environment using GSSIM grid simulator. Different types of grid scheduling algorithms that are compared are Backfilling, Heuristic based algorithm and QoS based algorithm. Different simulation experiments are used to compare different aspects of scheduling using different models of job, resources and workloads. The key performance metrics considered for scheduling experiments are makespan, job execution time and resource utilization of grid system.

5.1 Input Data Modeling

In general, input data in GSSIM consist of workload and resource descriptions. Workload contains information about jobs, their structure, resource requirements, relationships, time intervals etc. It is assumed that there is a queue of jobs submitted to a Grid scheduler.

5.1.1 Synthetic Workload description

The workload generation file is used in the scheduling experiments. Workload contains information about jobs, their structure, resource requirements, relationships, time intervals etc. Use of synthetic workloads allows researchers to make repeatable experiments and refer to specific workloads used in their research.

Synthetic workload generation file (workload.swf)

```
<WorkloadConfiguration>
        <JobCount count=”100” />
        <TaskCount                              avg=”1”
        distribution=”constant” />
        <TaskLength        avg=”1500”        dev=”500”
distribution=”normal” />
        <TaskInterval avg=”50” distribution=”poisson” />
        <Parameter metric=”cpucount” min=”1” max=”4”
distribution=”uniform” />
</WorkloadConfiguration>
```

According to the synthetic workload configuration, different jobs are generated using a uniform distribution.

GrmsJob1.xml

```
<grmsJob appId="1">
<task taskId="0">
        <requirements>
                <resourceRequirements>
                <computingResource>
                <hostParameter name="cpucount">
                <value>1</value>
                </hostParameter>
                </computingResource>
                </resourceRequirements>
        </requirements>
        <executionTime>

        <executionDuration>PT2H</executionDuration>
                <timePeriod>
                <periodStart>2008-11-
03T01:00:00.000+01:00</periodStart>
                <periodEnd>2008-11-
03T03:00:00.000+01:00</periodEnd>
                </timePeriod>
        </executionTime>
</task>
</grmsJob>
```

5.1.2 Resources

The second part of the input data that must be delivered for simulations is the description of resources. Each description contains information about resource parameters, e.g. number of free CPUs, memory, etc. Finally, availability of AR mechanism is indicated in the description.

ResourceParameters.xml

```
<hostParameters>
<resources>
        <computingResource resourceId="compRes1">
                <machineParameters>
                <hostParameter name="cpucount">
                <paramValue>3</paramValue>
                </hostParameter>
                <otherParameter name="reservation">
                <paramValue>true</paramValue>
                </otherParameter>
                </machineParameters>
        </computingResource>
        <computingResource resourceId="compRes2">
                <machineParameters>
                        <hostParameter
                        name="cpucount">
                        <paramValue>3</paramValue>
                        </hostParameter>
                        <otherParameter
                        name="reservation">
                        <paramValue>true</paramValu
                        e>
                        </otherParameter>
                </machineParameters>
        </computingResource>
</resources>
```

</hostParameters>

Each description contains information about resource parameters, e.g. number of free CPUs, memory, etc. Finally, availability of AR mechanism is indicated in the description.

6. GRID SIMULATION RESULTS

Different types of grid jobs generated using synthetic workload are submitted to simulator and execution results obtained are compared using performance metrics considered. We analyze the results to explain how different scheduling approaches affect the performance of grid system. The execution results indicate the effect of scheduling approach, number of tasks, QoS requirement on performance metrics and overall grid system.

Table 1: Parameters and Performance Metrics

Nt	Number of synthetic tasks generated by workload parameters
Nr	Number of resources with attributes
QoSt%	Percentage of total synthetic tasks with explicit QoS like hard deadline or advance resource reservation requirement
Makespan	The completion time of a set of synthetic tasks generated
Resource Utilization	Used time v/s idle time of resources
Success ratio	Percentage of QoS jobs completing successfully

6.1 Comparison of makespan with increase in number of tasks

Simulation results (Figure 5) indicate that heuristic and QoS approach improves job execution efficiency and reduces the makespan compared to backfilling algorithm.

Experiment Parameters: Nr=10, QoSj=15%

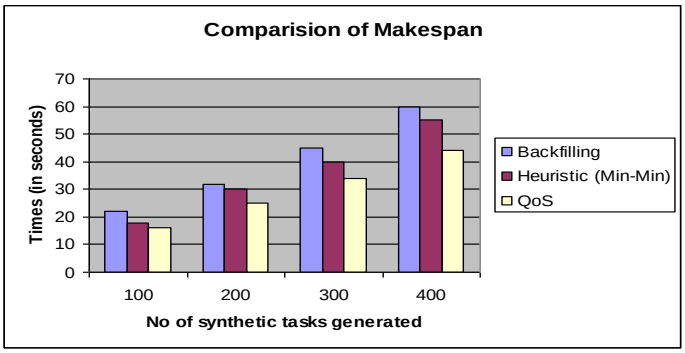

Figure 5: Comparison of makespan of different scheduling approach

6.2 Comparision of success ratio of QoS jobs with deadline

Experiment Parameters: Nt=100, Nr=10

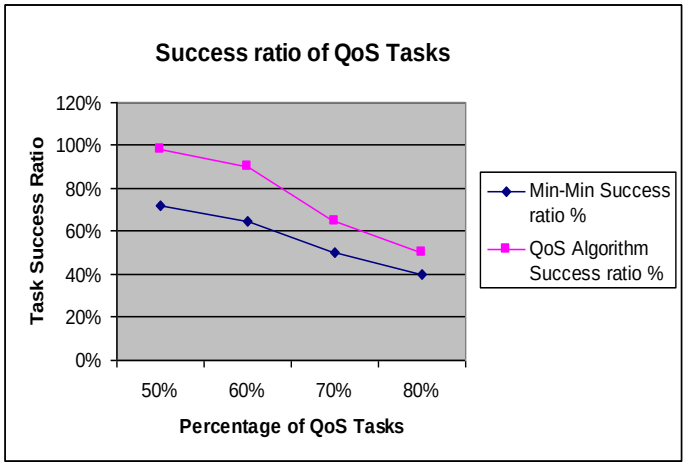

Figure 6: Comparison of success ratio of QoS jobs with deadline

Figure 6 indicates that the QoS algorithm show improvement upto 30% in increased number of successful jobs with deadline, provided the number of QoS jobs are less than 60% of total jobs submitted.

6.3 Effect of Scheduling Approach on Resource Utilization

Under heuristic approach, it is possible for low QoS tasks to occupy high QoS resources while high QoS tasks wait as low QoS resources remain idle. Figure 7 indicates that with the increase in tasks with hard QoS requirement like deadline and resource reservation, the resource utilization of the grid system decreases. QoS algorithm leads to improvement in resource utilization and better load balancing of grid system.

Experiment Parameters: Nt=100, Nr=10

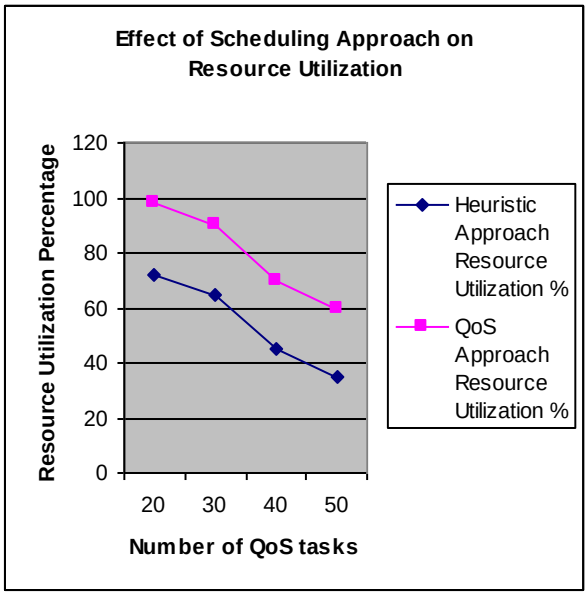

Figure 7: Effect of Scheduling Approach on Resource Utilization

7. CONCLUSION AND FUTURE WORK

This paper discusses experimental work using empirical results on performance evaluation and comparison of different grid resource brokering approaches using synthetic workloads and their impact on grid system performance. Different scheduling approach algorithms are implemented in simulated grid environment using GSSIM grid simulator. GSSIM helps in performance evaluation of scheduling algorithms using synthetically generated workload which is difficult in realistic grid environments. Different simulation experiments are performed to compare different aspects of scheduling using different types of job, resources and workloads. The execution results indicate the effect of scheduling approach used on efficient execution of grid jobs, success ratio of jobs with QoS requirement and resource utilization and load balancing of grid system. Simulation results indicate the need for user centric scheduling approach providing differentiated services to different users and support for service level agreement (SLA) negotiations between users and the resource providers of distributed systems.

8. REFERENCES

[1] Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure Second edition, Morgan-Kaufman, 2004.

[2] Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", Technical Report No. 2006-504, 2006.

[3] The Grid Scheduling Simulations Portal, http://www.gssim.org.

[4] Jinquan Z, Lina N, Changjun J, A Heuristic Scheduling Strategy for Independent Tasks on Grid, Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA '05), November 2005.

[5] Ching-Hsien Hsu, Zhan, J., Wai-Chi Fang, Jianhua Ma, Towards Improving QoS-Guided Scheduling in Grids, IEEE Third ChinaGrid Annual Conference, 2008.

[6] Buyya R., Murshed M., GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing Concurrency and Computation: Practice and Experience 2002:14 (13-15): 1175-1220.

[7] Parallel Workload Archive, http://www.cs.huji.ac.il/labs/parallel/workload.

[8] Grid Workload Archive, http://gwa.ewi.tudelft.nl.