GRID META BROKER SELECTION STRATEGIES FOR JOB RESERVATION AND BIDDING

D. Ramyachitra[#1], S. Poongodi[#2]

[#1] Asst.Prof , Department of Computer Science,
Bharathiar University, Coimbatore- 46.
[#1] jaichitra1@yahoo.co.in
[#2] M.Phil Scholar, Department of Computer Science
Bharathiar University, Coimbatore- 46.
[#2] poongodimsc@gmail.com

ABSTRACT

Grid computing is applying the resources of many computers in a network to a single problem at the same time. In scientific or technical problem it requires a great number of computer processing cycles or access to large amounts of data. In this paper we depict and evaluate broker selection strategies for job reservation and bidding. Especially, this paper analyzed two different types of existing algorithms simple and categorized aggregation algorithm. The first algorithm which aggregates the resource information acts as input for the categorized aggregation algorithm to assign rank for the resources. Meta broker allocates the job based on the rank. Form our assessment performed with simulation tool, we proposed advanced job reservation algorithm for resource allocation. Even though no resources are free to run the job, using this advanced resource algorithm we can reserve the resource for job allocation. In addition we proposed bidding technique when more than one users approach same resources. From the simulation results, we conclude that the proposed system reduces the execution time and generates better revenue for Meta broker.

KEYWORDS

Grid Computing, Metabroker, Resource data Aggregation, Broker selection, Job allocation, Advance reservations, Bidding.

1. INTRODUCTION

Grid computing usually consists of one main computer that distributes information and tasks to a group of networked computers to accomplish a common goal [1].

Grid scheduling is also called super scheduling, Meta scheduling and grid brokering. It is one of the advanced features of grid middleware. It is defined as the process of scheduling jobs where resources are distributed over multiple administrative domains. This process can include searching multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites.

In a Grid environment, a resource broker, also called meta-scheduler [2], is usually used to manage user submitted jobs and the scheduling of jobs for execution to the available Grid resources. A Grid meta-scheduler has its own interfaces for the functionalities it provides and also has its own job scheduling objectives. Each grid domain is typically managed by a grid resource broker; the task of scheduling on top of brokers can be called Meta brokering or broker selection.

The grid resource broker provides pairing services between the service requester and the service provider. This paring enables the selection the selection of best available resource from the service provider for the execution of a specific task [3].

In fig.1, different machines can be specified and the different meta brokering policies can be specified to schedule the jobs [4]. When the brokering system starts, all the different layers of the model are instantiated, from the local reservation tables which model how the jobs are mapped to the processors to the brokering component that manages the jobs submitted to the system. In a grid resource broker usually requires the specification of the job requirements from the user. In many cases, the meta brokering policies use these requirements to carry out the matchmaking with the local resources. To allow this, we have extended the Standard Workload Format to specify the job requirements in the workload to be simulated. Each requirement is composed of an identifier, an operator and a value. In the brokering system, the following requirements can be specified for each grid job: memory in MB (e.g., *1024 MB*), processor vendor (e.g., *Intel*, *AMD*), processor clock speed in MHZ (e.g., *1200 MHZ*), operating system (e.g., *Linux*, *AIX*), number of processors (e.g., *4 processors*) and disk size in MB (e.g., *1000 MB).*

Local resource manager manages a set of resources. It can include a queuing system with a local job scheduler that may have its own local policies. Computing resources are the physical machines where user jobs can be allocated and executed. They are also called Computing Elements (CE).

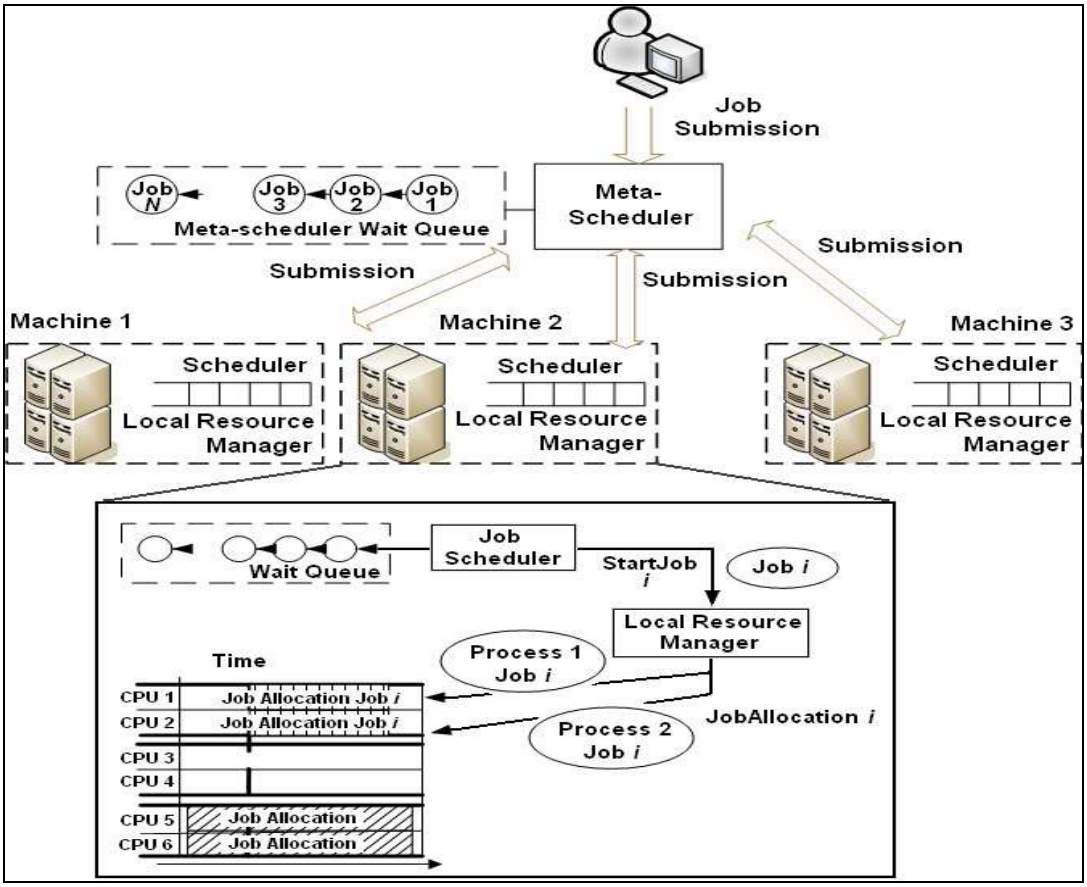

Figure 1. Brokering system [4]

2. RELATED WORK

Regarding Grid Meta broker selection strategies for job reservation and bidding the main paper have been published about Grid broker selection strategies using aggregated resource information [5].

Grid broker selection strategies using aggregated resource information, we have addressed the problem of broker selection in multiple grid scenarios. We have described and evaluated the BrokerRank policy and two variant of this policy: using resource information in aggregated form, and coordinating the scheduling with the underlying scheduling layers. We also have analyzed two different resource aggregation algorithms that have been used by the broker selection policies. Before evaluating broker selection policies, we have studied the scalability of Simple and Categorized resource aggregation algorithms. The results show that the algorithms are scalable in terms of resource information size, and their aggregation processing time is acceptable for an interoperable grid environment. However, we did not address the gain in matching time with aggregated resource information.

Schedulers that allow requesting resources of more than one machine for a single job may perform load balancing of workloads across multiple systems. Each system would then have its own local scheduler to determine how its job queue is processed. It requires advance reservation capability of local schedulers [6].

3. AGGREGATION ALGORITHM

In this section, we analyze two existing aggregation algorithms. The first one is **SIMPLE** aggregation algorithm, other one is **CATEGORIZED** aggregation algorithm.

The SIMPLE AGGREGATION [5] algorithms do the resource data as much as possible looking for maximum compression for scalability; this algorithm loses more detailed information. We use the algorithm to reduce the capacity of memory size and aggregate the values.

Algorithm For Simple Aggregation

Input: proc_type, Proc_speed, OS_type, Memory_size, RAM_size

Output: Aggregated Result

1. Give the input data proc_type, Proc_speed, OS_type, Memory_size, RAM_size
2. Aggregation result made on the basis of Proc_type and OS_type
3. The number of resources that have been aggregated under same category (count) and the sum of all resources values (total) computed.

Simple Aggregation Algorithm Result

CS_AMD Athlon

ProcType='{(AMD Athlon, <count=9>)}'

ProcSpeed='{(2180520,<count=9>,<total=2180520>)}'

CPUUtil='{(162<count=9>, <total=162>)}'

Total CPUs='{(189<count=9>, <total=189>)}'

The second algorithm CATEGORIZED [5] tries to find a good balance between the accuracy of the resource data and the scalability. In addition to the input set of resources, relationships and fixed categories, it also considers different attributes and threshold values.

In this algorithm we use the percentage of processor load for defining subcategories of ComputingSystem resources, and the percentage of used physical memory and used disk for OperatingSystem and FileSystem resources respectively. We also use three different thresholds values: LOW (0:33), HIGH (0:66) and MEDIUM (between LOW and HIGH) [5]. We can increase the level of detail by defining more threshold values. Therefore, as we have commented previously, the main purpose of this algorithm is to avoid the loss of important resource characteristics but maintaining the benefits of aggregation. For example, when we select a broker that contains an aggregated resource of Intel processor vendor and CPULoad attribute of subcategory LOW, we can be sure that some Intel-based computers with low CPU load will be available. The algorithm first computes the category and subcategory of resources depending on

attributes value given a set of thresholds that define the discrete categories. Afterwards it computes the information that contains the resource in aggregated form within the resources of the same category and subcategory. Finally, it establishes the relationships between the aggregated resources respecting the relationships between the original resources.

Algorithm for Categorized Aggregation

Input: Proc_Speed, Total_memory, AvailRam_size

Output: Ranked Brokers

1. Take Proc_Speed, Total_memory, AvailRam_size the resource values
2. Set the threshold value. Ranks are set to the brokers based on the threshold.
3. IF the resources have high capacity then set Rank as 1, else IF medium capacity set the Rank as 2, otherwise Rank as 3.
4. Resources allocate the job if it has Rank 1.
5. Store the resource values in metabroker database.

Categorized Aggregation Algorithm Result

CS_AMD Athlon _High

Proc Type='{(AMD Athlon ,<count=9>)}'

Proc Speed='{(2180520,<count=9>,<total=2180520>)}'

CPU Util='{(162<count=9>,<total=162>)}'

Total CPUs='{(189<count=9>,<total=189>)}'

Ranks are set to the brokers based on the threshold value. The threshold value is calculated using total memory, RAM size, CPU_speed. The High value is ranked as 1, the Mid value is ranked as 2, and then Low value is ranked as 3. We are using this rank Meta broker to select the high capacity resource to job allocation.

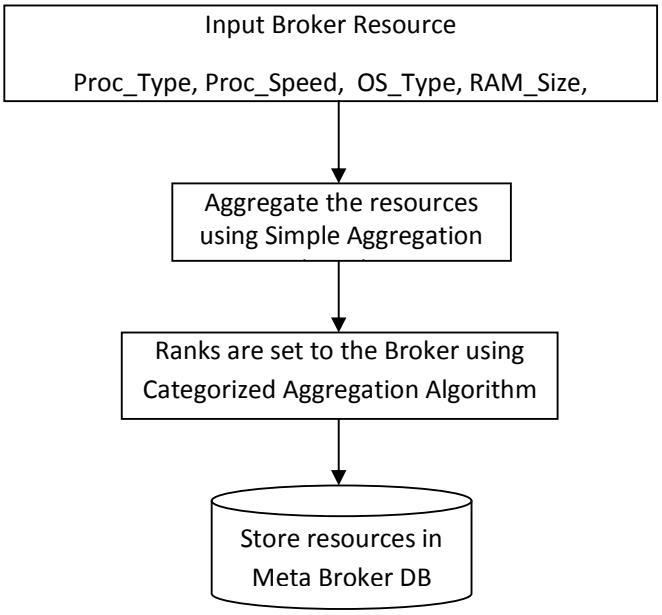

Figure 2. Flow Diagram of the aggregation resource

4. META BROKER ALLOCATES THE JOB TO THE USERS

Algorithm for Job Allocation

Input: Job_Length, File_size, Output_size

Output: Job execution Result

1.Users send the resource request

2. Meta broker match the request with the brokers based on Rank. If the request matches then the meta broker reply to the user. To submit your Job.

3.When the users receive the acknowledgement from the metabroker, users submit the job details for execution.

4. IF the job arrived for execution, metabroker add Job Submitted Queue and allocate the resource.

5. Update finished jobs and send result to the users.

6. After the execution, set the status of resource as FREE. Remove finished job from the Execution Set and add to Finished Set.

7. We use the First Come First Serve Policy to allocate the job

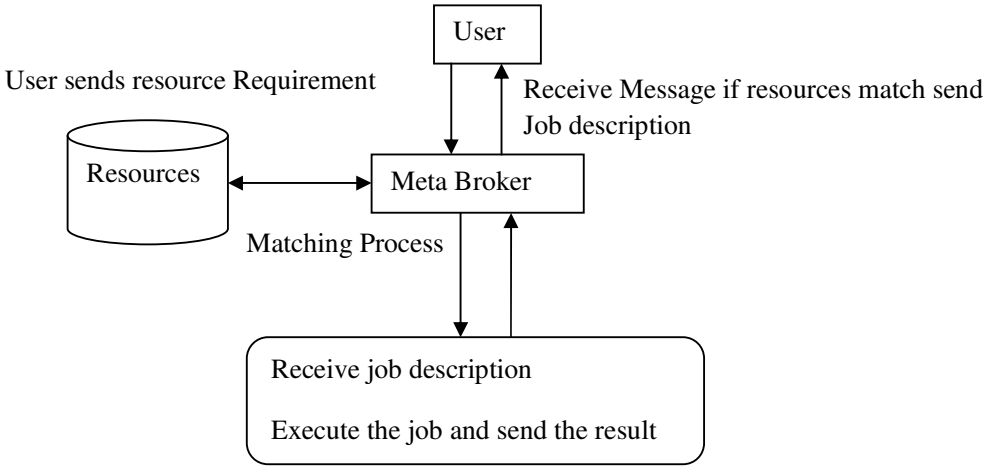

Figure 3. Flow diagram of the job allocation

5. ADVANCED JOB RESERVATION ALGORITHM

The advanced reservation feature makes it possible to obtain a guaranteed start time in advance. A guaranteed start time brings two advantages [7]. It makes it possible to coordinate the job with other activities, and resource selection can be improved as the resource comparison is based on a guaranteed start time rather than on an estimate. The reservation protocol developed supports two operations: requesting a reservation and releasing a reservation. A reservation request contains the start time and the requested length of the reservation, the requested number of CPUs, and optionally, an account to be charged for the job.

Proposed Algorithm for Advanced reservation

Input: Proc_type, Proc_Speed, OS_Type, Total_memory, AvailRam_size

Output: Reservation ID

1.User sends the resource request to Metabroker.

2.Metabroker matches the user request with the broker based on the Rank.

3.IF the user request match with available resource of the Metabroker then it accept the request

4.IF a resource in Metabroker is busy then it send the resource reservation ID to the user.

Using this advanced job reservation algorithm the matching time done by metabroker is reduced. So the execution time also reduces.

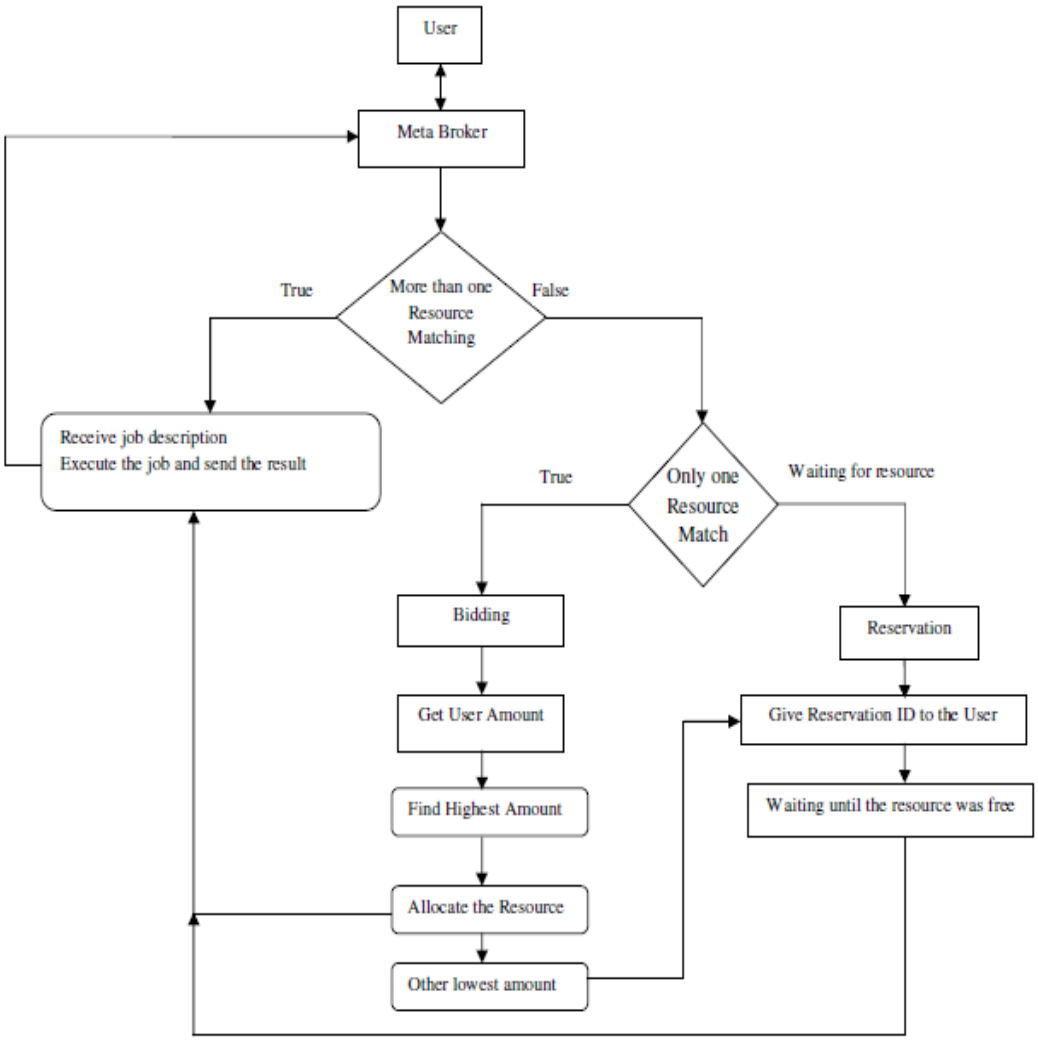

Figure 4. Flow diagram of the Proposed System

6. BIDDING ON USER JOB ALLOCATION

When more than one users approach the same resource request for job execution, Metabroker match the resource request and find the resource availability where as the resources availability will be sufficient for only one user then resource will be allocated based on bidding. Bidding process will be carried as follow as

Let us consider 4 users want same resource

1. Meta broker select the user who have bided for higher amount.
2. Remaining users will get advanced reservation ID.
3. Reservation ID is provided based on descending order (High to low) on bidding amount.

According to this assumption the main goal of the resources is to increase their own profits without considering the amount of time during which the resource is busy. The Bidding process generate the revenue to the Meta broker.

7. SIMULATION RESULTS

7.1 Performance Comparison of Original Memory size with Aggregated Memory size

The original Data got from the brokers takes more memory space. So the Meta broker use Simple aggregation algorithm to reduce the memory space and aggregate the resource information. This result is used to reduce the matching time of resource and reduce the memory space.

Table 1: Memory Size of Aggregated Data

No of resources	Original Data (KB)	Simple Aggregation Data (KB)
50	16	8
100	32	16
150	32	16
200	40	24
250	40	24
300	48	32

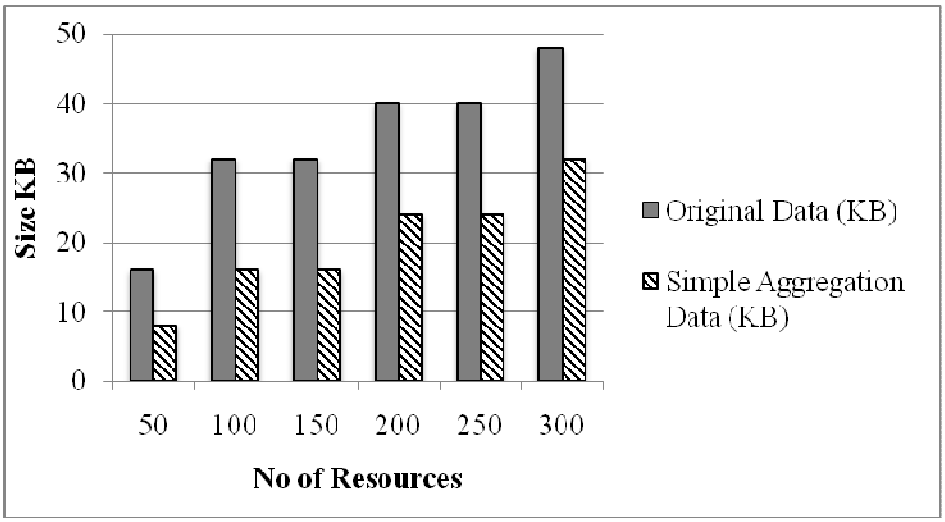

Figure 5. Memory Size of Aggregated Data

Meta broker uses Simple Aggregation algorithm and there is a reduction in the memory space from 50% to 67% compared to the storage of original data.

7.2 Performance Comparison of Job Execution Time with Reserved Job Execution Time

Using this Advanced Job Reservation Algorithm the matching time done by meta broker is reduced. So the Execution Time of a job also reduces. This Proposed system is used to reduce the Time.

Table 2: Job Execution Time

No of Jobs	Job Execution time (ms)	Reserved Job Execution time (ms)
50	150.804598	20.014128
100	251.954023	131.857782
150	217.931034	55.333176
200	251.034483	154.815164
250	57.931034	55.921827
300	253.793103	17.070874

Figure 6. Job Execution Time

The proposed Advanced Job Reservation Algorithm reduces the Execution time from 10 % to 75% compared to the existing one.

7.3 Performance Comparison of Matching Resources with Reserved Matching Resources

During resource reservation process the resource availability status is high whereas before reservation resource availability status is low.

Table 3: Matching Resources

User Request	Without Reservation	Reservation
5	8	10
10	18	20
15	17	18
20	24	26
25	28	30

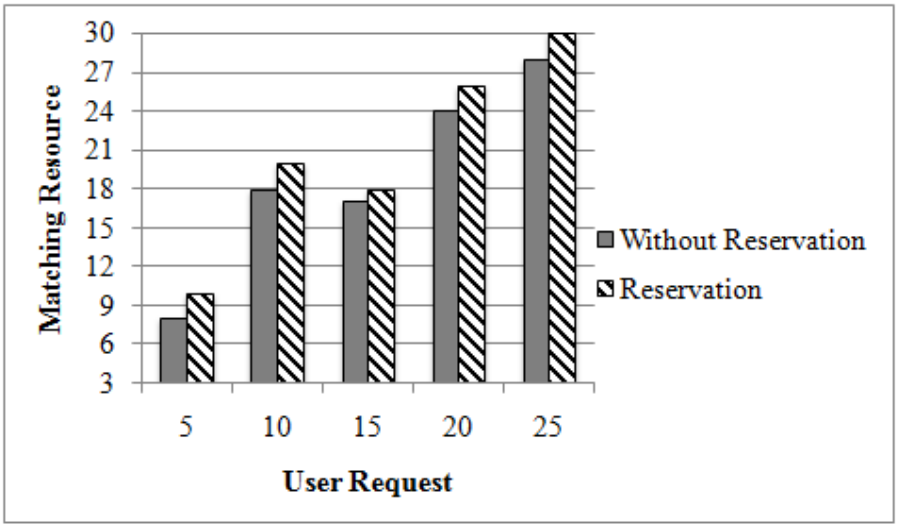

Figure 7. Matching Resources

Meta broker use Reservation process to increase the resource availability status from 80% to 93%.

8. CONCLUSION

In this thesis we use aggregation algorithms for aggregating the data. In the proposed system use of advanced job reservation algorithm reduces execution time compared to the existing one. The

usage of advanced job reservation algorithm reduces the matching time which in turn reduces the execution time of the user jobs. Also revenue generation is satisfactory in the proposed method of bidding algorithm.

9. FUTURE ENHANCEMENT

In future we can use brokering scheduling polices for allocating jobs. Brokering scheduling policies are RealTime, Earliest Deadline First, or Job Rank (JR)-backfilling policies for the job scheduling, and resource selection can be based on the matchmaking approach. We can also use dynamic reservation method for job reservation.

REFERENCES

[1] I. Foster, C. Kesselman, "*Computational Grids, the Grid: Blueprint for a New Computing Infrastructure*", Morgan aufmann, 1998. pp. 15-52.

[2] Rodero, F. Guim, J. Corbalan, L. Fong, Y. Liu, S. Sadjadi, "*Looking for an evolution of grid scheduling: Meta-brokering*", Grid Middleware and Services: Challenges and Solutions (2008) 105_119.

[3] Joshy Joseph, Craig Fellenstein (2004) *Grid Computing*, IBM press.

[4] Ivan Rodero, Francesc Guimb, Julita Corbalan, Liana Fong, S. Masoud Sadjadi, "*Interoperable Grid Scheduling Strategies*", Barcelona Supercomputing Center (BSC-CNS), Spain.

[5] Ivan Rodero, Francesc Guimb, Julita Corbalan, Liana Fong, S. Masoud Sadjadi, "*Grid broker selection strategies using aggregated resource information*", Barcelona Supercomputing Center (BSC-CNS), Spain

[6] Peter Gradwell, "*Overview of Grid Scheduling Systems",* Department of Computer Science, University of Bath.

[7] Erik Elmroth and Johan Tordsson, "*A Grid Resource Broker Supporting Advance Reservations and Benchmark-Based Resource Selection*", Dept. of Computing Science and HPC2N, Ume University, SE-901 87 Umea, Sweden.

Author's profile

Mrs.D. Ramyachitra has completed M.C.A., M.Phil in Computer Science. She is working as Assistant Professor in the School of Computer Science and Engineering, Bharathiar University, Coimbatore. She is also pursuing Ph.D., She has published 9 papers in national & international conferences and 6 papers in national & international journals. Her field of research interest is Grid Computing.

Mrs. S. Poongodi has completed M.Sc., in Computer Science. She is currently pursuing her M.Phil., in Computer Science in the School of Computer Science and Engineering, Bharathiar University, Coimbatore. She has published a paper in national conference and a paper in international journal. Her field of interest is Grid Computing.