AN INGENIOUS APPROACH FOR IMPROVING TURNAROUND TIME OF GRID JOBS WITH RESOURCE ASSURANCE AND ALLOCATION MECHANISM

Prachi Pandey, Shamjith KV, Shikha Mehrotra, Asvija B, R Sridharan
Centre for Development of Advanced Computing (C-DAC), Bangalore, India
{prachip, shamjithkv, shikham, asvijab, rsridharan}@cdac.in

Abstract*— *In a heavily used grid scenario, where there are many jobs competing for the best resource, the meta-scheduler is burdened with the task of judiciously allocating appropriate resources to the jobs. However, as the demand for the resources increases more and more, it becomes really difficult to manage the jobs and allocate resources to them and hence most of the jobs will be in the queued state waiting for the resources to be free. Gradually, it leads to a situation where the jobs stay in queued state longer than the execution state, resulting in highly increased turnaround times. The challenge therefore is to make sure that the jobs don't take an unreasonable time to complete because of the increased waiting time. In this paper, we discuss about the advance reservation mechanism adopted in Garuda Grid for assuring the availability of compute resources and QoS based resource allocation. Results of the experiments carried out with this setup confirm the reduction in queuing time of jobs in grid, thereby improving the turnaround time.

Keywords- Advance Reservation in grid, Guaranteeing performance in grid computing, improving turnaround time, QoS.

I. INTRODUCTION

In a typical computational grid[1][2], job management is handled by the Grid meta-scheduler[3][4]. It takes care of short listing the candidate resources as per the job requirements, managing job execution and job control. The meta-scheduler[3][4] schedules the job based on the availability of resources at that instant. However, it is incapable of guaranteeing the availability of a particular resource at any point of time. In an ideal case, where there is enough number of resources to satisfy each and every job request, it might be sufficient to have a metascheduler, which does not take care of guaranteeing the availability of resources to the user. But, practically, as the demand of the resources increases, and since most users want to execute their jobs on higher-ranking resources, it becomes necessary to have a mechanism, which will guarantee the availability of resources to a user at a particular time. In the absence of such a system, many jobs will have to go to the queue state and wait for the resources to be free. This in turn will increase the turnaround time of the jobs to a great extent decreasing the performance of the system as a whole.

Garuda advance reservation [7] facility in grid ensures the availability of resources required by a user or an application at specified future times. Advanced grid resource reservation, which can be independent of a job, can be requested by a user or an administrator and granted by the reservation manager based on the privileges of the user or application, and with the policies enforced on the resources. The reservation causes the associated resources to be reserved for the specified user, administrator, or an application for the duration of request. Thus as soon as the reservation becomes active, the job submitted with the reservation ID starts executing. This reduces considerably the time spent in the queue by the jobs and thus improves the turnaround time of the jobs.

The paper is thus organized in the following structure. Section II gives a brief description of the architecture of the reservation system and its components, followed by Section III, describing the features of the Reservation System. Section IV contains the implementation details of the System while Section V details a Case study of the Garuda Grid, where we conducted an experiment to prove how the turnaround time can be improved by having the Reservation System. Section VI includes a discussion on the impact of the system on the utilization of the grid. The paper concludes with a mention of the related works (Section VII) carried out in the field of Advance Reservation of resources in grid computing.

II. ARCHITECTURE

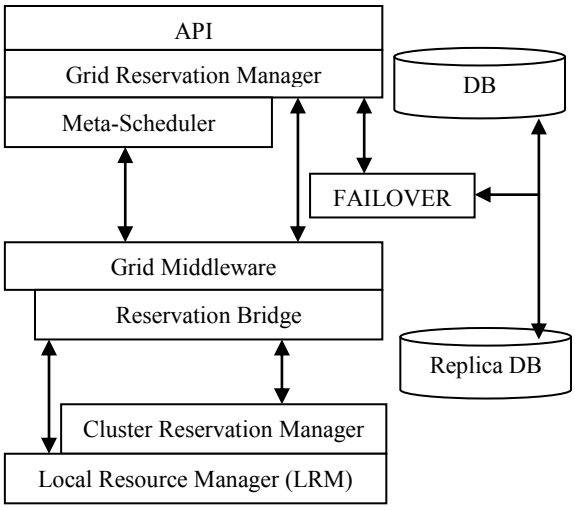


Figure 1. Architecture of the reservation system

Fig. 1 shows the architecture of Garuda Reservation System, depicting the components involved and their integration with Meta-scheduler [3][4], Middleware[7] and Local Resource Manager[9].

The Garuda Advance Reservation System consists of the following components:

- Advanced Reservation Manager.
- Reservation module for Grid Meta Scheduler.
- Reservation extension for Grid Middleware.
- Reservation enabled Local Resource Manager.
- Reservation accounting service.

Advanced reservation manager is the core of the grid reservation system. This component is responsible for managing the reservation activity across the grid resources. The reservation manager is integrated with the Grid Meta-scheduler, Grid Middleware and Local resource manager. The Grid Meta-scheduler, which judiciously schedules jobs on the grid, plays a major role in ensuring availability of resources for advanced reservations. The reservation module for Grid Meta-scheduler coordinates the submission of jobs with a valid reservation to the grid. The local resource manager accomplishes actual job execution on the computing resources, which is the bottom layer entity responsible for executing jobs on a cluster system[10]. The reservation facility has been imposed through the Local resource manager in order to guarantee the availability of the reserved resources during the reservation period. The reservation extension for Grid Middleware takes care of integrating the reservation system with local resource manager and meta-scheduler, thereby facilitating the reservation enabled job execution. The accounting service[11]available with reservation system keeps track of all the reservations made in the grid resources.

A. Grid Reservation Manager

The Garuda Grid reservation manger is responsible for initiating and enforcing the compute node reservation in the grid. The grid reservation manager along with Gridway[4] ensures that the jobs with reservation are given higher priority as compared to the unreserved jobs. The Grid reservation manager extends the interfaces required for reservation management ie. the creation, modification and cancellation of grid reservation both by users as well as applications. It also employs a resource selection algorithm to provide the best available resources for the grid users or applications. The resource selection is a highly dynamic algorithm, which is based on the QOS[12][13] parameters of each resource in the Garuda grid. The Grid reservation manager has been integrated with the Garuda QOS Engine[11][12] to shortlist the candidate resources according to the reservation request. The Grid reservation manager also implements the Garuda Reservation Policies to make sure of the fair usage of resources by the grid users, applications and services.

The application components, which constitute Garuda Reservation Manager, reside on every head node in the Garuda [5] grid to facilitate the reservation and job submission on the grid from every head node.

B. Cluster Reservation Manager[14][15]

It is necessary to have a layer for enforcement of reservation of resources in the local resource manager level to execute the reservation activity in a cluster environment. The Garuda cluster reservation manager is the component, which is responsible for enforcing the reservations in a Garuda cluster. The cluster reservation manager along with the Local resource manager and grid middleware guarantees the mapping of Grid reservation on to the computing resources in respective clusters. The cluster reservation manager resides on every cluster, which is enabled with Garuda grid reservation.

C. Garuda Middleware Reservation Bridge

The Garuda Middleware Reservation Bridge is the component that acts as glue between the Grid Reservation Manager and the Garuda Cluster Reservation Manager, thus making the Garuda Grid Reservation System complete across all the layers of Grid computing infrastructure. It has been integrated with the job submission module of Globus middleware to accomplish reservation management and job submission activities.

III. FEATURES

The key features of Garuda Grid Reservation System are as follows:

- Ensure resource availability: When a grid user makes an advance reservation, the Garuda Reservation System provides the reserved resources to the owner of the reservation for the complete reservation period. During this period, the grid reservation system does not allow any other grid user to encroach and submit jobs on those resources. Thus the Garuda Reservation System is able to guarantee the availability of the required number of resources well in advance, so that the grid users can plan their job run activities without having to worry about the resource availability.

- GSI[16] based reservation: Garuda Reservation depends on the Grid Security Infrastructure to impart security in the reservation system. In the reservation system, the Grid Distinguished Name (DN)[17] identifies each user uniquely. This makes it mandatory to have a valid grid certificate from an internationally recognized Certification Authority (CA)[16] to use the Garuda Grid Reservation System.

- Grid Reservation Failover mechanism: Failover recovery is a very important aspect of a component in a grid computing scenario. To make sure that the failure of any reservation component does not affect the smooth operation of the Garuda Reservation system, effective recovery mechanisms have been incorporated to address both the database and cluster system failures. In Garuda Reservation System, failover mechanism has been considered for two major components as described below.
    - Grid and Cluster Level Reservation Components: As mentioned before, the grid and cluster level reservation program components are deployed in each cluster in the Garuda grid, so that even if a particular cluster fails, the users can login into the nearby cluster and use the reservation commands to make an advance reservation.
    - Reservation Database System: Since the reservation database is a very important component for the efficient functioning of the reservation system, we cannot afford to lose the critical reservation data because of the database failure. To overcome this, a replica of the centralized reservation database has been created and maintained using a trigger based replication approach. In case of a failure, all the transactions are carried out with the database replica instead of the original database thus ensuring the high availability of Garuda reservation system at all times.

- Application Programming Interface: In order to facilitate the use of reservation mechanism directly within the applications, portals[18], PSE and services, the Garuda Reservation System exposes the Application Programming Interface (API) for all the reservation management activities. With the help of APIs, developers can easily plug the reservation system in to their applications or services to ensure the required amount of CPUs in a grid. This would improve the performance and better utilization of resources.

- Intelligent resource allocation based on QoS Parameters: The reservation system makes use of the Garuda QOS Engine to rank the available resources for reservation. This helps the user to choose the best resource for his job to ensure high throughput and availability of resources.

- Virtual Organization[19] support: The resource allocation algorithm respects the VO rules when making a resource reservation. This helps users to get their reservation created on only those resources which belong to their registered VOs.

- Avoiding resource under utilization: The reservation system also intelligently takes care of avoiding under utilization of resources, by identifying the unused reservations and de-allocating it for satisfying further requests.
- Integration with Gridway Meta-scheduler and Globus Middleware: The reservation system is not an independent component, but it has been built on top of the Gridway Meta-scheduler and Globus Middleware. Both the Gridway as well as the Globus middleware have been customized to handle reservation.

IV. IMPLEMENTATION

This system has been implemented as part of the middleware stack of the GARUDA grid project.

Reservation flow control in Garuda

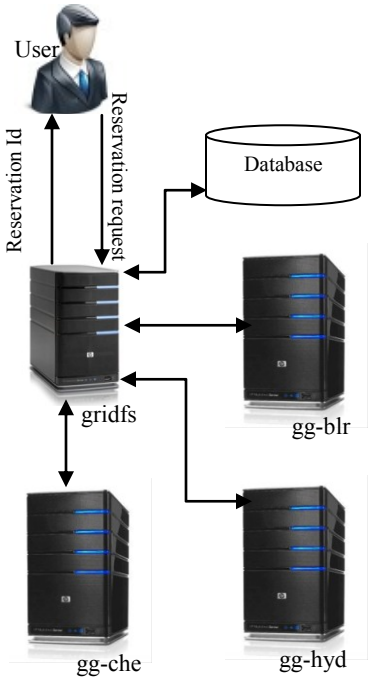


Figure 2. Flow of control for reservation

The steps involved in creating a reservation are detailed below

- Input parameters: for creating a new reservation, the parameters expected by the API are start time, end time/duration and number of resources to be reserved. The user credential with which the reservation needs to be created also must be provided.
- Authentication and authorization using GSI credentials and VO rules: The credential provided by the user is interpreted by the reservation system to identify the Certificate Authority, Distinguished Name and VO privileges. This information is used to

authenticate and validate the user through Globus and VOMS services.

- Resource allocation: Based on the inputs given by the user, the reservation manager queries the reservation repository to find out available resources. Then, it interacts with the Garuda QoS engine to shortlist best available resources for reservation, based on their availability, performance and reliability parameters.
- Interacting with reservation components in respective clusters through Globus communication protocols/services:
- Instructing the Maui scheduler to reserve compute nodes as per the user requirements.
- Reservation details are published in a global reservation information repository
- Unique identifier for the grid reservation is provided

Using this unique reservation ID the user can submit jobs to the grid when the reservation time starts. Facilities available with reservation system are,

- Advanced / Immediate Reservation of resources across multiple clusters,
- Modification of reservation duration and resources,
- Cancellation of existing reservation,
- Job submission with reservation.
- Listing active reservations of users
- Providing CPU and node reservation.
- Failover and replication for information repository
- Application Programming Interface in Java.
- Interface for Administrators to manage/control reservations.
- Automatic cancellation of un-used reservation slots.

V. CASE STUDY

Garuda makes use of the Gridway meta-scheduler to manage the jobs. The LRM used is Torque, which makes use of Maui scheduler to schedule jobs and enforce reservation on the nodes.

To study the performance of scheduling with advance reservation, we conducted the following experiment. We identified few HPC resources that are part of Garuda computational grid infrastructure, and are distributed across different geographical and administrative boundaries. A compute intensive application was identified as our workload. The Gridway meta-scheduler is used to submit jobs to the respective HPC resources in the grid. The experiment was repeated multiple times with same data sets, both with and without enabling reservation on HPC resources. The results obtained were analyzed to understand the variations in turnaround times in both the scenarios.

PERFORMANCE METRICS

The following performance metrics were considered for evaluation.

- Mean waiting time (Avg. amount of time the job waits before it is scheduled to a resource)
- Execution time (time spent in execution of the task)
- Turnaround time (total time taken between the submission of a program/process/thread/task (Linux) for execution and the return of the complete output to the customer/user)

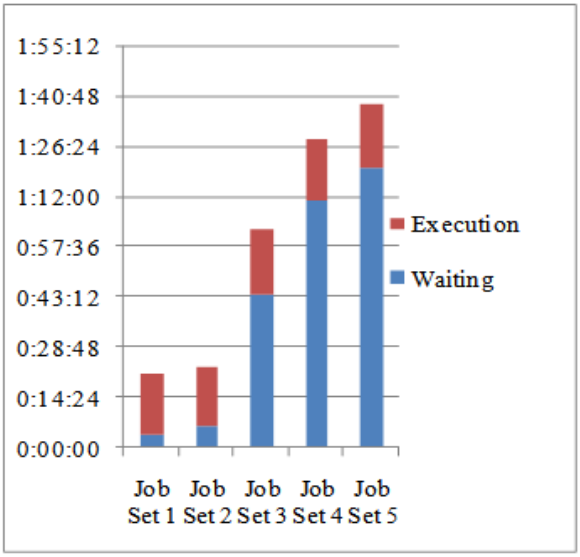
Scenario I: Without the Reservation System

The compute intensive application was submitted multiple times at different time slots. Jobs submitted in a particular slot are considered as one job set. We noted the average of waiting, execution and turnaround times of jobs in each set. Table 1.0 shows the average time taken by the jobs in five different job sets.

Table 1

Job Set	Waiting	Execution	Turnaround
Job Set 1	0:04:00	0:17:16	0:22:02
Job Set 2	0:06:00	0:17:27	0:24:14
Job Set 3	0:44:00	0:18:31	1:02:49
Job Set 4	1:11:00	0:17:27	1:38:42
Job Set 5	1:20:00	0:18:26	1:37:41

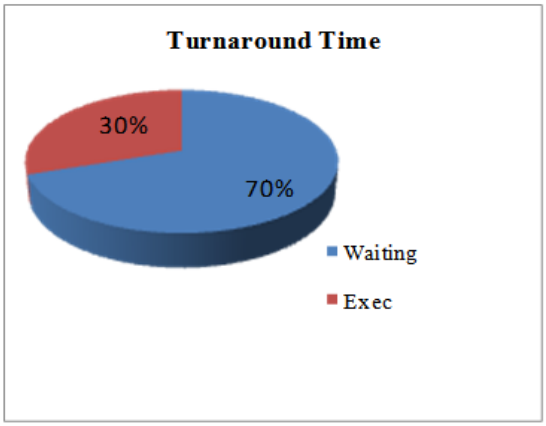
Based on the output obtained, a bar graph is plotted which depicts the average time spent by the jobs in waiting state and in the state of execution.



Graph 1: Execution Vs Waiting Time without reservation

From the Graph 1, we observe that initially the waiting time for jobs is very minimal, owing to the fact that the resources are free, but as the resources become busy and more and more jobs are submitted, the waiting time increases and finally it reaches a stage where it starts to exceed the execution time. The execution time however, remains more or less the same throughout.

Assuming that all other factors are constant, the turnaround time would be a simple combination of the waiting time and the execution time. On calculating the average of waiting time, execution time and turnaround time of all job sets, we present the percentage division of the turnaround time by means of a pie graph.



Graph 2: Turnaround time without reservation

From Graph 2, it is clear that the job spends more time in waiting for the resources (almost 70%) than in execution (around 30%).

Scenario II: With the Reservation System

The same experiment was performed again in the test bed with the Reservation System enabled. We created a time bound reservation on the resources and then using the respective Reservation IDs, jobs were submitted in multiple sets. Table 2 shows the average time taken by the jobs in five different job sets.

Table 2

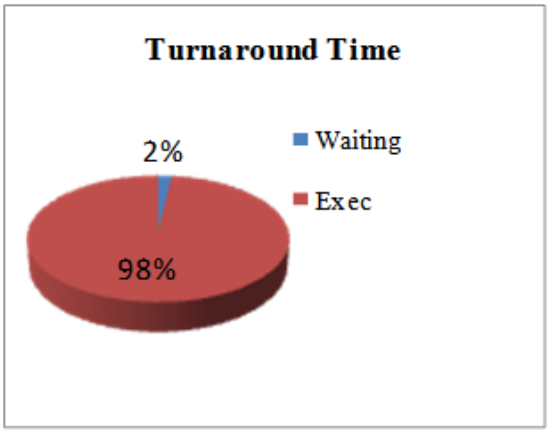
Job Set	Waiting	Execution	Turnaround
Job Set 1	0:00:09	0:08:03	0:08:32
Job Set 2	0:00:09	0:08:05	0:08:35
Job Set 3	0:00:09	0:08:07	0:08:37
Job Set 4	0:00:09	0:08:05	0:08:37
Job Set 5	0:00:08	0:07:15	0:07:45

With the output obtained, a bar graph was plotted which depicts the average time spent by the jobs in waiting state and for execution.

Graph 3: Execution Vs Waiting Time with reservation

From Graph 3, it is very evident that the waiting time of jobs has been minimized; in fact it is almost nullified.

To estimate the amount of time spent by the job in waiting as compared to execution state, we draw a pie chart for the turnaround time ignoring the other factors.

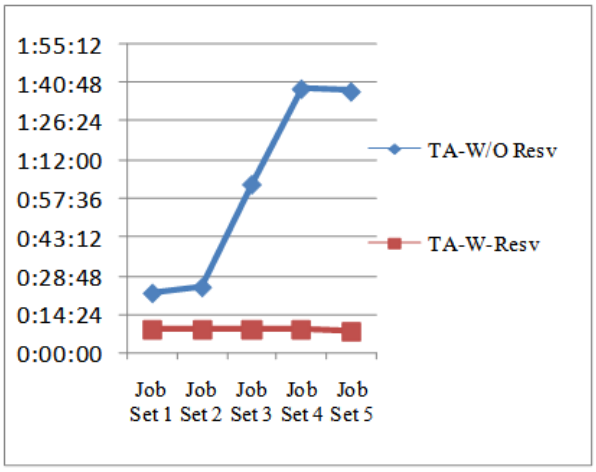


Graph 4: Turnaround time with reservation

It is to be noted that the above graph does not mean that the execution time has increased as compared to the case without reservation. It just shows that now the execution time constitutes most of the turnaround time and the waiting time is just a small fraction of it.

EXPERIMENTAL RESULTS AND DISCUSSION

Finally, we plot a graph to compare the turnaround time of both cases turnaround time without reservation (TA-W/O Resv) and turnaround time with reservation (TA-W-Resv)



Graph 5: Turnaround time plot with and without reservation

It is quite evident from the graph that there is a huge improvement in the turnaround time with the Resource Reservation System enabled, as compared to without it. This is because the waiting time is reduced considerably when the job is submitted with reservation

VI. IMPACT OF RESERVATION SYSTEM

Enforcing prior reservation of the resources ensures that the applications execute with shorter turnaround times and get benefited with higher sustained memory bandwidths and lower message latencies. Nevertheless, a couple of concerns need to be addressed for deriving optimal benefits out of the reservation system. Foremost is the threat that implementing a reservation system can result in lower utilization of the resources, since there might be gaps between the successive reservations which are too small to be able to schedule a task. However, our studies conducted in a production grid infrastructure indicate a minimal degradation in utilization, as a result of this factor. Nevertheless, this small tradeoff could well be justified with the compelling performance improvements perceived by the applications running on reserved resources.

An interesting argument with existing reservation systems is that their usage will merely result in the waiting time of the job, getting shifted from the resource to the user. However, in the case of a grid, where the resources are being heavily used, the information of availability of resources in future will help the users plan their simulation activities accordingly instead of waiting indefinitely for the required resources. The current framework also provides a mechanism for requesting immediate reservation if the job has to be executed immediately.

Another major challenge in implementing reservation systems is to ensure that the resources are being utilized prudently during the reserved time slots. Jobs that exceed the reserved duration of time can run into termination, as a result of stringent policy enforcements. Such measures would typically result in the users overestimating the execution time. One of the measures to avoid such a scenario is to provide an automatic extension of the reservation slot by a predefined factor, to facilitate the graceful terminations of user jobs. Simulation studies carried out indicate that awarding an extra 25 percentage of the reservation time, would result in a judicious termination of the user jobs which exceed the pre-reserved slots. There can also be cases when the user fails to cancel a reservation which he does not intend to use in future. This would block the time slot and would not allow other users to utilize the time as well. To prevent this, the system implements the automatic cancellation feature which keeps a check on the unutilized reservations and cancels them automatically after a specific reservation time has elapsed without being used.

VII. RELATED WORK

Globus team, in the pre-web service version of the grid middleware, presented a General-purpose Architecture for Reservation and Allocation (GARA)[20] in which distributed computing and communication resources provide a reservation capability immediately or for some future time span. However, it has been deprecated in the new Web Service (WS) version of the Globus toolkit. Although some of the commercial meta-schedulers like PBSPro[21], Moab[22] etc also provide the advance reservation capabilities, they have not been widely accepted by resource providers in a grid-computing environment, because of their cost and managerial complexities. The conventional grid reservation systems provided by these meta-schedulers mostly fail to address key features that are essential in a grid computing environment like direct integration with Globus middleware in clusters, resource underutilization avoidance mechanism, and reservation across multiple clusters and Virtual Organization support. The research we pursued to address these concerns gave a way for the Garuda Reservation System. The experiment results provided in this paper confirm that the approach ensures that there is a marked improvement in the turnaround times of jobs submitted with Reservation as compared to without it.

VIII. CONCLUSION

In this paper, we have presented an approach to improve the turnaround time of jobs in grid by adapting an Advance Reservation Mechanism. The experiment results have confirmed that by enforcing the reservation of resources in the grid, we could almost eliminate the waiting time for jobs and hence improve their turnaround time. This reservation system has been deployed and being used by various grid users of GARUDA grid including climate modeling, fluid dynamics and bioinformatics.

REFERENCES

[1] Computational Grids., I. Foster, C. Kesselman. Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999.

[2] I. Foster. What is the Grid? A Three Point Checklist. http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf, 2002

[3] E. Huedo, R.S. Montero, I.M. Llorente. Grid Architecture from a Metascheduling Perspective. Computer, 43 (7):51-56, July 2010.

[4] E. Huedo, R.S. Montero, I.M. Llorente. The GridWay Framework for Adaptive Scheduling and Execution on Grids. Scalable Computing: Practice and Experience, 6(3):1-8, September 2005.

[5] Ram, N., Ramakrishnan, S. "GARUDA: India's National Grid Computing Initiative," CTWatch Quarterly, Volume 2, Number 1, February 2006. http://www.ctwatch.org/quarterly/articles/2006/02/garuda-indias-national-grid-computing-initiative/

[6] National Knowledge Network website, http://nkn.in/

[7] W. Smith, I. Foster, and V. Taylor, Scheduling with Advanced Reservations, *Proc. of the Int. Parallel and Distributed Processing Symposium (IPDPS) Conf.*, Cancun, Mexico, 2000, 127–132.

[8]Foster, I., Globus Toolkit version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, 2005

[9]Torque http://www.adaptivecomputing.com/products/torque.php

[10]Cluster Computing, http://en.wikipedia.org/wiki/Computer_cluster

[11] Zhengyou LIANG, Ling ZHANG, Shoubin DONG, Wengou WEI, "Charging and Accounting for Grid Computing System"

[12] Colling D, Ferrari T, Hassoun Y, Huang C, Kotsokalis C, McGough A, Patel Y, Ronchieri E, Tsanakas P. On quality of service support for grid computing. Proceedings of the Second International Workshop on Distributed Cooperative Laboratories (Grid Enabled Remote Instrumentation), Davoli F, Meyer N, Pugliese R, Zappatore S (eds.). Springer: New York, 2008; 313–327.

[13] Asvija B, Kalaiselvan K, Sridharan R, Dr. S.R. Krishnamurthy. "A performance based QoS aware resource brokering framework for the grid"

[14] Brett Bode, David M. Halstead, Ricky Kendall, and Zhou Lei Scalable Computing Laboratory, Ames Laboratory, DOE Wilhelm Hall, Ames, IA 50011, USA, help@scl.ameslab.gov David Jackson, Maui High Performance Computing Center . "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters"

[15]Maui Cluster Scheduler. http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php

[16] The Globus Security Team, "Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective". Version 4 updated September 12, 2005

[17] Distinguished Name, http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/glossary.html#dn

[18] Arackal, V.S., Arunachalam, B., Bijoy, M.B., Prahlada Rao, B.B., Kalasagar, B., Sridharan, R., Chattopadhyay, S. "An access mechanism for Grid Garuda". Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference, Dec 2009, Bangalore.

[19] Katzy, B.R.; , "Design and implementation of virtual organizations," System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on , vol.4, no., pp.142-151 vol.4, 6-9 Jan 1998 doi: 10.1109/HICSS.1998.655269

[20] Foster, I., Roy, A. and Sander, V., A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. In *Proc. 8th International Workshop on Quality of Service*, 2000.

[21] PBS Website, http://www.pbsworks.com/?AspxAutoDetectCookieSupport=1

[22] Moab website, http://www.adaptivecomputing.com/resources/docs/mwm/6-0/moabusers.php