INTEGRATING HETEROGENEOUS COMPUTING RESOURCES TO FORM A CAMPUS GRID

By

SIDDHARTHA ELUPPAI SRIVATSAN

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2009

To the Lord, who watches over everything

ACKNOWLEDGMENTS

I express my most sincere appreciation to my supervisory committee chair, Dr. Herman Lam. I thank him for the education, advice, and the encouragement that he had provided me with during the course of my study at the University of Florida.

I thank Dr. Craig Prescott for lending his knowledge and support, and providing technical guidance. I also thank Dr Paul Avery for providing me with all the apparatus required to carry out the experiments for my thesis. It is a great privilege to have worked with such far-thinking and inspirational individuals. This thesis would not have been possible without their support and guidance. I would also like to express my deepest appreciation to my parents Srivatsan and Usha Srivatsan; their love, understanding, patience and personal sacrifice made this thesis possible.

TABLE OF CONTENTS

page

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

INTEGRATING HETEROGENEOUS COMPUTING RESOURCES
TO FORM A CAMPUS GRID

By

Siddhartha Eluppai Srivatsan

August 2009

Chair: Herman Lam
Major: Electrical and Computer Engineering

In this thesis, a methodology and an architecture for integrating heterogeneous compute clusters at the University of Florida to form a campus grid is described. The system, named GatorGrid, provides a way of scheduling jobs across the various research clusters, user authentication across administrative domains, and file staging using a mix of open-source and proprietary technologies such as OpenLDAP, Lustre file system, Torque resource manager and MOAB scheduler. In doing so, separate administrative domains can remain heterogeneous and autonomous, but still participate in a combined campus grid. Also, it is ensured that there are minimal changes to the job scripts and the way in which jobs are submitted to the combined clusters. A test bed was built and a reduction in queue time was observed.

CHAPTER 1
INTRODUCTION

A compute grid is an infrastructure that allows loosely-coupled, heterogeneous, and geographical dispersed sets of hardware, software and human resources to be shared and act in concert to perform large computing tasks. A compute grid provides a way of scheduling jobs across the various clusters, user authentication across domains, and file staging using a mix of open-source and proprietary technologies. Such compute grids can vastly increase an organization's computing capacity, raw storage capacity, and can improve overall performance (parallel execution), scalability, and fault-tolerance (offloading computation to available nodes). There are four types of compute grids, namely cluster grids, campus grids, enterprise grids, and global grids. Cluster grids are also known as departmental grids. Each of these clusters can be located at a particular site within an organization and is owned by different groups. A campus grid differs from a cluster grid in that resources belong to multiple owners. Enterprise grids contain resources located at multiple sites of the same enterprise. Finally, global grids contain resources from multiple enterprises.

Currently at the University of Florida, there are different computing domains, each containing a cluster grid as shown in Figure 1-1. Examples of cluster grids at UF include the clusters at the High Performance Computing (HPC) Center in the College of Engineering, Quantum Theory Project (QTP) at the Physics Department, High Performance Computing and Simulation Research Labs (HCS) at the Department of ECE, and the Interdisciplinary Center for Biotechnology Research (ICBR). There is no central IT department that is responsible for all the computing domains. The different research groups, each with their own computing requirements, fund their own computing domains and are expected to maintain all the software and hardware for running their applications. This has resulted in each domain having to maintain its own file

systems, User (UIDs) and Group (GIDs) Identifiers and its own resource managers. Also, some of the clusters have a large number of users and are over-utilized, while others are under-utilized.

In this thesis, a methodology and an architecture will be described for integrating heterogeneous compute clusters at the University of Florida to form a campus grid, named GatorGrid. A test bed has been implemented to integrate nodes representative of two clusters in the Physics Department, maintaining their autonomy and heterogeneity. The advantages of such a campus grid are as follows. All the computing resources in the campus grid would appear as a seamless entity to a researcher. The result is a tremendous scale up in terms of the computing power available to the researcher as a campus grid leads to aggregation of resources and sharing of workload across the organization. Also, from a user point of view, maintaining local autonomy means that researchers do not have to learn new ways of submitting their jobs. From a point of view of IT infrastructure installation and deployment, the system would offer greater flexibility in that newer clusters can join the grid when needed but still maintain local control. Our approach involves a methodology to provide a way of scheduling jobs across the various clusters, user authentication across domains, and file staging using a mix of open-source and proprietary technologies such as OpenLDAP, Lustre file system, Torque resource manager, and MOAB scheduler.

However, providing such a campus grid has its challenges. Not all of the above-mentioned technologies may be currently in use in all the participating clusters. For example, in the cluster maintained by the Quantum Theory Project (QTP), user identifiers (UIDs) and group identifiers (GIDs) are maintained in flat files. However, if it participates in the campus grid, it will need to upgrade to manage UIDs and GIDs using LDAP servers. Furthermore, when upgrading to such technologies, maintaining local autonomy is of utmost importance.

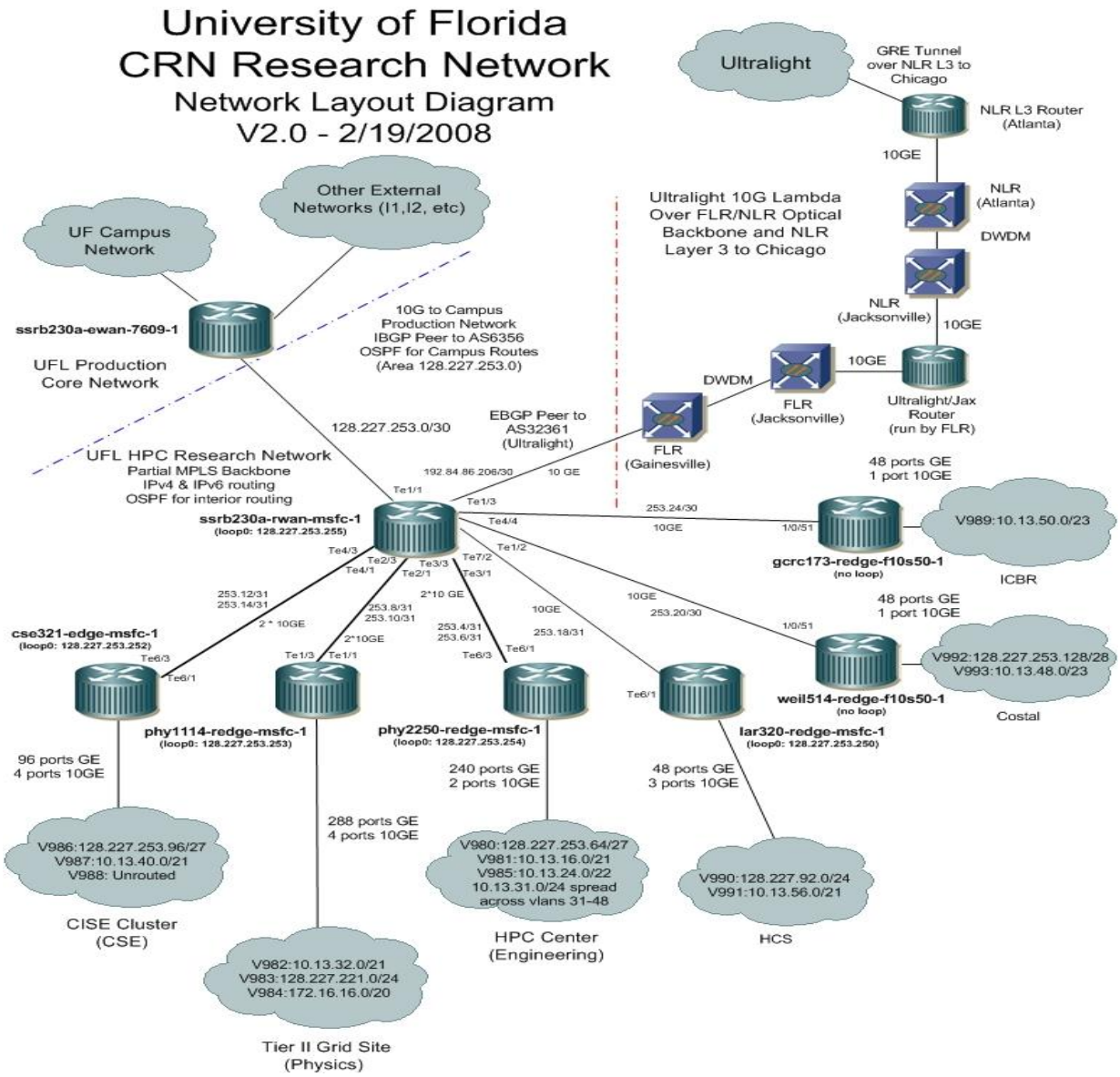

Figure 1-1 UF computing domains

The remainder of this thesis is divided into five chapters. Chapter-2 outlines related works in the area of campus grids. In Chapter-3, an architecture of GatorGrid is given. Discussion is given on how the key components such as LDAP proxy servers, MOAB scheduler, and Lustre file system are used to solve the problem of user authentication across domains, job migration between clusters, and file staging, respectively. Chapter-4 provides the details of the design and implementation of a test bed, including the installation and configuration of the LDAP proxy and the configuration of MOAB job managers in a peer-to-peer fashion. The test bed was then

subjected to various workloads for testing. In Chapter-5, the test results are provided and discussed. Finally, Chapter-6 gives some concluding remarks and discusses the issues one might face if this were to be extended to form a more comprehensive State of Florida grid.

CHAPTER 2
LITERATURE REVIEW

There were several earlier attempts to build campus grids. Some of them are still ongoing efforts to establish a campus grid such as the UT grid [1] while, others such as University of Wisconsin Madison's (UW Madison) campus grid [2] and the Boiler Grid [3] are configured and in use. The UT grid is a collaborative effort between the University of Texas and IBM to build and deploy a campus grid. The UT grid provides a Graphical User Portal (GUP) for users to sign on and submit jobs graphically and a GridShell for users to sign on through the command line. There is also an ongoing effort to allow multiple resource managers like PBS, LSF, SGE and Condor to work together. As far as data management is concerned, the UT grid plans to use the Avaki data grid. Although the UT grid introduces a novel Graphical User Portal for user authentication, it may not be adoptable for our use as it may interfere with administrative domains adding their own users and thus local autonomy would be sacrificed.

In the UW Madison grid, Condor pools in various departments are made accessible via Condor "flocking". Users submit jobs to their own private or department Condor resource manager. Jobs are dynamically matched to available machines. A representation of the scenario is provided in Figure 2-1 taken from [2]. Moreover, their users want to collaborate outside the bounds of the campus and do not want to be limited to sharing resources with departments who have made identical technological choices. Hence, they used the OSG software stack to install services such as Globus Gatekeeper, GUMS, etc. and use it for user authentication, data handling and storage. This is illustrated in Figure 2-2 taken from [2].

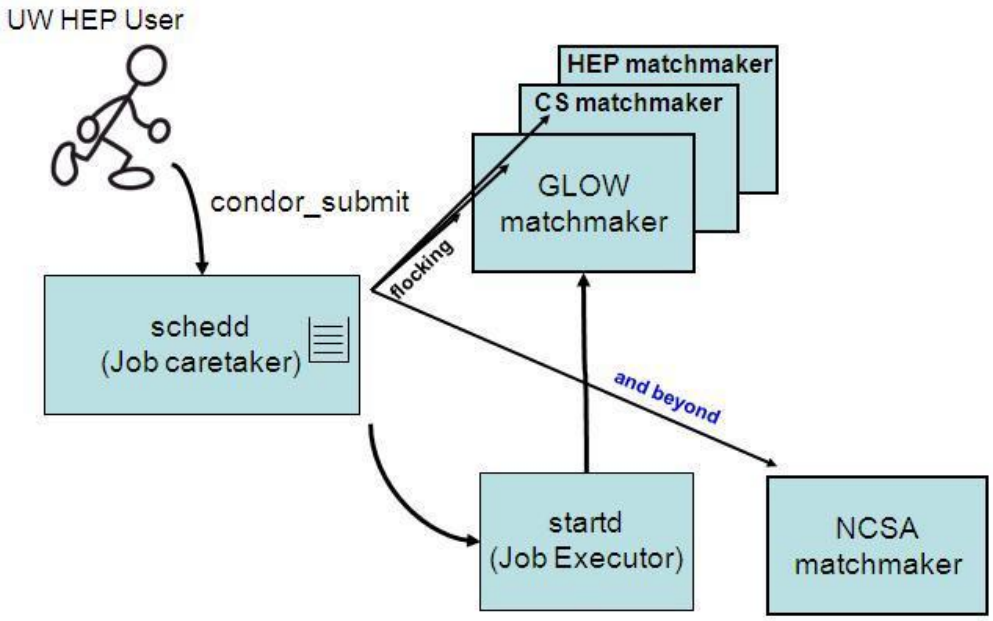

Figure 2-1. UW Madison campus grid setup

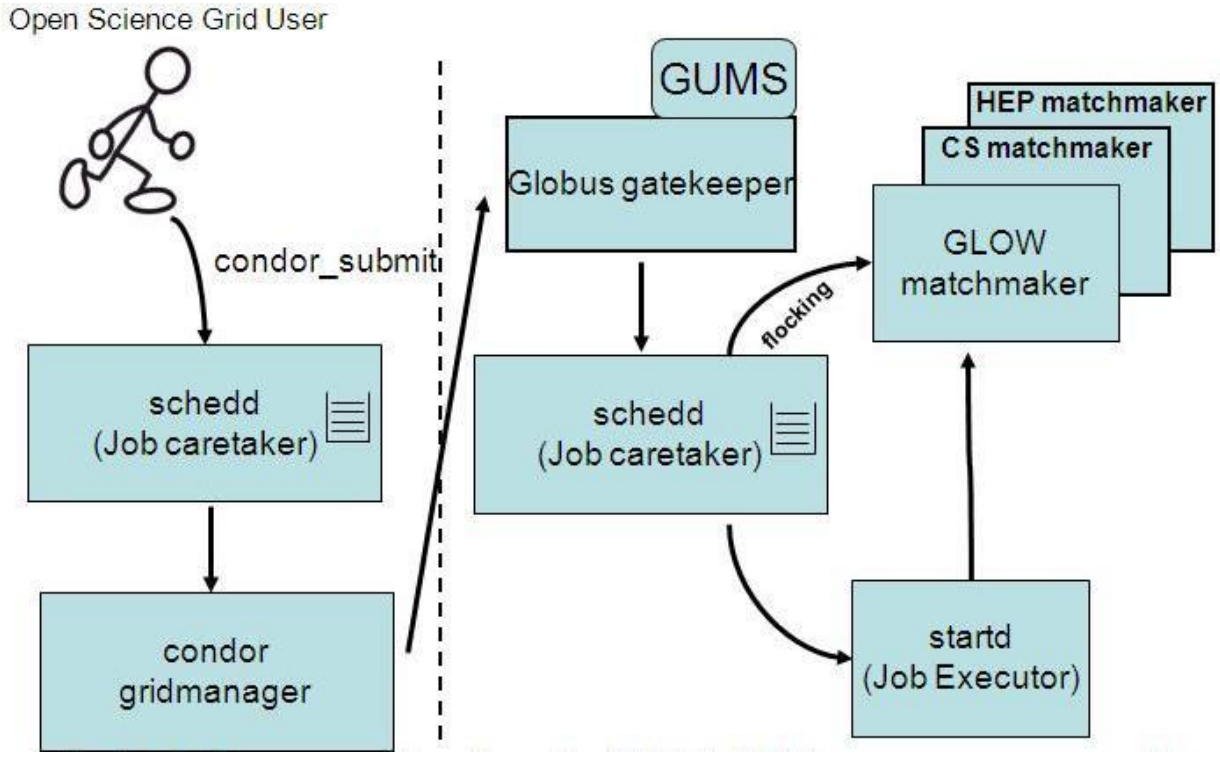

Figure 2-2 UW Madison campus grid with OSG software stack installed

All users in this case have to become members of a particular virtual organization and obtain grid credentials [5]. The credentials are used to submit jobs to the Globus Gatekeeper,

which uses the matchmaker to execute jobs at the appropriate available resources. Adopting this method to implement our campus grid means that all users of GatorGrid will have to enroll with a certificate authority and obtain grid credentials, thus creating administrative hassles and also sacrificing local autonomy. Also, the UW Madison grid uses Condor resource manager in all of its departmental clusters, which is not the case for our campus grid.

The Boiler grid, which consists of a consortium of several Indiana universities, has more than 7700 computers ranging from desktops to powerful research computers. The BoilerGrid uses Condor, which is used to collect idle cycles on all Linux computational resources. All of these resources are scheduled with PBS. When no PBS job is running on a given node, it is free to execute Condor jobs. However, when PBS elects to run a new job on such a node, any active Condor job is immediately checkpointed (if possible) and removed from the node. However, this solution still does not solve the issue of job migration in a manner where most of the resource managers could communicate with each other, migrating jobs from over-utilized clusters to the under-utilized ones. Instead, a new access point (job submission point) is created and jobs are now submitted to a new resource manager altogether.

In [6], a system that utilizes two tools, MOSIX and Globus, to use the computing power wasted in unused student laboratories. This system consists of two layers. In the first layer is a set of machines under the control of one administrator, forming a cluster. The second layer is a group of such clusters. Cluster system software such as MOSIX, OpenMOSIX, OpenSSI, etc. could be used for process migration in the first layer. Integrating the second layer is a typical grid problem solved by the use of the Globus Toolkit, as at that time Globus was well integrated with MPI. Figure 2-3, taken from [6] shows the test bed implemented using this technique. The paper also suggests that the MOSIX clusters are small – up to about 10 machines. The reason is that

the bigger the size of the cluster, the more is the amount of communication and data processing necessary for intelligent migration. Also, if for some reason a cluster fails, all the tasks residing on the dedicated station for that cluster may begin to execute slowly. If more processes are started by Globus, it might freeze the machine, leading to many problems. This renders the solution unsuitable as the various departmental clusters at UF that have more than 10 machines.

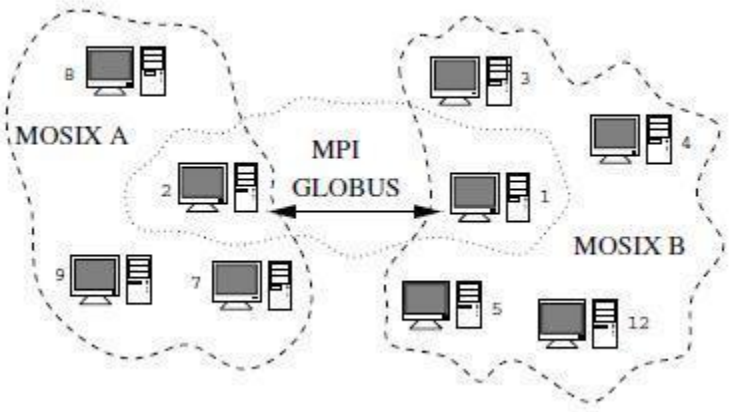

Figure 2-3 An example of how a grid using the two level approach would appear

In [7], a system for aggregating heterogeneous resources from distinct administrative domains into an enterprise-wide compute grid is described. The paper claims that the solution is completely open source, which is an advantage. Although the above-mentioned scheme is a technique to build an enterprise grid, some of the motivations for our work for a campus grid are drawn from [7]. File staging is taken care of by GridFTP and scp in [7]. However, establishing this scheme would mean much change to the way a user submits his/her job scripts. This is not acceptable in our case as we would like to minimize the changes in the way job scripts are submitted and run in the combined compute grid. Instead, we will use distributed parallel virtual file systems, using the Lustre filesystem [10].

This system also uses grid credentials for the various users, similar to the UW Madison grid. This is not desirable. Hence, there is a need to look into other solutions for the user

authentication issue. Currently many clusters are implementing user authentication using a scheme called as an LDAP [14]. Instead of storing user accounts in a file in the file system, user accounts are stored in a directory service provided by OpenLDAP. Unlike local authentication, which is a flat file, the directory service is hierarchical in nature, allowing us to sort and organize our user accounts by location, function, or department. A method to install OpenLDAP in a Fedora Core environment and how the various configuration files need to be configured is provided in [15]. Several methods to integrate the disparate directories are described in [14]. After analyzing the different methods to integrate LDAP directory servers and comparing each method to how well it solved the problem at hand, the loose directory interconnection method was chosen. Figure 2-4 taken from [14], shows this method.

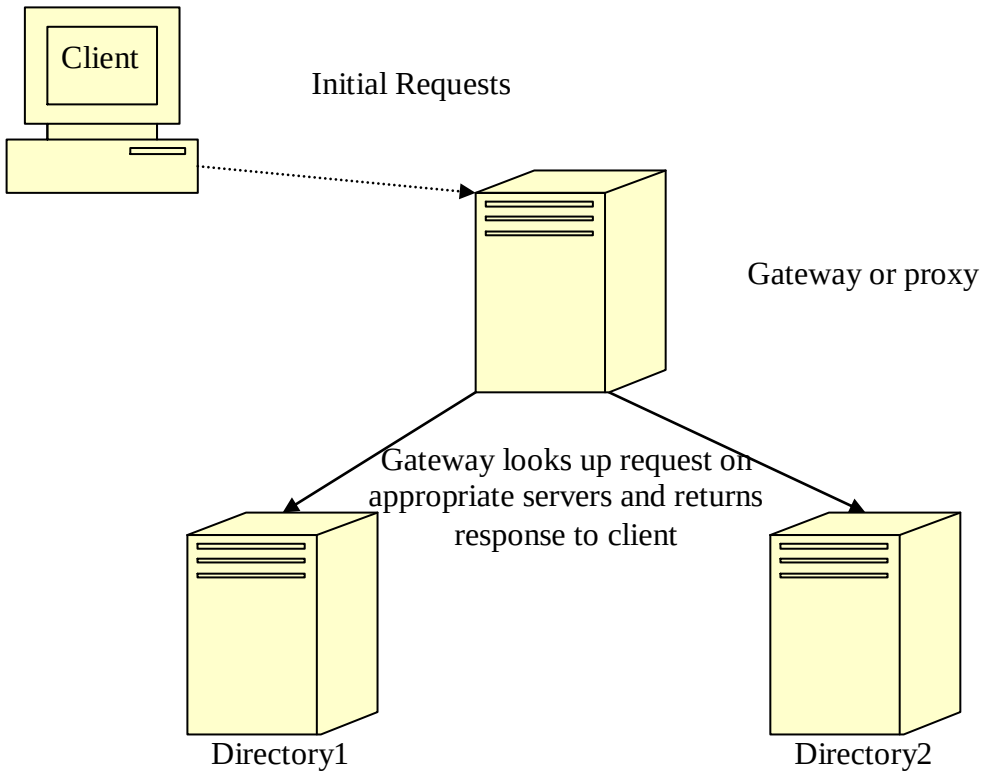

Figure 2-4. Loose directory interconnect method for connecting LDAP servers.

Note that [14] also describes other methods, such as master directory method, directory synchronization method and data harvester method. The master directory method used the idea

of index servers. Although much of the material from [16] could be used to create an index server, much administrative overhead is created for the participating clusters in the form of installing, configuring and maintaining the index servers. The directory synchronization method uses replication to move data from one directory to the other. This method again creates administrative hassles for the participating clusters. For example if QTP research group owns a cluster, and if it decides to add modify or delete a user, it has to inform all the participating clusters of the campus grid that it has changed its directory entries. Then the other clusters also follow suit and replicate the new directories, which is clearly a lot of work. Using the harvester again requires extra configuration work for the participating clusters in the form of adding an events database. None of the above would be necessary if one were to adopt the loose directory interconnection method.

The final problem is that of job migration. In [7], a different Distributed Resource Management System (DRMS) or resource manager was allowed to exist locally as long as there was some trust relationship between the different DRMS. In that paper, a solution based on a master-slave trust relationship is used. The disadvantage with this approach is that new administrative duties involving maintenance of the master system would have to be handled by one of the participating clusters. Instead, we would prefer to establish a peer-to-peer trust relationship. From [17], it was observed that this kind of relationship could be established between two MOAB instances in different clusters and a MOAB instance also recognized a local resource manager in the same cluster, thereby allowing jobs to migrate.

According to [17], there are three types of relationships possible between two MOAB instances, namely Centralized Management, Source Destination Management and Local Management. The Centralized Management relationship establishes a source MOAB instance to

which jobs are submitted. The central MOAB instance has the details of the various resources available across different clusters, which helps it make an intelligent decision. In the Source Destination relationship, a little bit more sovereignty is provided by allowing local job submission. This takes care of the biggest disadvantage of the Centralized Management by allowing jobs to run if the source fails. However, if there are three or more clusters forming the grid, the Source Destination model does not allow utilization of the third cluster. This disadvantage can be overcome if all the MOAB instances collect resource utilization information of all the other clusters and allow jobs to migrate in and out of the cluster where it was submitted. This is the kind of relationship the Local Management model aims at establishing. A diagrammatic illustration of this relationship is shown in the Figure 2-5 taken from [17]. The local management establishes a peer-to-peer MOAB configuration which is desired.

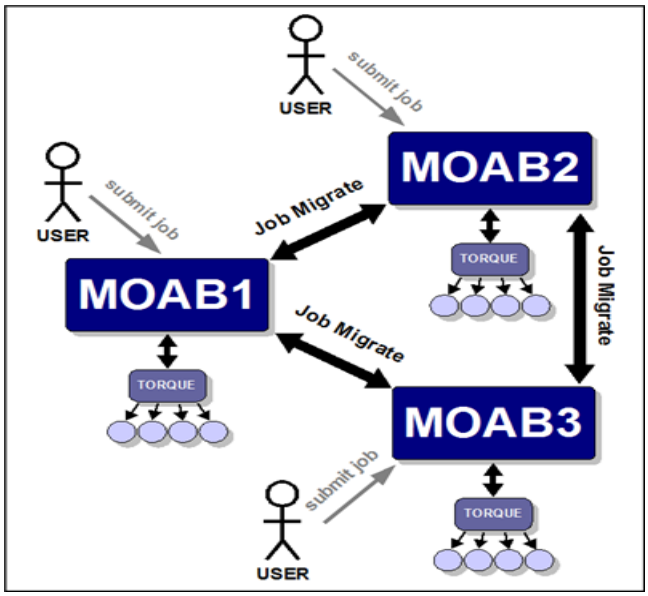

Figure 2-5. Local management

This means that all the administrators of the various domains simply have to maintain their local MOAB and local resource manager configurations and still ensure proper working of the campus grid.

CHAPTER 3
GATORGRID ARCHITECTURE

The GatorGrid environment consists of a set of heterogeneous research clusters. Each of these clusters maintains its own hardware and software, user authentication schemes, job managers, and file systems. Examples of such clusters are the HPC Center in the College of Engineering, Quantum Theory Project group (QTP) in the Physics Department, and the HCS Research Laboratory in the ECE Department. The key requirements of GatorGrid are as follows:

- User authentication across different administrative domains
- Sharing mount point information across clusters to minimize changes to job scripts
- Retaining current access points and job submission points
- Allowing job migration from an over-utilized cluster to an under-utilized one, but still applying local usage policies and QOS's within each cluster
- Maintaining local autonomy

A simplified system architecture of GatorGrid that integrates only two clusters is shown in Figure 3-1. Each cluster is assumed to consist of the following components:

- A user authentication scheme such as LDAP
- A data management scheme such as a distributed file system like Lustre
- A job manager such as the MOAB scheduler
- Various directories that contain data
- Worker nodes, client nodes, or compute nodes on which the actual computing takes place

The client nodes of the two clusters are made to point at a LDAP server configured as a proxy (LDAP proxy). The proxy then checks the appropriate LDAP server and authenticates the user, providing user authentication across domains. MOAB schedulers in each of the clusters are configured with a grid license and the trust relation between the two MOAB schedulers is peer-to-peer. This helps the resource managers to communicate with each other to allow job migration, but still maintaining local usage policies. The Lustre file system in each of the clusters

helps the user to access at the various mount points in different clusters. Each of the key components of the GatorGrid architecture is discussed below.

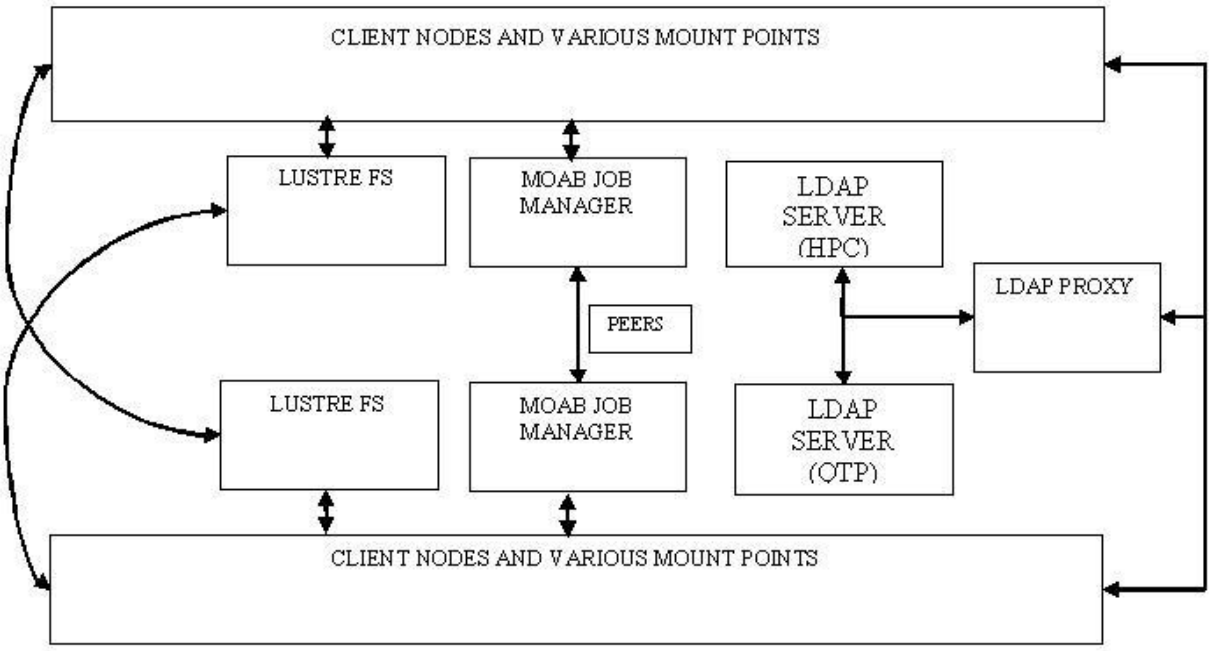

Figure 3-1. Simplified GatorGrid architecture

User Authentication Using LDAP Proxy

LDAP stands for light weight directory access protocol. An LDAP server stores information such as the UIDs and GIDs in a hierarchical fashion. Hence, user authentication can be carried out using LDAP. OpenLDAP is an open source implementation of this protocol. An OpenLDAP server can be configured as a proxy, which is essential for implementing the loose directory interconnection method for integrating disparate directory servers to provide user authentication across domains.

Every LDAP server consists logically of two parts: a frontend and a backend. The frontend speaks the LDAP protocol and contacts the backend upon the client's requests. The backend actually provides the data. Figure 3-2 shows the OpenLDAP architecture.

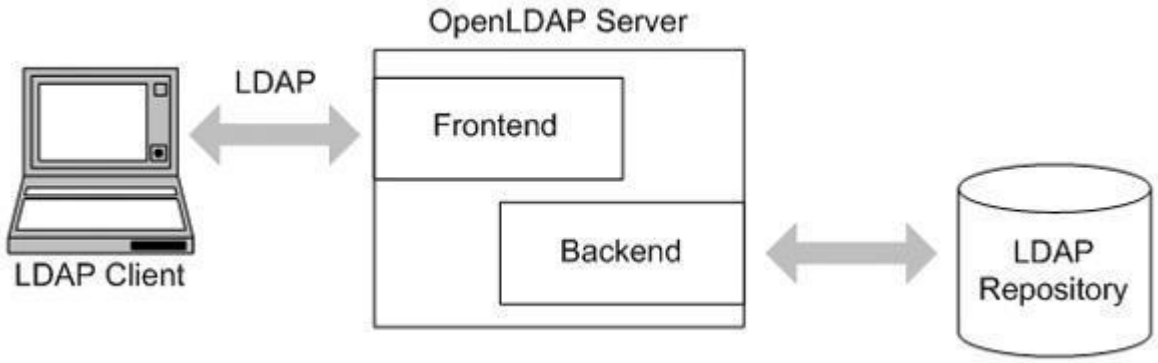

Figure 3-2. OPENLDAP architecture

This architecture offers enormous flexibility. One can access a different data store by simply using a different backend. OpenLDAP itself is shipped with a number of such backends. The most frequently-used ones are the database backends BDB and LDBM. The main backend used in GatorGrid is the "ldap" module. The idea of a proxy is to use the “ldap” backend not to access a repository directly but to contact another LDAP server that holds the data. Figure 3-3 shows this architecture. Multiple LDAP servers can now be contacted by this OpenLDAP proxy to retrieve UIDs/GIDs. This provides authentication for various users across computing domains.

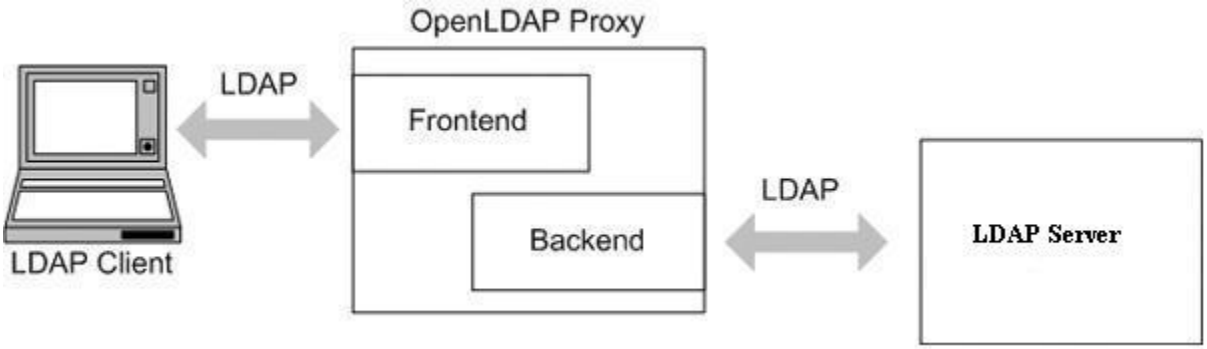

Figure 3-3. OPENLDAP proxy architecture

MOAB Job Manager

For GatorGrid, we needed a job manager that can interface with the local resource manager as well as another instance of a job manager in another cluster through a trust relationship. Also,

this job manager should maintain current access points and apply local QOS policies. These requirements can be satisfied using the MOAB job manager.

A MOAB job manager is an intelligent management middleware that provides simple web-based job management, graphical cluster administration and, management reporting tools. Resource managers such as PBS, Loadleveler, LSF, SGE can be interfaced to this job manager. MOAB is in use currently at the clusters of HPC Center and the QTP. This job manager can recognize another MOAB instance in another cluster as well as a resource manager like Torque, SGE, etc. in the same cluster. Thus, when the current queue is full, jobs can be migrated between clusters and queue waiting times can be reduced. This method is advantageous especially since MOAB has a built-in translator. This allows integrating different clusters having different resource managers. This also implies that the research groups will need to make little changes to the way of submitting jobs to GatorGrid, especially if they want to allow migration. The MOAB instances, when configured as peers, leave all previous access points intact. Thus, users may still continue submitting jobs at the same access points, but able to utilize the vast combined computing power of GatorGrid. Another advantage is the data staging capabilities that are available. A cluster may not want to share the details of the mount points. In this case they may want to utilize data staging to run their jobs. This can also be made possible by configuring the MOAB job managers as peers.

Lustre File System

Sometimes the user's home area is not the place where all the data needed to execute the various jobs are stored. This is because some participating clusters may have quotas for home areas but provide huge shared areas to store data and execute jobs. A distributed file system, with good scalability and performance, can be used to share mount point information between clusters. In an earlier experiment between UF and Florida International University over the

Florida Lambda Rail, the Lustre file system had an I/O of 3GB/s for /ufhpc/scratch and 2 GB/s for /crn/scratch mount points of the HPC cluster. Hence Lustre was chosen as the file system for GatorGrid.

Lustre has a storage and file system architecture suitable for large clusters. It is designed to run over a wide range of network fabrics. Lustre is supported by Red Hat and SUSE flavors of the Linux operating systems in IA-32, IA-64, x86-64 and PowerPC platforms with TCP/IP, Quadrics Elan 3 and 4, Myri-10G, Myrinet-2000, Mellanox, or Infiniband interconnects.

A detailed explanation of the Lustre file system is provided in [10] and further material with respect to its installation configuration is available in [11, 12, 13]. A Lustre file system has three major functional units:

- A single metadata target (MDT) per file system that stores metadata, such as filenames, directories, permissions, and file layout, on the metadata server (MDS)
- One or more object storage targets (OSTs) that store file data on one or more object storage servers (OSSs). Depending on the server's hardware, an OSS typically serves between two and eight targets, each target is a local disk file system up to 8 terabytes (TBs) in size. The capacity of a Lustre file system is the sum of the capacities provided by the targets
- Client(s) that access and use the data. Lustre presents all clients with standard POSIX semantics and concurrent read and write access to the files in the file system.

When a client accesses a file, it completes a filename lookup on the MDS. As a result, a file is created on behalf of the client or the layout of an existing file is returned to the client. For read or write operations, the client then passes the layout to a *logical object volume* (LOV), which maps the offset and size to one or more objects, each residing on a separate OST. The client then locks the file range being operated on and executes one or more parallel read or writes operations directly to the OSTs. With this approach, bottlenecks for client-to-OST communications are eliminated, so the total bandwidth available for the clients to read and write data scales almost linearly with the number of OSTs in the file system. From this discussion it is

not hard to understand that if clients could refer to the MDTs of all the participating clusters, then a user logged into any cluster would be able to access all the data required, stored in different mount points to execute his/her jobs. A sample implementation of Lustre file system is shown in Figure 3-4.

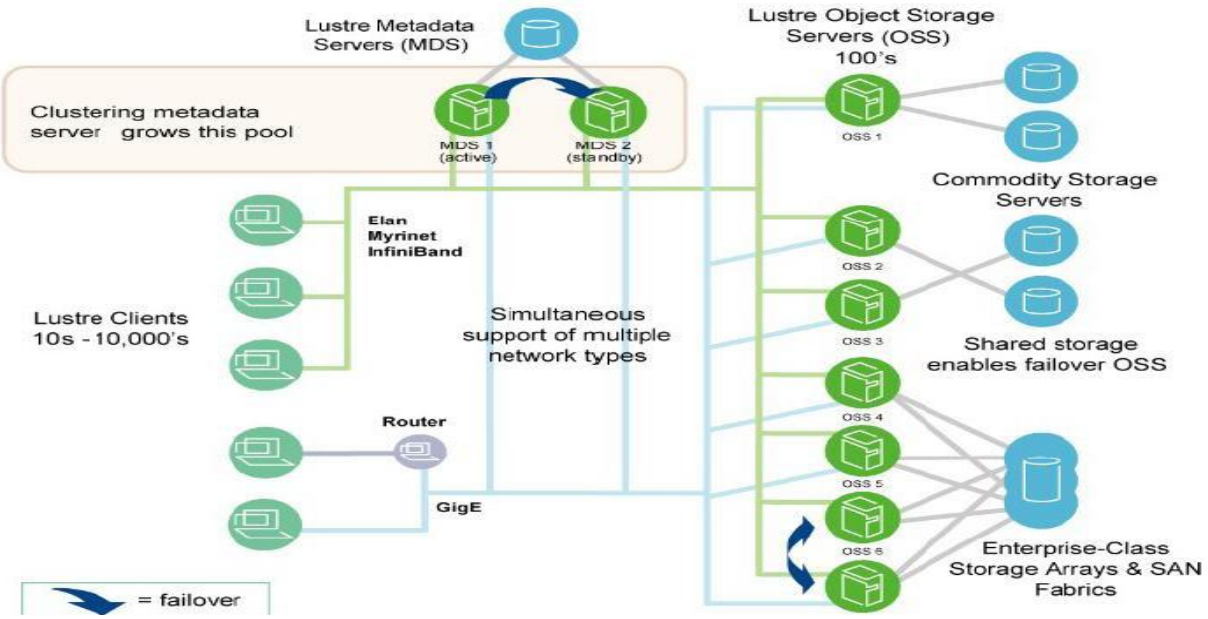

Figure 3-4. LUSTRE sample implementation.

In summary, by using the LDAP proxy to provide user authentication across domains, Lustre file systems to share mount point information, and MOAB job managers to provide job migration and retaining all the access points, GatorGrid provides an architecture for integrating the various research clusters at UF into effective campus grid. Also, by using these technologies, an administrator is allowed to add, delete, or modify local user accounts, manage the jobs running at his cluster and add, delete, or modify storage attached to the cluster. Thus, local autonomy is still maintained and all the requirements outlined for GatorGrid are satisfied.

CHAPTER 4
TESTBED DEVELOPMENT

This chapter describes the key configuration steps and issues one may face while integrating the research clusters into a campus grid. For this test bed, the High Performance Computing Center (HPC) and the Quantum Theory Project (QTP) cluster are used.

The HPC cluster currently allows user authentication using LDAP, uses the Lustre file system for data management, and uses the MOAB job manager and Torque resource manager to manage the various resources and jobs. On the other hand, although the QTP cluster uses MOAB already, it uses files for user authentication. Also, we assume that the HPC cluster is willing to share information about the various mount points, whereas QTP does not. It must be noted that these clusters are production systems and hence changes cannot be made directly to them. Instead, a test bed is to be configured with nodes representing both HPC and QTP clusters.

Configuring and Deploying LDAP Proxy

For the test bed development, three nodes were used, namely moab-hpc, ldap-qtp and osg. Both moab-hpc and ldap-qtp have two AMD Opteron processors running at 1.8GHz and 2GB of RAM. The node osg has 4 dual-core AMD Opteron processors running at a clock frequency of 2.2 GHz and 4GB of RAM. The node moab-hpc is representative of the HPC cluster and the node ldap-qtp is representative of the QTP cluster. These were nodes on the local network invisible to the outside world. However, the third node, osg, is a publicly visible node and is used to host the LDAP proxy server. Also, it is assumed that the Lustre file system is already present and used by both moab-hpc and ldap-qtp. Hence, a discussion on providing Lustre capabilities is not dealt with here.

Testing the Idea of LDAP Proxy Using Virtual Machines

Note that the three nodes used for building the test bed were actually production worker nodes of the HPC and QTP clusters. Hence, in order to not disrupt the production system, we verified the working of the LDAP proxy before it is implemented in the test bed. This verification was performed using User Mode Linux virtual machines. The virtual machines that were used had a Fedora Core 4 file system. Figure 4-1 shows the setup for the experiment.

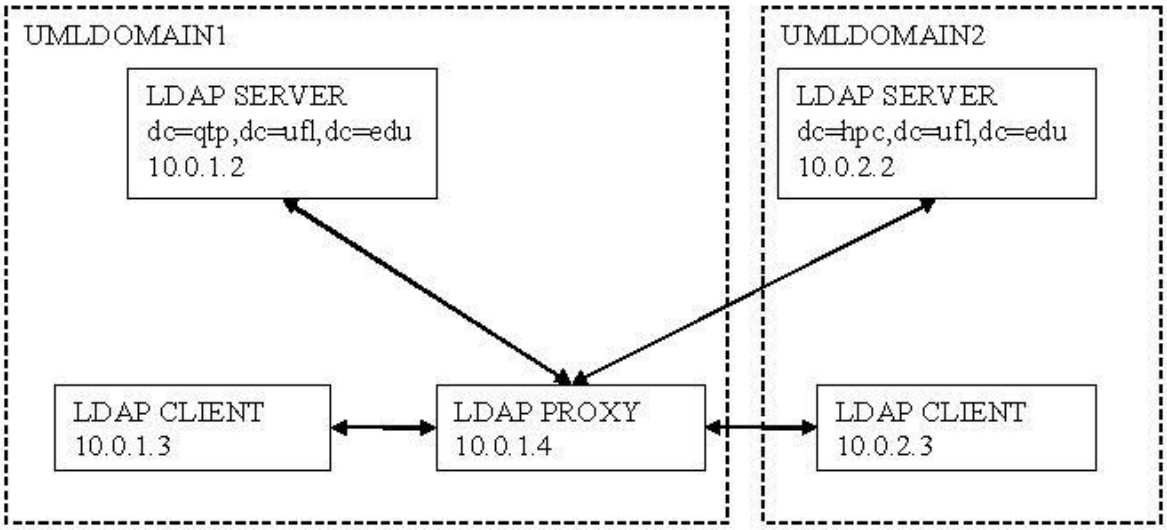

Figure 4-1. Diagrammatic illustration of virtual machine validation of the LDAP proxy method

As shown in the Figure 4-1, five VM's were used. Three of them belonged to one domain (umldomain1) and the other two belonged to the umldomain2. One of the three virtual machines of umldomain1 was configured as the proxy and was made accessible by virtual machines belonging to both domains, while the other two were configured as a client server LDAP pair with a user siddhu. The two virtual machines of the domain umldomain2 were setup as the client server LDAP pair with user prescott. The clients of both domains were then made to point to the LDAP proxy server. Also, the following lines were added to the /etc/ldap.conf file in the clients of both the domains.

nss_base_shadow ou=People,dc=qtp,dc=ufl,dc=edu?one

nss_base_passwd dc=hpc,dc=ufl,dc=edu?one?objectClass=posixAccount

nss_base_passwd dc=qtp,dc=ufl,dc=edu?one?objectClass=posixAccount

nss_base_group ou=Group,dc=hpc,dc=ufl,dc=edu?one?objectClass=posixGroup

nss_base_group ou=Group,dc=qtp,dc=ufl,dc=edu?one?objectClass=posixGroup

By doing so, it was shown that the retrieval of user and group information from both the servers by any client was allowed, thus verifying the LDAP Proxy method.

Migrating QTP's Flat File Based Authentication to LDAP

The next step was to migrate the flat files of the QTP cluster to LDAP. The new QTP cluster's LDAP server was configured in the node ldap-qtp. This migration was done with the migrate scripts that were shipped with OpenLDAP. However, there was an issue while migrating QTP's automount maps using these migrate scripts. The various automount maps of QTP have a hierarchical structure. The entries in auto.master point to the files in /etc/, which have entries showing the directories to be mounted. Figure 4-2 shows how the automount maps are looked up for the user "as". When we use the script migrate_automounts.pl on the auto.master it may result in the file having entries as shown below

dn: nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: top

objectClass: nisMap

nisMapName: auto.master

dn: cn=/ufl/qtp/admin,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /ufl/qtp/admin

nisMapEntry: /etc/qtp-admin rw,hard,intr,timeo=15,actimeo=1

nisMapName: auto.master

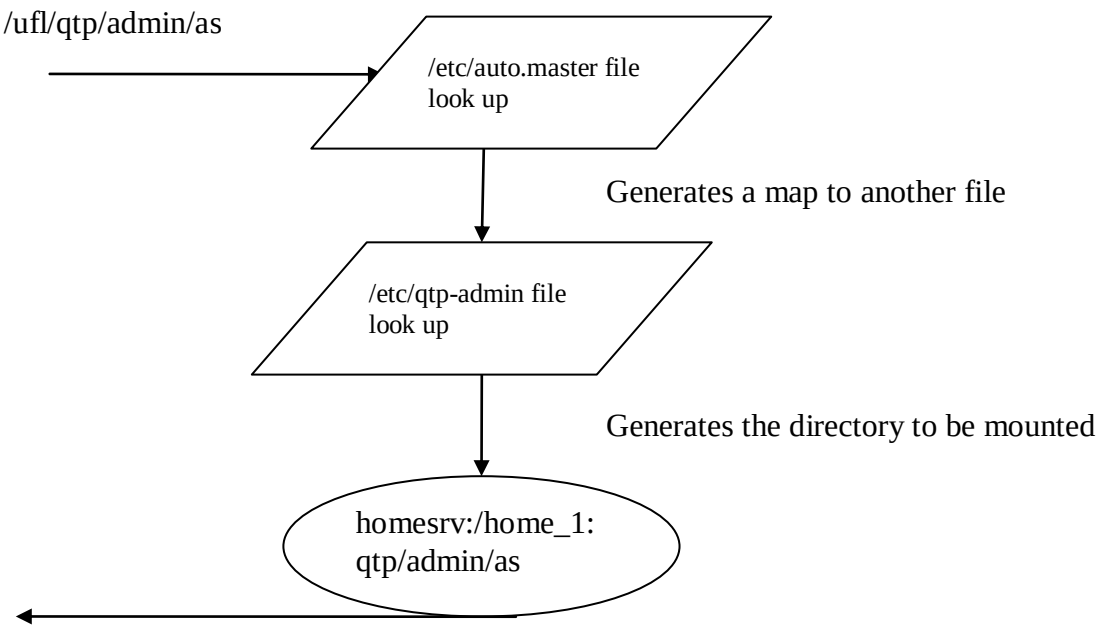

Figure 4-2 Existing automount maps at QTP

These are not the desired entries for the LDAP. This is because the top level entry of the LDAP returns a request for a file. For example, assume that user “as” logged in using the LDAP. To load his home directory, LDAP looks into the top of the hierarchy of automount maps it created using the above generated entries. The LDAP based on the entries looks for /etc/qtp-admin file rather than using another entry stored in it. The situation is diagrammatically illustrated in Figure 4-3. This is not desirable as client nodes would try to look for the local file which is non-existent.

Hence, a simple script is written to change the entries as shown below.

dn: nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: top

objectClass: nisMap

nisMapName: auto.master

dn: cn=/ufl/qtp/admin,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /ufl/qtp/admin

nisMapEntry: nisMapName=qtp-admin,dc=qtp,dc=ufl,dc=edu

nisMapName: auto.master

This forces the LDAP to look for an entry nisMapName=qtp-admin,dc=qtp,dc=ufl,dc=edu and, from that entry, the home directory of "as" can be mounted. These corrected LDAP entries are then imported to LDAP. The final look up is illustrated diagrammatically in Figure 4-4.

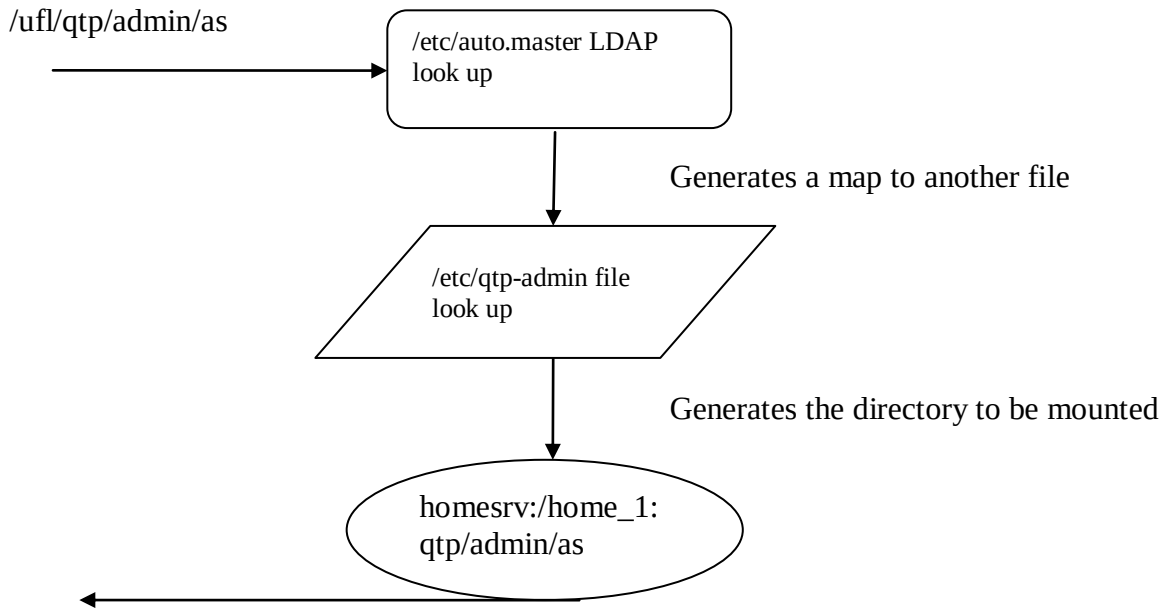

Figure 4-3. Wrong maps generated by the migrate script

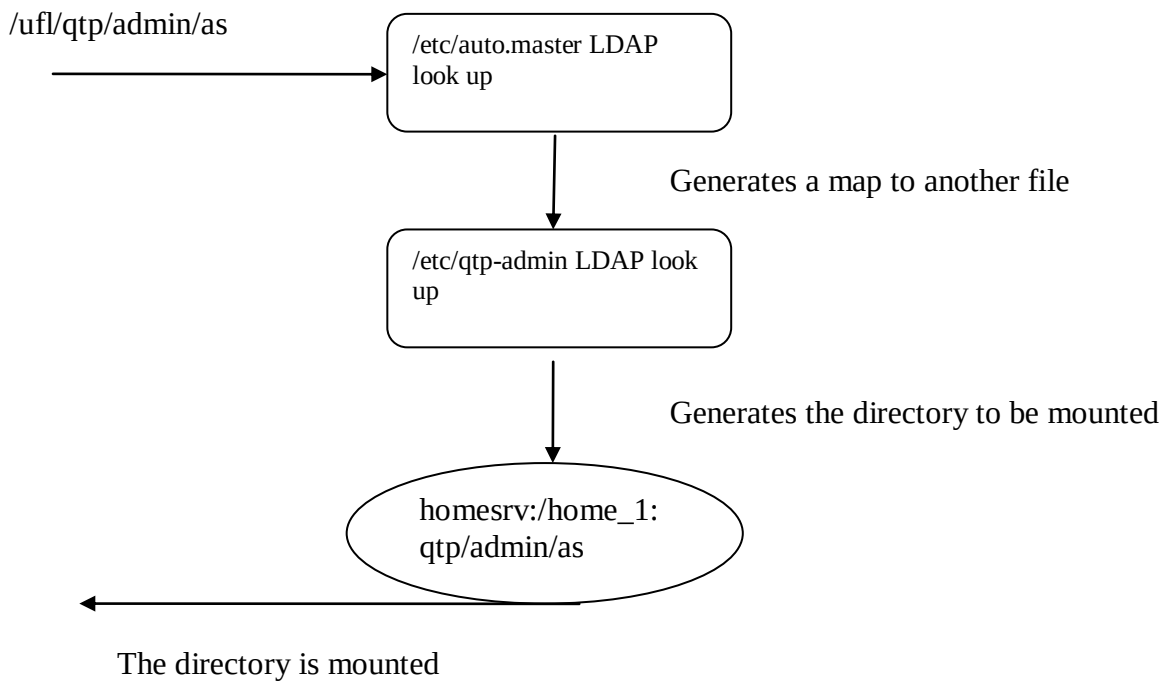

Figure 4-4. Desired maps

LDAP Proxy Server Auto-mount Issues

For the next step, the LDAP Proxy was configured in the node osg. However, there was an issue with the automount maps. The home areas belonging to the HPC cluster were not being mounted by ldap-qtp using the automount maps. In general, it is necessary that the LDAP client libraries are installed on all client nodes and that /etc/openldap/ldap.conf has the BASE and URI set to the appropriate LDAP server to access the user accounts and automount maps. In our case, we needed the URI to point to the proxy server and multiple bases are accepted. However, multiple bases could not be provided in the /etc/openldap/ldap.conf file. If multiple bases were provided, only the first base specified was being accessed, which was the base corresponding to the QTP cluster's LDAP server. A solution to this problem was to use the /etc/sysconfig/autofs configuration file. This file, in AUTOFS version 5, was used to specify the schema (rules governing LDAP entries) that a cluster wanted to implement for its automount maps. There is another entry called search base, about which the configuration file specified the following:

SEARCH_BASE - base dn to use for searching for map search dn.

Multiple entries can be given and they are checked

in the order they occur here.

However, when multiple entries were provided, this also did not behave as expected. Hence, the work around for accessing HPC cluster's directories through LDAP from QTP was to add entries to QTP cluster's LDAP server such that a reference would be sent to QTP's LDAP server. The map entry returned is an LDAP look up to the LDAP server of HPC cluster, thereby allowing home areas to be mounted. The work around is shown in Figure 4-5.

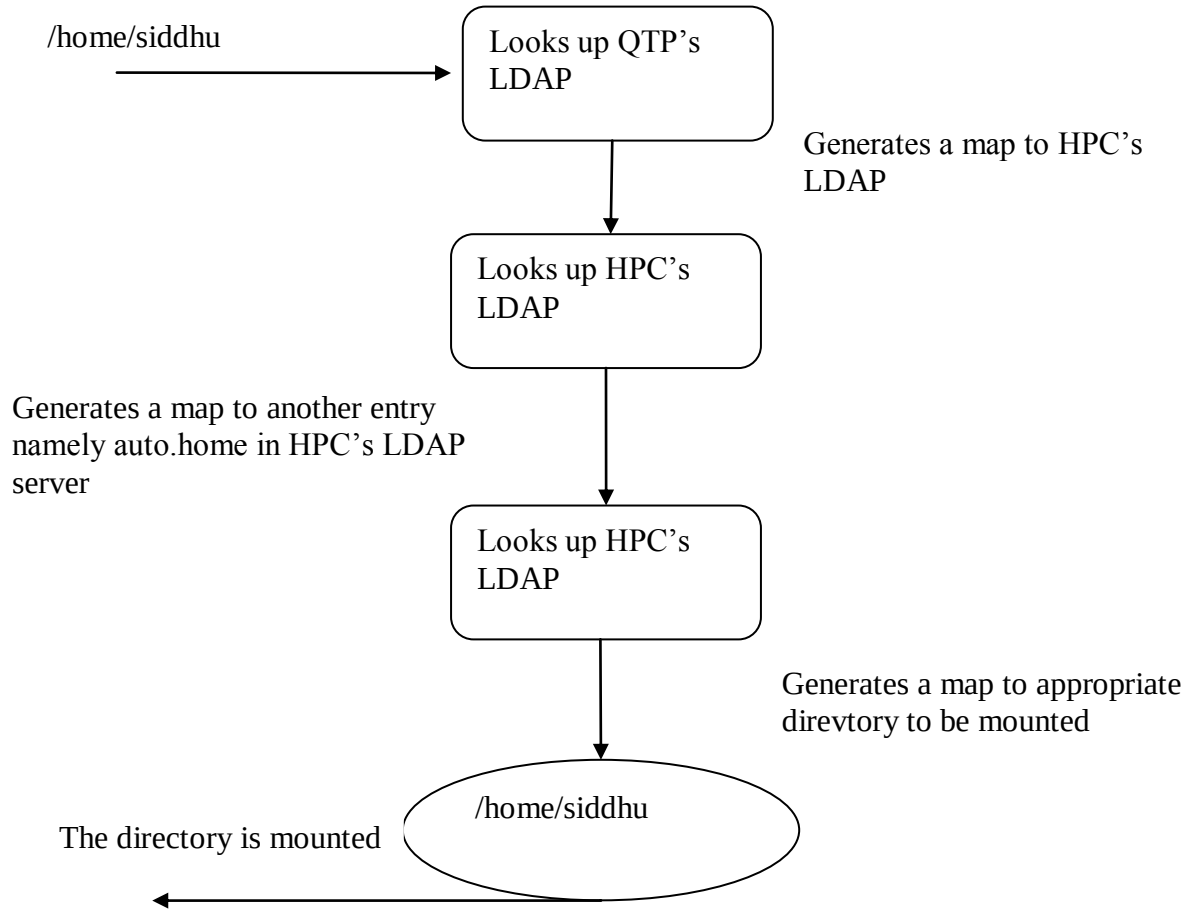

Figure 4-5. Work around for the LDAP proxy automounts issue.

The LDIF entries added to QTP cluster's LDAP server to effect this change are:

dn: cn=/grid,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /grid

nisMapName: auto.master

nisMapEntry: ldap:nisMapName=auto.grid,dc=hpc,dc=ufl,dc=edu -rw,async,nfsvers=3,udp

dn: cn=/scratch,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /scratch

nisMapName: auto.master

nisMapEntry: ldap:nisMapName=auto.scratch,dc=hpc,dc=ufl,dc=edu -rw,async

dn: cn=/apps,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /apps

nisMapName: auto.master

nisMapEntry: ldap:nisMapName=auto.apps,dc=hpc,dc=ufl,dc=edu -ro,async,nfsvers=3,udp

dn: cn=/dist,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /dist

nisMapName: auto.master

nisMapEntry: ldap:nisMapName=auto.dist,dc=hpc,dc=ufl,dc=edu -rw,async,nfsvers=3,udp

dn: cn=/home,nisMapName=auto.master,dc=qtp,dc=ufl,dc=edu

objectClass: nisObject

cn: /home

nisMapEntry: ldap:nisMapName=auto.home,dc=hpc,dc=ufl,dc=edu

nisMapName: auto.master

Configuring MOAB Job Manager in a Peer-to-Peer Fashion

To enable the exchange of workload and resource status information and to distribute jobs and data among clusters, the resource managers need to be configured to communicate with each other under an established trust relationship. In GatorGrid, the different MOAB instances are configured to communicate with each other under a peer-to-peer trust relationship in order to provide local autonomy and minimize extra administrative duties.

There are four steps in establishing a peer-to-peer grid configuration. The first step is to have MOAB installed and configured on both clusters with a grid license. Only MOAB version 5.2.4 or higher can be used for this purpose. Older versions do not allow jobs to be migrated to the routing queues of other clusters. Next, a new line is added to the MOAB configuration file (moab.cfg) that corresponds to the MOAB scheduler in the other cluster. Also, for this step it is suggested that one uses the name of the cluster as the name of the scheduler. This is followed by step 3, which deals with the creation of a file called moab-private.cfg on each cluster, in the same location as moab.cfg. The remote MOAB scheduler from the moab.cfg file was specified, and corresponding secret key and administrative privileges are set in this file. The final step was to restart both MOAB job schedulers and run mdiag -R -v to see the status of the grid

communication. If everything was configured properly, we will be able to see the state as active for both the clusters. If there are problems, error messages generally provide information as to why the communication is failing. A sample moab.cfg that was actually used for ldap-qtp is shown below for establishing a peer to peer configuration.

```
#######################################################################################
        #
        #  Moab Configuration File for moab-5.2.4
        #
        #  Documentation can be found at
        #  http://www.clusterresources.com/products/mwm/docs/moabadmin.shtml
        #
        #  For a complete list of all parameters (including those below) please see:
        #  http://www.clusterresources.com/products/mwm/docs/a.fparameters.shtml
        #
        #  For more information on the initial configuration, please refer to:
        #  http://www.clusterresources.com/products/mwm/docs/2.2initialconfig.shtm
        #
        #  Use 'mdiag -C' to check config file parameters for validity
        #
#######################################################################################

        SCHEDCFG[ldap-qtp]    MODE=NORMAL SERVER=ldap-qtp.ufhpc:42559
        ADMINCFG[1]           USERS=root
```

```
TOOLSDIR            /opt/moab/tools
LOGLEVEL            5

#############################################################################
#
#  Resource Manager configuration
#
#  For more information on configuring a Resource Manager, see:
#  http://www.clusterresources.com/products/mwm/docs/13.2rmconfiguration.shtml
#
#############################################################################

RMCFG[ldap-qtp]     TYPE=PBS
RMCFG[ldap-qtp]     SUBMITCMD=/usr/bin/qsub
RMCFG[moab-hpc]   TYPE=moab      SERVER=moab-hpc:42559
```

The moab-private.cfg in lap-qtp had the following entry.

```
CLIENTCFG[RM:moab-hpc]  KEY=12345  AUTH=admin1
```

By following the procedure outlined above, one can integrate clusters in a peer-to-peer fashion. However, it may so happen that one of the participating clusters does not want to share their mount point information. Under such circumstances MOAB data staging can be used. The executables and data files required for execution could be staged into a shared area of the intended cluster for execution and the job could then be executed. Setting up MOAB data staging is a three-step process. The first step is the setting up of SSH keys between the main MOAB

server (as root) and the data storage machine. The second step involves editing the moab.cfg file to set up the data staging machine as another resource manager. This step is also used to link the resource manager to other resource managers and clients who will be using it. For the third step, the tools/config.dstage.pl and dataspaces.tab files need to be removed for the new changes to take effect. The configuration was done in both ldap-qtp and moab-hpc. Finally, it was tested using the msub command with data staging options. An example test command is shown below:

msub –W x='mstagein=file:///home/siddhu/mdate.pl, ssh://moab-hpc/tmp/' testjob.job

The extra configuration added to ldap-qtp's moab.cfg is shown below:

```
RMCFG[ldap-qtp]      TYPE=PBS
RMCFG[ldap-qtp]      SUBMITCMD=/usr/bin/qsub
RMCFG[ldap-qtp]      DATARM=data
RMCFG[moab-hpc]      TYPE=moab
RMCFG[moab-hpc]      SERVER=moab://moab-hpc.ufhpc:42559
RMCFG[moab-hpc]      DATARM=data
CLIENTCFG[DEFAULT]      LOCALDATASTAGEHEAD=moab-hpc
RMCFG[data]                  TYPE=NATIVE RESOURCETYPE=STORAGE
RMCFG[data]          VARIABLES=DATASPACEUSER=SIDDHU, CACHE=/tmp
RMCFG[data]                  SERVER=ldap-qtp
```

CHAPTER 5
EXPERIMENTAL RESULTS

This chapter describes a series of experiments, which were performed to evaluate the test bed implementation based on three characteristics:

- *Functioning of the test bed*. Five major requirements have to be verified, namely user authentication across domains, ability to access the different mount points of both clusters, job migration between clusters, application of local QOS policies to local clusters, and maintenance of local autonomy.
- *Reliability of the test bed*. In a production environment, often due to technical issues, some nodes go offline. Thus, it is of interest to a cluster administrator to observe what happens to a job when it migrates to the other cluster, but the node intended for execution in that peer cluster goes offline.
- *Effect on queue times*. In a grid scenario more processors are made available to run jobs submitted to a cluster. This could result in a reduction in queue times, which needs to be verified.

Experiment for Test Bed Verification

As mentioned above, in Chapter 3, five major requirements of the system were identified. They are user authentication across domains, ability to share mount point information among clusters, job migration between clusters, application of local QOS policies to local clusters, and maintenance of local autonomy. The requirement of local autonomy need not be verified since changes to the LDAP server, Lustre file system, or MOAB job manager can only be done by the administrators of the respective clusters even after performing the grid configurations. The ability to apply local QOS policies is verified indirectly in other experiments, where multiple users are used to submit jobs with QOS attached to them. This leaves us with the task of verifying three of the above requirements, namely user authentication across both the clusters, ability to access file systems of both the clusters, and job migration between the clusters.

If two users, one belonging to each of the clusters, are able to log in and view their home areas on peer clusters, it would mean that the problem of single user sign-on and the ability to

access both file systems have been solved. Two users, siddhu and bwang, belonging to the HPC and QTP clusters, respectively were used for verification. First, as siddhu the accessibility of his home directory is checked and later as bwang, the accessibility of his home directory was verified. The sample output is provided below.

```
[root@ldap-qtp ~]# su siddhu
[siddhu@ldap-qtp root]$ cd
[siddhu@ldap-qtp ~]$ pwd
/home/siddhu
[siddhu@ldap-qtp ~]$[siddhu@ldap-qtp ~]$ ls
callin.sh              ldap-qtp.final.txt          moabqsub.txt
cluster.tar            ldap-qtp.moab.log           mse.pl
condor_job             login.lib                   ms.pl
date1.pl               mdate.pl                    msr.pl
date.pl                mfinal.txt                  ne.txt
ffmnew1.txt            mg1.txt                     new
ffmnew2.txt            mg2.txt                     new1.txt
final.txt              mg3.txt                     new2.txt
fmnew1.txt             mmfinal.txt                 newap
fmnew2.txt             mmmfinal.txt                new.txt
foo                    mmmmfinal.txt               qm1.txt
foo.out                mmoabmub.txt                qmg1.txt
gram_job_mgr_12538.log  mnew1.txt                  qnew.txt
gram_job_mgr_12842.log  mnew1.txt.old              qs.pl
```

```
gram_job_mgr_13916.log  mnew2.txt                qtime.pl
gram_job_mgr_14163.log  mnew2.txt.old            second_job
gram_job_mgr_14507.log  mnew.txt                 sidd_test
gram_job_mgr_14841.log  mnew.txt.old             simple
gram_job_mgr_15139.log  moab-hpc.7747.e          simple.c
gram_job_mgr_15660.log  moab-hpc.7747.o          simple.error
gram_job_mgr_15913.log  moab-hpc.7748.e          simple.log
gram_job_mgr_16161.log  moab-hpc.7748.o          simple.out
gram_job_mgr_16600.log  moab-hpc.7749.e          submit.txt
gram_job_mgr_17214.log  moab-hpc.7749.o          subm.pl
gram_job_mgr_18348.log  moab-hpc.final.migrated.txt  subq.pl
gram_job_mgr_25770.log  moab-hpc.final.msubed.txt    test
gram_job_mgr_3420.log   moab-hpc.final.txt           testjob.job
jasper-1.900.1          moab.log                 testjob.job~
keep.ap.26-10-08        moabmsubonly.txt         testjob.job1
ldap-qtp.4388.e         moabmsub.txt             usercert.pem
ldap-qtp.4388.o         moabmub.txt              userkey.pem
[siddhu@ldap-qtp ~]$ exit
exit
[root@ldap-qtp ~]#

[root@ldap-qtp ~]# su bwang
bash: /usr/local/lib/global.shrc: No such file or directory
```

```
[bwang@ldap-qtp:/root/ ]$ cd
[bwang@ldap-qtp:~/ ]$ pwd
/ufl/qtp/kmm/bw
[bwang@ldap-qtp:~/ ]$ ls
anis2_1.hin  divcon.pdb          profile          try1.pdf
anis2_2.hin  Getting_Started.pdf  QTTFQ.hin        try1.ps
anis2_3.hin  mail                Sanibel_2007.ppt  UF_COM_Poster_Mar_15_07.ppt
anis2_4.hin  M_OH.hin            shifts           vgalusr1.vr
anis_2.HCS   PA_NMA_2.hin        test.txt         WIND
anis_2.hin   Parectadial_R.hin   test.txt~        WINDOWS
anis_2.xyz   Parectadial_s.hin   tfx.hin          wukong
bwang        P_OH.hin            try1.log         WWW
[bwang@ldap-qtp:~/ ]$ exit
exit
```

In addition, we used the command “df” along with the option “-h” to observe the directories mounted in ldap-qtp. The HPC cluster’s file systems, namely the /scratch, /crn, /hpcdata, were mounted, thus verifying user authentication and sharing of mount point information between clusters. The sample output shows it below.

```
[root@ldap-qtp ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda3              33G   12G   20G  37% /
/dev/hda1              99M   23M   71M  25% /boot
tmpfs                1006M     0 1006M   0% /dev/shm
```

192.168.1.24:/scr_2 7.5T 1.7T 5.9T 22% /scr_2

10.13.16.40@tcp:/ufhpc

28T 24T 4.2T 86% /ufhpc/scratch

10.13.31.209@tcp:/crn

81T 40T 42T 49% /crn/scratch

10.13.31.221@tcp:/hpcdata

4.1T 2.1T 2.1T 51% /ufhpc/hpcdata

[root@ldap-qtp ~]#

To verify job migration, two simple sleep jobs were submitted. One of the jobs was submitted to the ldap-qtp cluster while the other was forced to migrate to moab-hpc. Job execution in ldap-qtp and job execution due to migration in moab-hpc by watching the queues using showq were observed. Further proof of job migration is observed in experiments described below.

Experiment for Test Bed Reliability

The main objective of this experiment is to observe what happens to a job if it has already migrated to the other cluster but the node intended for executing the job goes offline. In this experiment, the user siddhu is made to submit five sleep jobs to moab-hpc. The five submitted jobs were monitored continuously and when one of them migrates, the node to which the job migrated is made to go offline. When the queue at ldap-qtp (the cluster to which the job migrated) was observed, we could observe the migrated job waiting. When the node was brought back online, the idle job executed. This demonstrates that the setup is capable of reliably executing migrated jobs. In future versions of GatorGrid, consideration would be given to moving the migrated jobs back and executing them at the original cluster, as this might reduce the queue time, rather than waiting for the node to come back online.

Experiment on Effect on Queue Times

Queue time reduction is verified using two experiments: single-user submission and multi-user submission with QOS. In the first experiment, a single user is made to submit jobs in a cluster scenario and then in a grid scenario. In the cluster scenario, the user submits jobs to the resource manager directly and then through the MOAB job manager. This is used to determine if submitting to the job manager resulted in additional queue time. In the grid scenario the jobs are submitted to the MOAB job manager, but with the peer-to-peer grid configuration turned on. This is used to determine the effect on queue time due to job migration.

In the second experiment, a more realistic case is considered. Both clusters are assumed to have local fair share policies implemented by local QOSs attached to the jobs submitted based on the users or groups submitting the jobs. Also, instead of a single user submitting a job, three users are made to submit jobs in an interleaved manner and the effect on queue times of their jobs due to job migration is observed.

Single-User Submission

The user siddhu was used for this experiment. The MOAB configuration file did not have any fair share configurations when this experiment was carried out. The user siddhu was made to submit 1000 jobs to moab-hpc three different times. In the first case, the jobs were submitted using the qsub command to the Torque resource manager. These jobs are labeled as qsub'ed jobs in Figure 5-1. In the second case, the jobs were submitted to the cluster using the MOAB submission command msub. These jobs are labeled as the msub'ed jobs in Figure 5-1. The 1000 jobs were submitted for the third time, using msub again, but with the peer-to-peer grid configuration turned on (so that jobs can migrate). These jobs are labeled as grid jobs. In all three cases, the other cluster (ldap-qtp) was given 500 local jobs at the same time. Simple scripts were

then used to identify the time at which a job was loaded into the queue and the time at which a job successfully started execution.

A graph of number of jobs versus queue time is shown in Figure 5-1 for the three cases. From the graph, it was observed that submitting to the MOAB job manager (Case 2) does not add any additional queue time. In additional, a clear decrease in queue times is observed in the case of jobs submitted in the grid scenario (Case 3) when compared to the cluster scenario. The reason is that, after job migration, four cores (2 from moab-hpc and 2 from ldap-qtp) are available for executing the last 500 jobs from moab-hpc.

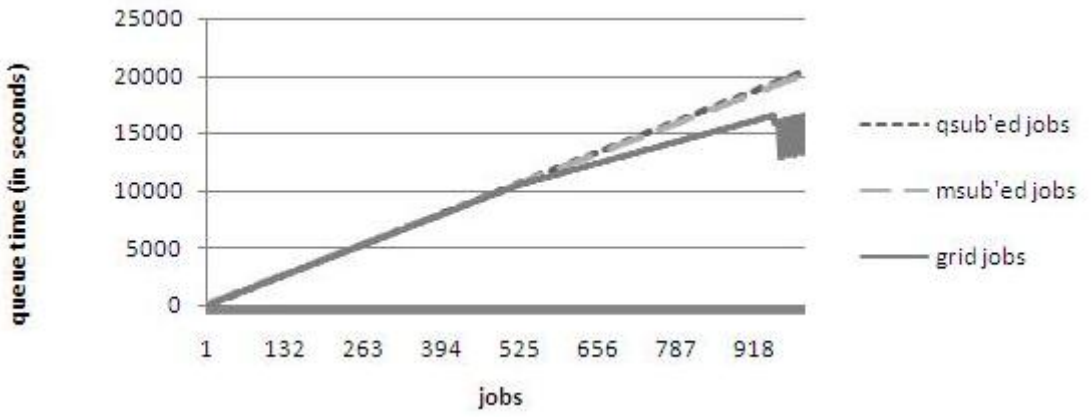

Figure 5-1. Queue Time statistics for single user job submission

All the jobs submitted were simple 40 sec sleep jobs. Hence, 1000 jobs would take approximately 40000 seconds on a single core. But, moab-hpc had 2 cores in which jobs could be run simultaneously. Hence the expected queue time would be approximately 20000 seconds, which was observed. Similarly, ldap-qtp had 2 cores. Hence, for 500 40-second sleep jobs we observed an expected peak queue time of approximately 10000 seconds.

In the grid case, four cores are available for executing the last 500 jobs from moab-hpc. Thus, we expect the maximum queue time to be approximately 15000 seconds if there is no overhead from job migration. However, we observed a peak time of approximately 16480

seconds. From this, one can conclude there is indeed an observable cost associated with job migration. But, the point to note is that HPC workloads are rarely as low as 40-seconds. If bigger jobs running for hours were submitted, this cost of job migration becomes negligible.

Multiple User Submission with QOS

In the second experiment, the cluster represented by moab-hpc was assumed to be sponsored by two departments: Electrical Engineering and Physics. The user siddhu was assumed to belong to the Electrical Engineering Department while users taylor and prescott were assumed to belong to the Physics Department. The cluster represented by ldap-qtp is assumed to be sponsored by other groups. It is also assumed that the cluster utilization does not go beyond 90% in ldap-qtp. Hence, the remaining 10% is available for use in the campus grid.

These changes are incorporated by adding two QOS classes to moab-hpc: ee and phys. The QOS class ee is given 40% of the CPU time on moab-hpc, with an increase in priority should the usage drops below 40%. The QOS class phys is given 60% of the CPU time with a reduction in priority if usage becomes higher. Also, taylor's jobs are assumed to be more critical and hence a higher priority is assigned to those jobs. However, the configuration makes sure this preference between users does not affect the cluster utilization between the two QOS classes. The ldap-qtp cluster divides the 90% of the CPU time equally among its groups and defines a default QOS class grid for the remaining 10% of the time. A job that migrates from moab-hpc to ldap-qtp is not owned by any of the local groups of ldap-qtp. Hence, the default QOS of grid is assigned to it.

Once this setup has been established, user siddhu is made to submit 300 jobs, user taylor is made to submit 350 jobs and user prescott is made to submit 350 jobs. Submission is made in a cluster scenario first (Case 1) and again in the grid scenario (Case 2 where job migration is allowed). In both cases, 500 local jobs are also run in the other cluster. Simple scripts were then

used to identify the time at which a job was loaded into the queue and time at which a job successfully started execution.

A graph of number of jobs versus queue time is shown in Figure 5-2 for taylor's jobs. It was observed that in both cases, queue times were almost the same (the "thick" grey plot indicating grid jobs are running close to the darker plot which are the queue times in the cluster scenario). This is because taylor's jobs were given higher priority than those of prescott's and because of the site-wide fair share policy in place, some of the jobs of siddhu were also executed. So, by the time all the jobs of taylor went towards completion, there were no free processors available yet for job migration, as 500 local jobs were still executing in ldap-qtp.

A graph of number of jobs versus queue time is shown in Figure 5-3 for siddhu's jobs. Since this user has only 40% of the cluster time to run his jobs, some of his jobs were still waiting at the time the 500 local jobs in ldap-qtp have finished. Hence, some of siddhu's jobs can be migrated to ldap-qtp for execution. As a result, it was observed that the total queue time for Case 2 is reduced from 16246s to 13539s, as compared to Case 1 for this user.

A graph of number of jobs versus queue time is shown in Figure 5-4 for prescott's jobs. The jobs submitted by this user can begin execution only after all the local jobs in ldap-qtp have finished. Hence, it was observed that that the total queue time for Case 2 is reduced from 20,169s to 16543s, as compared to Case 1 for this user.

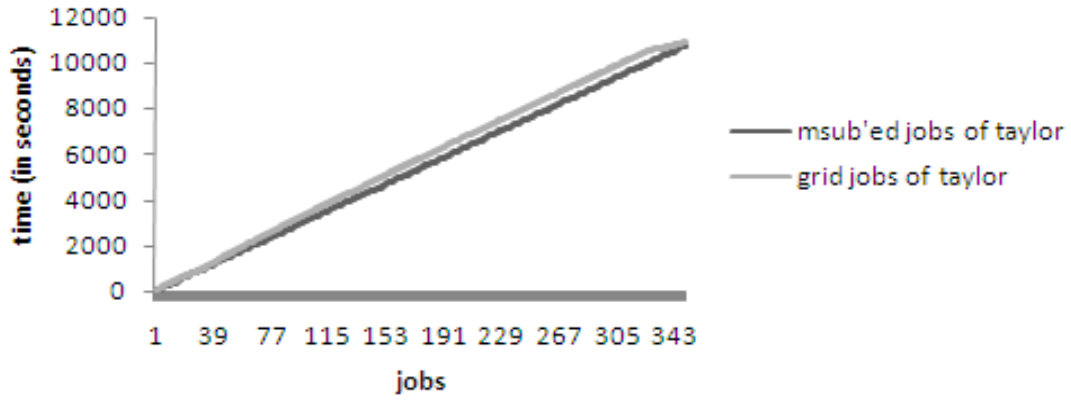

Figure 5-2. Queue times for Taylor's jobs

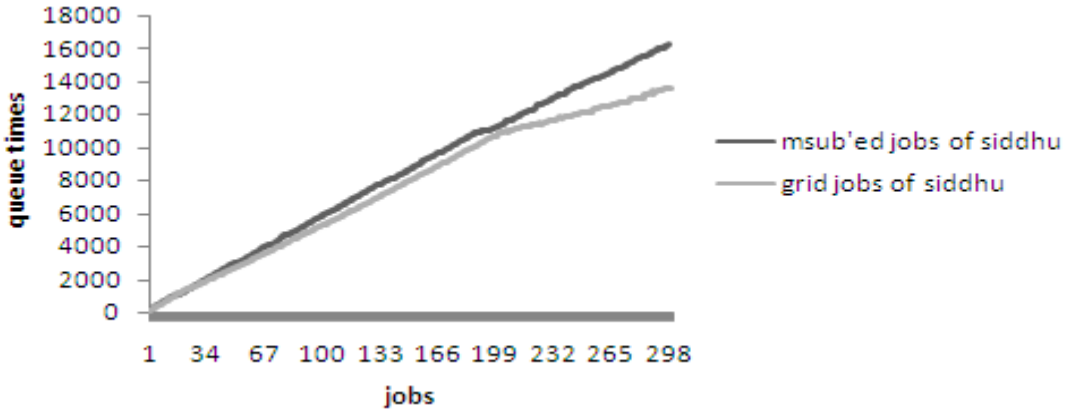

Figure 5-3. Queue times for siddhu's jobs

Finally, A graph of number of jobs versus queue time is shown in Figure 5-5 of all 1000 jobs submitted to moab-hpc by taylor, prescott and siddhu. Note that since the jobs were submitted in an interleaved manner and a QOS is setup, the queue times are random and hence the lines tracing it appear like a band. However, one can observe the peak queue time in case of a cluster scenario to be higher than the grid scenario, again demonstrating the advantage of job migration in GatorGrid.

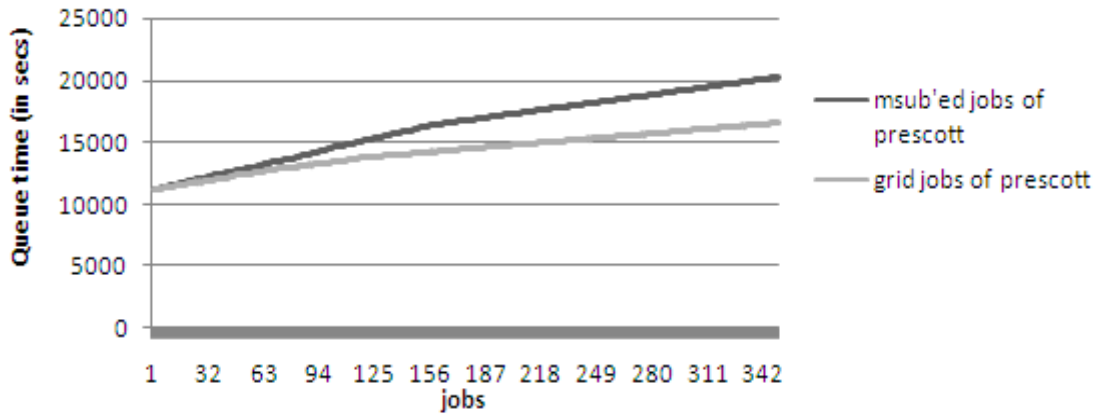

Figure 5-4. Queue times for Prescott's jobs

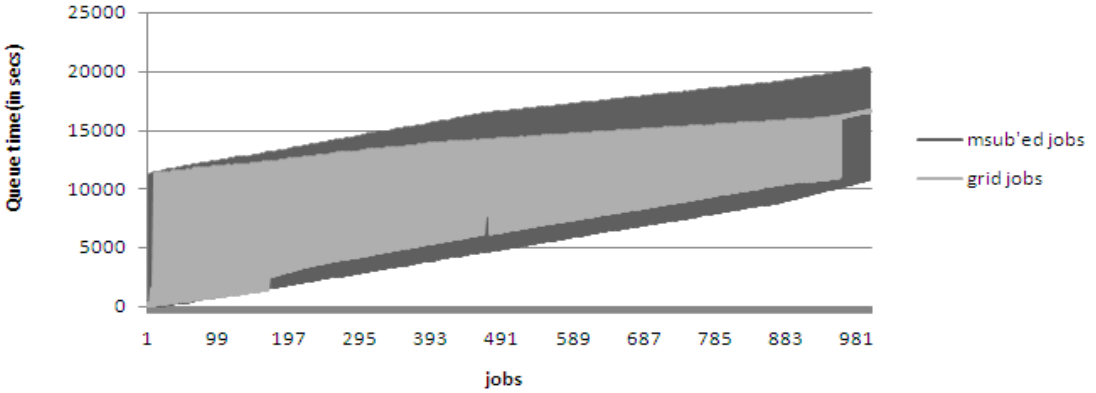

Figure 5-5. Queue times for all jobs

CHAPTER 6
CONCLUSIONS

In this thesis, a methodology and an architecture for integrating heterogeneous compute clusters at the University of Florida to form GatorGrid was described. This campus grid provides a way of scheduling jobs across the various research clusters, user authentication across administrative domains, and file staging using a mix of open-source and proprietary technologies such as OpenLDAP, Lustre file system, Torque resource manager and MOAB scheduler.
In doing so, separate administrative domains can remain heterogeneous and autonomous, but still participate in a combined campus grid. Also, it was ensured that there are minimal changes to the job scripts and the way in which jobs are submitted to the combined clusters.

A test bed for GatorGrid was built. The details its design and implementation were described, including the installation and configuration of the LDAP proxy and the configuration of MOAB job managers in a peer-to-peer fashion. The test bed was then subjected to various workloads for testing, including verification of the functionality and reliability of the test bed. Also, tests were performed to verify the reduction of queue times for users.

An important advantage of such a campus grid is that all the computing resources in the campus grid would appear as a seamless entity to a researcher. The HPC cluster has more than 800 cores and the QTP cluster has more than 400 cores. When combined, GatorGrid would provide a tremendous scale up in terms of the computing power available to a researcher and the sharing of workload across the organization. Moreover, with the increase in the number of processors available, it is possible to speed up applications that can take advantage of the parallel-processing power.

Finally, another advantage of this architecture is that it is easily scalable. The Lustre filesystem can support 10-10,000's nodes and MOAB cluster suite existing in the HPC cluster is

already supporting more than 800 cores. We do not expect the performance to deviate widely from what we have observed in the test bed. An interesting future work would be to extend and scale the GatorGrid architecture to form a more comprehensive State of Florida grid, using the Florida Lambda Rail (18). Of course, there are many challenges, both technical and non-technical. For example, if a cluster wanting to participate in this State of Florida or GatorGrid effort and does not have any of the three technologies, namely OpenLDAP, MOAB job manager or Lustre file system, then they will have to migrate to it. Although the installation and configuration of each of these technologies is quite simple, MOAB cannot be linked to all local resource managers well. For instance, there are still questions on how well MOAB interacts with the Condor resource manager. In addition, MOAB is a proprietary technology and every cluster may have to pay for its grid licenses. This might put a financial constraint on some of the clusters wanting to participate. But, considering the tremendous compute power that becomes available to the researcher, this may in the long run may not be a huge price to pay.

LIST OF REFERENCES

1. Boisseau, J. R. 2004. UT Grid: A Comprehensive Campus Cyberinfrastructure. *Proceedings of the 13th IEEE international Symposium on High Performance Distributed Computing* (June 04 - 06, 2004). High Performance Distributed Computing. IEEE Computer Society, Washington, DC, 274-275. DOI=http://dx.doi.org/10.1109/HPDC.2004.39.

2. **Bradley, Dan. 2008.** TeraGrid [Education Outreach and Training]. *http://www.teragrid.org/eot/.* [Online] TeraGrid, July 14, 2008. [Cited: July 16, 2008.] Available from: www.teragrid.org/eot/files/DanGLOWSC06.ppt.

3. **Rosen center for advanced computing. 2007.** BoilerGrid. *http://www.rcac.purdue.edu.* [Online] Purdue university, 2007. [Cited: July 18, 2008.]. Available from: http://www.rcac.purdue.edu/boilergrid/index.cfm.

4. **Yurkewicz, Katie. 2006.** Science Grid This Week, July 12, 2006 - One campus grids voyage. *http://www.isgtw.org/.* [Online] International Science Grid This Week, July 12, 2006. [Cited: July 19, 2008.]. Available from: http://www.interactions.org/sgtw/2006/0712/utgrid_more.html.

5. **Teckenbrock, Marcia. 2007.** CertificateWhatIs. *https://twiki.grid.iu.edu.* [Online] Open Science Grid, December 4, 2007. [Cited: July 22, 2008.]. Available from: https://twiki.grid.iu.edu/bin/view/Documentation/CertificateWhatIs

6. Kozakiewicz, A. and Karbowski, A. 2006. A Two-Level Approach to Building a Campus Grid. In *Proceedings of the international Symposium on Parallel Computing in Electrical Engineering (September 13 - 17, 2006). PARELEC. IEEE Computer Society, Washington, DC, 1*21-126. DOI= http://dx.doi.org/10.1109/PARELEC.2006.11

7. G. Mateescu, M. Sosonkina, "IMAGE: an approach to building standards-based enterprise grids," *Parallel and Distributed Processing Symposium, International*, pp. 404, Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, 2006. 404-412. DOI= http://dx.doi.org/10.1109/IPDPS.2006.1639661.

8. On Demand Grid Computing Technologies, Altair Engineering, 2009. Available from: http://www.pbsgridworks.com/PBSTemp1.3_2.aspx?top_nav_name=Products&item_name=OpenPBS&top_nav_str=1.

9. MOAB and Torque Documentation, Cluster Resources, 2001. Available from: http://www.clusterresources.com/products/documentation.shtml.

10. Lustre documentation, Sun Microsystems, 2008. Available from: http://wiki.lustre.org/index.php?title=Main_Page.

11. Lustre File System: High-Performance Storage Architecture and Scalable Cluster File System. Sun Microsystems. 2008. White paper. Available from: http://www.sun.com/software/products/lustre/docs/Lustre-networking.pdf

12. *Lustre File System Networking.* Sun Microsystems. 2008. White Paper. Available from: www.sun.com/software/products/**lustre**/docs/**Lustre-networking**.pdf

13. Best Practices for Architecting a Lustre-based Storage Environment. Data Direct Networks. 2008. White paper. Available from: http://www.datadirectnet.com/pdfs/WP_BestPractices_Lustre_032108.pdf

14. **Arkillis, Brian.** *LDAP Directories Explained An introduction and Analysis.* s.l. : Addison-Wesley, 2003. 0-201-78792-x.

15. **Vidmar, Anze.** How to setup and maintain OpenLDAP server for your network. Available from: http://www.fedoranews.org/mediawiki/index.php/How_to_setup_and_maintain_OpenLDAP_server_for_your_network.

16. Desire: Development of a European Service for information on research and education, Institute for Learning and Research Technology, April 20, 2000. Available from: http://www.desire.org/html/services/directoryservices/.Moab Workload Manager Administrator's guide. Cluster Resources Inc. Available from: http://www.clusterresources.com/products/mwm/docs/.

17. MOAB Workload Manager Administrative guide, Cluster Resources Inc. Available from: http://www.clusterresources.com/products/mwm/docs/

18. Florida LambdaRail-FLR infrastructure. Florida LambdaRail, LLC, 2005. Available from: http://www.flrnet.org/infrastructure.cfm.

BIOGRAPHICAL SKETCH

Siddhartha Eluppai Srivatsan received his M.S from the University of Florida in the summer of 2009 and received his B.Tech from Vellore Institute of Technology in the spring of 2006. His research interests include Grid Computing and digital design using field programmable gate arrays. He worked under Dr. Paul Avery of the Physics department as a research assistant contributing to the Open Science Grid's in its Education Outreach and Training efforts and to the Campus Grid efforts at the High Performance Computing Center, Department of Physics at the University of Florida.