

A COLLABORATIVE HIGH PERFORMANCE
AND
GRID COMPUTING PORTAL

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

In

The Department of Electrical and Computer Engineering

by
Prathyusha Akunuri-Venkata
B.S., JNTU, Hyderabad, India, 2002
December 2007

I dedicate this work to my beloved sister Prashanthi who left this world eight years back. She has always been my inspiration and motivation from my childhood.

Acknowledgments

I express my sincere gratitude to everyone in any way connected to the work presented in this report. I am thankful to the Louisiana Optical Network Initiative (LONI), The Department of Electrical and Computer Engineering and the Center for Computation & Technology for providing me with the opportunity to work in the field of my interest. I would like to specially thank Dr. Daniel S. Katz and Dr. Gabrielle Allen for their constant support, guidance, encouragement and invaluable time that they put in to help me make progress towards completion of this thesis. I am extremely grateful to my beloved husband Archit Kulshrestha for helping me at all stages during the project without which it would have been a daunting task to complete. I sincerely thank Dr. J. Ramanujam for agreeing to be a co-chair of the committee. I am very thankful to Mr. Charlie McMahon, Mrs. Allie Hopkins and Mr. Brian Ropers-Huilman for giving me an opportunity to work with LONI. I would also like to express my sincere thanks to all my LONI colleagues for coordinating with me in building the LONI Portal. I would like to extend my sincere thanks to Ian Kelley for his advise and support. Last but not least, I am indebted to my parents and my grandfather for helping me become the person that I am today.

Table of Contents

Acknowledgements	iii
List of Tables	vi
List of Figures	vii
Abstract	ix
1 Introduction	1
1.1 Motivation	1
1.2 Overview of Grid Computing	2
1.3 Overview of High Performance Computing	4
1.4 User Interfaces for High Performance Computing and Grids	5
1.5 Web Portals	6
2 High Performance and Grid Computing Portal Requirements	8
2.1 Requirements for Building a HPC Portal	8
2.2 User Classes and Their Requirements	10
2.2.1 User Classes in HPGC Portal	11
3 Infrastructure and Technologies	13
3.1 Infrastructure	13
3.1.1 CCT Grid	13
3.1.2 LONI Grid	13
3.2 Grid Software	14
3.3 High Performance Computing Software	15
3.4 Network Software	16
3.5 Portal Software	16
3.5.1 GridSphere Portal Framework	16
3.5.1.1 Out-of-the-box GridSphere Portlets	18
3.5.1.2 GridPortlets	19
4 Architecture and Implementation	21
4.1 Architecture of HPGC Portal	21
4.2 Authentication Mechanism	22
4.2.1 Details of LDAP Implementation	23
4.3 Implementation of Portlets Using a VO Centric View	34
4.3.1 GridPortlets	34
4.3.2 Other Portlets	41
4.4 Challenges in Creating a Multi-VO Portal	42
4.4.1 Users and VOs	42

4.4.2	Grid Credential Management	42
4.4.3	Resource Management	43
5	Design and Components	44
5.1	Layout	44
5.1.1	Themes	45
5.2	Portlets	45
5.2.1	Portlets Summary	45
5.2.2	Profile Settings Portlet	45
5.2.3	GPIR Browser Portlet	46
5.2.4	GridPortlets	47
5.2.5	Allocations Portlet	50
5.2.6	MRTG Portlet	51
5.2.7	RancidConfig Portlet	52
5.2.8	AboutHPGC Portlet	52
5.2.9	Weekly News Update Portlet	52
6	Conclusions and Future Work	54
6.1	Advantages of Integrating CCT and LONI HPC Portals	54
6.2	Related Work	55
6.2.1	TeraGrid Portal	55
6.2.2	SuraGrid Portal	56
6.2.3	Comparison	57
6.3	Other Related Solutions	58
6.4	Conclusion	59
6.5	Future Work	59
	Bibliography	61
	Vita	64

List of Tables

3.1	CCT Grid	13
3.2	LONI Grid	14
5.1	Summary of work	45
6.1	Comparison of HPC Portals	58

List of Figures

3.1	Globus ToolKit 4.0 Architecture	15
3.2	GridSphere Framework Architecture	17
4.1	HPGC Portal Components and Architecture	22
4.2	Authentication Message Sequence	31
5.1	Guest User Layout of HPGC Portal	44
5.2	Profile Settings Portlet	46
5.3	(i) GPIR Browser Portlet - CCT (ii) GPIR Browser Portlet - LONI	47
5.4	GPIR Browser Portlet - HPC	47
5.5	Resource Registry Portlet	48
5.6	Grid Credentials Portlet	48
5.7	(i) Grid File Browser Portlet - CCT (ii) Grid File Browser Portlet - LONI	49
5.8	Grid File Browser Portlet - HPC	49
5.9	(i) Job Management Portlet - CCT (ii) Job Management Portlet - LONI	49
5.10	Job Management Portlet - HPC	50
5.11	(i) Allocation Portlet - CCT (ii) Allocation Portlet - LONI	50
5.12	Allocation Portlet - HPC	50
5.13	LONI MRTG Portlet	51
5.14	LONI RancidConfig Portlet	52
5.15	AboutHPGC Portlet	53

5.16	Weekly News Portlet	53
6.1	TeraGrid Portal Login Page [23]	55
6.2	TeraGrid HPC Queue Prediction Portlet [23]	56
6.3	SURAGrid Portal Login Page [25]	56
6.4	SURAGrid Portal after user Logs in [25]	57

Abstract

The world of high performance and grid computing is so vast that it always poses constant challenges and high learning curves to researchers and scientists. A Portal-based solution facilitates a simple web-based graphical user interface that helps the non computer science users take advantage of highly evolving, on-demand, distributed computing to carry out their research activities. A collaborative portal brings together various resources, applications and services under one roof and provides a single point of entry and access to the users.

This thesis reports on work that endeavors to build a unique High Performance and Grid Computing Portal that combines two existing high performance portals into a single portal with which a user can easily access and use to carry out various high performance and grid computing tasks that previously required two portals. This report addresses the requirements for building a high performance computing portal, the various solutions that can be offered, variations in the solutions due to uniqueness of each individual grid infrastructure, problems that arise in combining two existing high performance computing portals, and solutions to these problems. As a test case, we have combined the CCT HPC Portal and the LONI Portal, both of which were developed by the author of this thesis. The portlets that have been developed during this work include, but are not limited to, network monitoring, grid accounting, reporting, and generic portlets that could be used by any high performance computing portal upon making a few configuration changes. This work also introduces integration of LDAP authentication with the GridSphere framework which enables a Virtual Organization centric view of the portal. This solution opens up two large grid infrastruc-

tures to the LONI and LSU grid communities through one web interface without worrying about different logins and grid certificates. The portal was developed based on the user's perspective. Access to the portal is controlled by the affiliation and role of the user in each or both of these organizations.

1 Introduction

1.1 Motivation

Grid Portals are becoming increasingly popular in high performance computing environments today. Their major advantage is an easy-to-use graphical user interface with single sign on, collaborative portlets and many useful features available collectively in a single place.

Here at LSU, there are two major grid computing environments: CCT and LONI. Each of these infrastructures has its own grid portal. But much of the user community is common between the two environments and it is tiresome for these users to access two different portals for carrying out similar tasks on the two different set of machines. This work is based on the realization that there is a need for a collaborative high performance and grid computing portal that would allow a user to access both the CCT and LONI grids depending on his/her affiliation. Not all CCT users use the LONI grid and vice-versa; hence, to address this concern, a portal was developed that is designed completely for a user perspective and depends on the affiliation of the user with CCT and/or LONI. This is a unique and flexible solution that will help grid portals be more robust and usable by all kinds of users. This solution eliminates the need to develop a separate grid portal for every grid infrastructure. This helps encourage collaboration between grid communities and helps work towards a common goal that would benefit the entire society. This thesis addresses the design issues and advantages and shortcomings of a collaborative high performance and grid computing portal.

This thesis details the necessity of a collaborative high performance and grid computing portal (HPGC). This chapter gives a brief introduction of grid computing, high performance computing, GridSphere and various interfaces for HPC and Grids. The next chapter goes on to explain the

requirements for building an integrated High Performance and Grid Computing portal to meet the needs of users, administrators and developers and also explains the user classes. Chapter 3 explains the hardware and software stack of CCT and LONI Grids. It also introduces the grid and network monitoring software. It introduces the GridSphere Portal framework and explains all the out-of-the-box portlets including GridPortlets that are currently deployed in the HPGC Portal. Chapter 4 explains the architecture and implementation of the HPGC portal. This chapter provides a detailed explanation of the authentication mechanism at various levels. Chapter 5 explains the implementation and design components of the HPGC Portal including the layout, portlets, and configuration using the concept of Virtual Organizations. The thesis concludes by describing the advantages of combining two large HPC Portals like those of the CCT and LONI and the sample solutions that have been implemented. This chapter also shows a comparison and surveys details of some typical existing High Performance Computing portals such as those of the TeraGrid and SURAGrid with the HPGC portal. It ends with a discussion on future work that can improve the integration of portals.

1.2 Overview of Grid Computing

Grid computing enables researchers to carry out data-intensive and/or computation-intensive tasks on rich and dynamic environments that can adapt to various changing conditions. The phrase “Grid Computing” is derived from a popular form of computing called distributed computing that refers to a method of processing different parts of a program simultaneously on multiple systems that are connected over a network. Distributed computing is also a form of parallel computing that takes into account the infrastructure differences between the the multiple systems across which different parts of a program are running, such as difference in file systems, etc., when a program is

parallelized.[1] Grid computing's focus on the ability to support computation across administrative domains sets it apart from traditional distributed computing.

The first preliminary idea of grid computing was suggested by Len Kleinrock in 1969. He said "We will probably see the spread of computer utilities, which, like present electric and telephone utilities, will service individual homes and offices across the country"[2]. This was later defined by Ian Foster and Carl Kesselman, known as fathers of grid computing, as "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." [1] This was redefined by Ian Foster and Steve Tuecke to address the social and political issues of a grid. They believed that "Grid Computing is concerned with "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations." The key concept is the ability to negotiate resource-sharing arrangements among a set of participating parties (providers and consumers) and then to use the resulting resource pool for some purpose." [2]

Computational grids, commonly collectively called "The Grid," represent hardware and software infrastructures that enable dynamic resource sharing between individuals or institutions grouped as virtual organizations. These resources are often many computers, connected to a network, that are applied to solve a single scientific problem. Grids provide a rich and dynamic environment with the potential to enable a researcher to couple models in complex and self-organizing workflows and react to changing environments and resource conditions. The Grid also provides capabilities for accessing or exchanging distributed data and distributed computational resources.

Grids are often functionally classified as *Computational Grids* and/or *Data Grids*. Computational grids are used to perform processor intensive tasks such as simulations, visualization, etc. System software on such grids consist of software that enables them to receive job processing re-

quests such as queueing systems and resource managers. Data grids are primarily used to store large amount of data and are characterized by large storage spaces and special data handling software and services such as the Replica Location Service[30]. There are several other forms of grids such as equipment grids, enterprise grids, etc., depending on their capability and focus.

1.3 Overview of High Performance Computing

High-performance computing (HPC) is a term that arose after the term “supercomputing.” HPC is defined as a branch of computing that concentrates on developing supercomputers and software to run on supercomputers. Developing parallel processing algorithms and software–programs that can be divided into pieces so that each piece can be executed simultaneously by separate processors– is the main area of concentration of this discipline. A popular web definition of HPC is “A branch of computer science that concentrates on developing supercomputers and software to run on supercomputers. A main area of this discipline is developing parallel processing algorithms and software: programs that can be divided into little pieces so that each piece can be executed simultaneously by separate processors.” The wikipedia definition of HPC is “use of (parallel) supercomputers and computer clusters, that is, computing systems comprised of multiple (usually mass-produced) processors linked together in a single system with commercially available interconnects. This is in contrast to mainframe computers, which are generally monolithic in nature. While a high level of technical skill is undeniably needed to assemble and use such systems, they can be created from off-the-shelf components. Because of their flexibility, power, and relatively low cost, HPC systems increasingly dominate the world of supercomputing. Usually, computer systems in or above the teraflop-region are counted as HPC-computers.”[5] This definition is used to show popular opinion about HPC among non-computer savvy users.

1.4 User Interfaces for High Performance Computing and Grids

Interacting with Grids or HPC components can be a complex and tiring process, particularly for novices. This usually involves typing various commands or executing complex scripts. To encourage a large user community that includes non-computer-savvy researchers and scientists to take advantage of grids and HPC, there is a need for a simple and interactive graphical user interface (GUI). This GUI should mask all the intricacies of a complex grid and underlying grid services and provide a easy to use interface that can be accessed via the internet. The interfaces that are currently available to grid and HPC users are

- Command Line interface - This involves typing possibly long and complex commands to carry out required grid tasks. As part of this process, users write complex scripts such as job submission scripts and execute them on the command line. Though this is acceptable for some advanced users, it seems to be a tiring and tricky process to most of the non computer savvy users. It involves reading a lot of documentation and following complex rules to be able to achieve the desired behavior of the grids for a particular job.
- Web Applications - This is a more evolved form of user interface than the command line. Programming languages such as Java, C, C++ are used to build web applications that can carry out some of the complex tasks on the grid. A potential shortcoming of a web application is that it sometimes involves installation of an application on the client computer. Also, it is always challenging to bring together various web applications under one single roof. Additionally, providing a centralized security solution can also be tricky. Finally, the components of a web application are often not re-usable.
- TCL Applications - This is an older style of applications that sometimes have a GUI that is less

easy to use than those of web applications. TCL interpreters have to be present on the client computer. These interpreters are not as widely available as internet browsers.

- Portals - This is a very popular form of GUI for grids and HPC these days. It provides a platform for various related applications to reside together. It provides a single point of entry and single sign-on to the user to carry out various different tasks on grid without having to worry about the underlying complexities. Most portlets have re-usable components that makes it easy for grid portal developers to develop related components. A Portal is accessed via a web browser from anywhere and does not need any components to be installed on the client system.

1.5 Web Portals

There are many definitions of a web portal. According to IBM, who develop the industry's most popular portal framework, WebSphere Portal Server, "a portal is a Web site that provides users with a single point of access to Web-based resources by aggregating those resources in one place and by requiring that users log in only to the portal itself, and not to each portlet they use." [7]

There are many portal frameworks, for example WebSphere Portal [6], WebLogic Portal [8], SAP Netweaver [9], PlumTree Portal [10], Gridphere Portal Framework [21], Liferay [11], Jakarta Pluto [12], Jakarta JetSpeed [13], etc. Most of these Portal frameworks are Java-based and are JSR-168 compliant. Some of them are open source and are freely available for download. But the specific interest for this project was a Portal framework that is not only JSR-168 compliant but also provides a framework for developing grid based portlets. The only framework with an API for development of grid based portlets is the GridSphere Portal Framework. [21]

GridSphere is a open source grid-enabled web-based portal framework. It was developed as a part of the GridLab project, funded by the European Commission. Though the project has ended,

development of GridSphere is still continuing. This framework provides a base for developing grid-based portlets and comes with out-of-the-box GridPortlets that are designed to carry out basic grid tasks. It is JSR-168 compliant, which makes it a flexible framework for developing non-grid-based portlets also.

2 High Performance and Grid Computing Portal Requirements

2.1 Requirements for Building a HPC Portal

The following are the advantages of having a user portal in a High Performance and Grid Computing environment

- Users will have one place to access all the applications that are useful in carrying out grid tasks, and a single place to access different resources.
- The interface is easy to use and can be accessed from anywhere via a web browser
- A single sign-on enables the user to carry out his/her tasks without having to log onto each machine on the grid separately.
- It provides one place where users can get all the information about the Grid such as allocations, resource availability, job submission, file systems and anything that is relevant for carrying out their grid related tasks
- Users can receive automated notifications via SMS, email or instant messages based on events such as job completion, change in resource status, etc.

A set of requirements for building a user-friendly High Performance and Grid Computing portal are:

- It shall have a stable and reliable identity management and security system for authentication and authorization.
- It shall give users information such as load, memory, processors, storage, etc, about all the resources on the Grid.
- It shall allow user to carry out file transfers and job submission on the grid.

- It shall provide information about the job managers, resources managers, schedulers and queues available on the compute resources.
- It shall provide allocation information.
- It shall provide documentation and context sensitive help.

Identity management and security are closely coupled components that are critical to having a solution that is both secure as well as user-friendly. A good solution is to have a common user management system such as LDAP[18] that is used for user management on machines as well as for the Portal. This allows users to have a common user name across the grid and also easier-to-remember userid and password instead of multiple credentials. Most portals require user to request credentials that are needed to access the portal, and upon approval, they are notified via email. Most of the time, the user credentials are not same as the credentials used to access the grid. Examples of such a grid portals are TeraGrid and SURAGrid.

A portlet that gives useful information about the resources of a Grid without actually logging onto each of the machines is a useful feature of any HPC portal. The GPIR portlet developed by GridPort is one such portlet that can be customized to display useful information such as load, CPU speed, disk size, memory and jobs. It gives detailed information about all the jobs that are running, queued, idle or are in other states. It also displays useful information about each job such as the owner, the start time and the number of processors requested. This is one way a user can judge the availability of a machine for a particular task. The load can be used to estimate the availability of processors on a machine.

Portlets that allow user to carry out grid tasks such as file transfers using GridFTP and job submission via a easy-to-user Graphical User Interface (GUI) are another useful component of

a HPC Portal. Grid Portlets are open-source out-of-the-box portlets that can be obtained from the GridSphere site and customized to specific needs of a grid. Also, various versions of job submission portlets for specific needs can be developed by using the GridPortlets API. GridPortlets also have support for Globus Web Services. They can be configured to use WS-MDS to obtain scheduler, queue, and job manager information for each of the grid resources running WS-MDS.

Another useful feature HPC Portal users need is allocation information. Each user can be associated with various projects and can have multiple allocations. Remembering all that information can be tiresome and also confusing. A portlet showing users all the allocations that they hold with various VOs and the balances available would help manage the allocations better and take complete advantage of the grid resources.

Useful documentation about the grid and the usage of the portal itself is the most important element of any portal. This helps users become familiar with the portal and also helps them carry out their tasks effectively without having to waste time searching for documentation elsewhere. This makes the portal a one-stop-shop for all the information about the grid and any other useful components of the grid. Having a help tab on each portlet explaining the configuration options available and other useful information about the portlet is one easy way of giving users access to information. Linking various wiki pages or other document repositories to the portal is another way. Customized portlets can be developed to access these documents by using GridSphere and JSR-168 API.

2.2 User Classes and Their Requirements

In general, every portal has different set of users with different roles and access levels. The three major default user classes in GridSphere are

- Super User - This role is similar to root on linux systems. This role should ideally only be used to configure the portal at the get-go. It is recommended not to use this role for regular administration tasks.
- Admin - The administrator has unlimited access to the Portal. This includes adding and deleting users, deploying applications, making global configuration changes to the portal that affects all the users.
- User - Users in this role can view applications that they are authorized to access. They can edit the applications if authorized. They can also make changes that affect just their portal layout and view and do not cause any global change.

2.2.1 User Classes in HPGC Portal

The HPGC Portal is a Virtual Organization (VO) based Portal. Users from two different VOs can access the portal. The different classes of users, required because of separate user management (LDAP) infrastructures and Certificate Authorities across both the grids (LONI and CCT), are:

- CCT Class - These are the LSU HPC users who have CCT LDAP user credentials and use CCT Certificate Authority signed Certificates to access the HPC grid.
- LONI Class - These users who belong to any of the six different LONI partner schools across the state. LSU is also one of the LONI partners. These users do not use CCT grid resources for their tasks.
- Combined Class - These are the users at LSU that have access to both LONI and CCT grids and have certificates and credentials from both the grids. There are again three different user classes in this category.

- Users with same user id and password across both CCT and LONI grids.
- Users with same userid but different passwords
- Users with different userids.

Users from each of these classes get different access to the portal. All CCT users can access the portlets that are specific to the CCT Grid. Users with access to LONI have access to portlets that are specific to LONI resources. Users with same userid and password across both the grids have access to both sets of portlets (LONI and CCT). Most of the portlets have both views combined. For users that have access to both VOs but have different passwords, the portal automatically sets their class to that of the VO whose credentials they choose to use to logon for the first time. Users can later change their organization manually to obtain other views. There are some issues that need to be addressed when dealing with users from different VOs. Different users can possibly have the same userid in different VOs and the same user can have different userids on different VOs. In the first case the portal will use the password to authenticate the user and a successful LDAP bind will be used to identify the correct VO. In the second case however the portal must be somehow made aware of the userids of the user on the other systems. This can be done by maintaining a mapping on the portal and this will need human intervention to unambiguously resolve userids from usernames. The other alternative is to provide the user a portlet that would allow him to register his credentials for the other VO and then the portlet would maintain a list of userids. Since we do not have different users with the same userid on the LONI and CCT VOs, this issue was not addressed and will be regarded as future work depending on the specific VO that we need to integrate into the portal.

3 Infrastructure and Technologies

3.1 Infrastructure

3.1.1 CCT Grid

The Center for Computation & Technology is an innovative and interdisciplinary research environment for advancing computational sciences, technologies, and the disciplines they touch. Our efforts branch out from the center to serve Louisiana through international collaboration, promoting the progress in leading edge and revolutionary technologies in academia and industry.

The CCT Grid as shown in the Table 3.1 consists of four systems. All the CCT machines are located at Louisiana State University. Tezpur is not currently ready for use but is expected to be available by October 2007.

TABLE 3.1: CCT Grid

Name	Type	OS	Nodes	CPUs	Peak Performance
Tezpur	Dell Intel Xeon	RHEL V4	360	1440	15.322 TFlops
Pelican	IBM Power P5+	AIX V5.3	30	368	2.797 TFlops
Santaka	SGI Intel Itanium2	SUSE 9	1	32	0.192 TFlops
Nemeaux	Apple G5	Mac OS X	32	64	0.256 TFlops

3.1.2 LONI Grid

The Louisiana Optical Network Initiative, or LONI, is a fiber optics network that connects supercomputers at Louisiana's major research universities, allowing computation speeds than 1000 times the rate previously possible, and transforming the research capability of Louisiana's educational institutions. With an anticipated 85 teraflops of computational capacity, LONI will be one of the nation's largest grid computing environments. Governor Kathleen Babineaux Blanco has pledged \$40 million over ten years for the development and support of LONI.

The LONI grid as shown in the Table 3.2 will include five IBM Power5 systems located at five partner institutions. Each of the P5s have a peak performance of 0.851 Tera Flops and have 16GB of RAM per node. The LONI grid will also include six 5 TFlop Intel Linux clusters, one of which is housed at each partner institutions (Eric is an example of one of these systems; the other five systems have not yet been named). Finally, the LONI grid will also include a 50 TFlop Linux cluster called Queen Bee which is located at state’s Information Systems Building in Baton Rouge. This is among the top 25 super computers in the world.

TABLE 3.2: LONI Grid

Name	Location	Type	OS	Nodes	CPUs
Queen Bee	Downtown, Baton Rouge	Dell Intel Xeon	RHEL V4	680	5440
Bluedawg	Louisiana Tech	IBM P5	AIX 5.3	14	112
Ducky	Tulane Univ.	IBM P5	AIX 5.3	14	112
Zeke	Univ. of Louisiana,Lafayette	IBM P5	AIX 5.3	14	112
Neptune	Univ. of New Orleans	IBM P5	AIX 5.3	14	112
Lacumba	Southern Univ.	IBM P5	AIX 5.3	14	112
Eric	Louisiana State Univ.	Dell Intel Xeon	RHEL V4	128	512

3.2 Grid Software

Grid Middleware: Globus is an open source Grid middleware package that bundles together services and libraries that allow users and other application software to perform basic tasks such as resource monitoring, discovery, security and file management. “Its core services, interfaces and protocols allow users to access remote resources as if they were located within their own machine room while simultaneously preserving local control over who can use resources and when.”[14]

The Globus ToolKit has two implementations, one using C-based Unix services and the other using the web services reference framework (WSRF) in C and Java. Both of these are installed on resources that are part of the Grids at CCT and LONI. These services are used by GridPortlets that are part of HPGC Portal and that enable users to carry out grid tasks.

are the software components of CySolidus[16]. A portal interface connecting to the Gold database and retrieving all the allocation information of a user based on his/her affiliation and displaying them when the user logs into the portal has been developed and deployed on the HPGC Portal.

3.4 Network Software

MRTG: The Multi Router Traffic Grapher is open source network monitoring software that monitors many types of network devices. It provides graphical analysis of network traffic across the routers. This software is used by LONI to graph the traffic across all the interfaces connecting the LONI network across the state of Louisiana. A portal interface for MRTG has been developed and deployed on the HPGC Portal.[19]

RANCID: The Really Awesome New Cisco confIg Differ is open source software that monitors a “router’s configuration, including software and hardware (cards, serial numbers, etc) and uses CVS (Concurrent Version System) or Subversion to maintain history of changes.”[20] This software is used by LONI network administrators to access router configuration. A portal interface for Rancid that displays all the router configurations that can be only accessed by network administrators has been developed and deployed on the HPGC Portal.

3.5 Portal Software

3.5.1 GridSphere Portal Framework

GridSphere provides an easy and flexible portal framework for third-party portlet application developers. GridSphere is JSR-168 compliant. It has some out-of-the-box portlets such as Login Portlet, Layout Manager Portlet, Profile Settings Portlet and more administration portlets such as Personalized Layout Manger, Groups, Roles, etc. GridSphere also comes with out-of-the-box grid portlets such as Credential Registry Portlet, Resource Registry, Files, Resources and Job Manager

Portlets that can be configured and customized to individual grid needs. GridSphere is a very robust and easy-to-use framework that has the following important features that helped in building the HPGC Portal.[21]

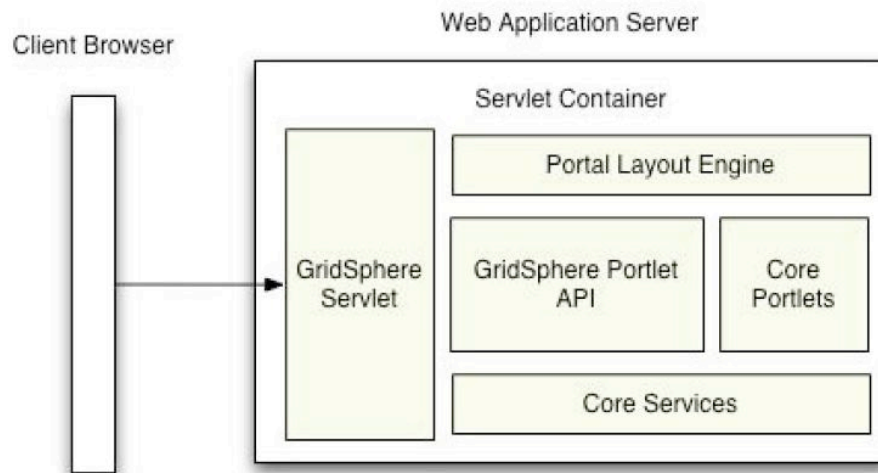


FIGURE 3.2: GridSphere Framework Architecture

- Provides a legacy portlet API called GridSphere Portlet API that enables developers to build portal applications for grids easily. It is also completely JSR-168 compliant.
- Flexible, XML-based portal presentation description can be easily modified to create customized portal layouts
- Built-in support for Role Based Access Control (RBAC) separating users into guests, users, admins, and super users
- Sophisticated portlet service model that allows for creation of “user services,” where service methods can be limited according to user rights
- Support for multiple international languages

3.5.1.1 Out-of-the-box GridSphere Portlets

GridSphere comes with built-in basic portlets for carrying out administrative tasks such as management of users, portlets, layout and messaging. It also comes with informative portlets that can be customized to display static data. The HPGC Portal utilizes the functionality of these out-of-the-box GridSphere portlets:

- **Login Portlet:** This portlet is used to restrict access to the Portal. This has been customized to use LDAP as the security solution.
- **Welcome Portlet**
 - **Settings Portlet:** It has two components 1. Profile Settings where user can modify his/her contact information. 2. Group Membership component allows user to manage his/her membership to all authorized portlets.
 - **Layout Portlet:** Allows user to choose a theme. It also allows user to create a new layout and place the portlets according to their choice.
- **Administration Portlets:** These are a set of six portlets that are available only to administrators. Regular users are not able to view these portlets.
 - **Portlets:** This portlet is used to deploy portal applications and configure various options such as login, authentication modules, error handling, etc. This portlet also shows all the users currently logged onto the portal.
 - **Users:** This portlet is used to create a new user if the portal uses GridSphere-based local authentication system. It displays all the users that have ever logged into the portal and their contact details.

- **Groups:** This portlet is used to create a group and associate a single or set of portlets with it. It allows the Administrator to specify the role that is required to access the group. This portlet is also used for managing the existing groups.
- **Roles:** This Portlet is used to create new roles and associate users with those roles.
- **Layouts:** This Portlet is used to edit the Portal header and footer information, choose a default global theme, edit the guest user layout and edit the layout associated with each of the groups.
- **Messaging:** This Portlet is used to configure messaging services like AOL AIM, Cingular SMS and Email. This components can be used for notifications by any of the portlets.
- **Informative Portlet:** This portlet gives a template to display static information.

3.5.1.2 GridPortlets

GridSphere comes not only with its core functionality, but it also has an integrated Grid portal toolkit called GridPortlets. Developers can use GridPortlets to create custom Grid-based portal applications that abstract the details of underlying Grid technologies and offers a consistent and uniform high-level service API. The GridPortlets services provide functionality for resources, proxy credentials, remote files, jobs, and they support persistent information about credentials, resources, and jobs submitted by users. The GridPortlet service API currently supports Globus Toolkit[3]. GridPortlets comes with five well-designed, easy-to-use Globus-based portlets: resource registry, resource browser, credential management, job submission, and file management. These portlets can also be modified and extended for customization based on the portal requirements. The HPGC Portal development project is based on developing a collaborative Portal which is Virtual Organization based. Each of these portlets have been modified to acquire the required

behavior.

- **Resource Registry Portlet:** This is a portlet that is available only to Administrators and is used to configure the grid resources that the users can access and use to carry out their grid tasks. Grid resources, proxy server and portal credentials are configured in this portlet. GRAM resources and other hardware and software details can be configured in this portlet.
- **Credential Retrieval Portlet:** This portlet lets the user retrieve credentials from the MyProxy[4] server that is configured in Resource Registry Portlet to accept the user's grid credentials and gain single sign-on access to all the Grid resources.
- **Resource Browser Portlet:** This portlet has three unique sub portlets that show system information, services running on each of the resources and queue and scheduler information on all grid resources to which the user has access.
- **File Browser Portlet:** This portlets allows users to browse and manage physical files on Grid resources. It uses GridFTP.
- **Job Submission Portlet:** This portlet allows users to submit and manage jobs on their resources using Globus-PreWS and Globus-WS.

4 Architecture and Implementation

4.1 Architecture of HPGC Portal

The HPGC Portal makes use of preexisting and new components to meet the requirements from Chapter 2. GridSphere and GridPortlet frameworks provide a set of basic services that are used to perform various tasks. These services were modified to be able to support multiple VOs. Figure 4.1 shows the various components within the portal sever and their interactions specifically for handling VO specific requests.

The user interacts with the Portal UI components through a browser. In order to login to the portal the user must enter his username and password in the login portlet. The portlet then transfers control to the LoginService that uses the User Management service to obtain the User object for the user with the given login name. If successful, the LoginService invokes the authmodules depending on the order of priority and tries to authenticate the user. GridSphere provides various authentication mechanisms such as password based authentication using he GridSphere database, certificate based authentication using the browser certificate store etc. It is possible to assign a priority value to these mechanisms which is an integer between 0 and 100. The highest priority auth mechanism for the HPGC Portal, developed as part of this thesis work, is LDAP authentication and is assigned a priority value of 100 and is hence used as the first authentication mechanism. The LDAP auth module tries to bind with one of the LDAP servers and if successful, it returns success and the user is logged in to the portal.

The GridSphere database is used to store various persistent pieces of information to permanent storage, such as user information. In order to be able to handle VOs, the users need to have an organization that is associated with a VO, and the resources that will be used need to be associ-

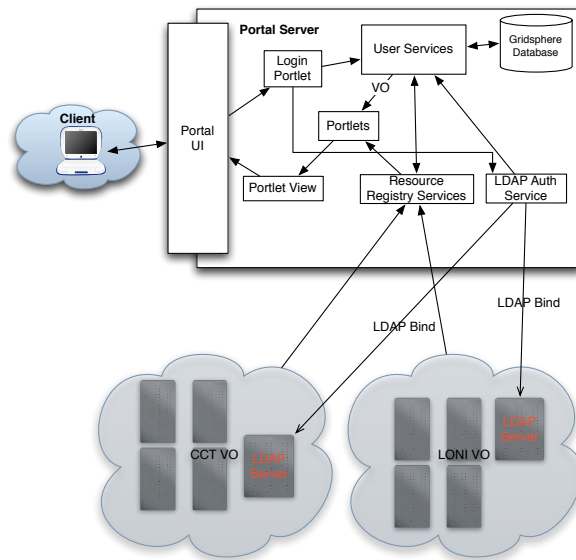


FIGURE 4.1: HPGC Portal Components and Architecture

ated with a VO. GridPortlets maintain a Resource Registry that holds information about compute resources, services, etc. All of these must be associated with a VO so that the right set of resources is presented to the users based on their VO membership. Implementing VOs in GridSphere is explained in more detail in Section 4.3.1.

4.2 Authentication Mechanism

Every portal solution must be backed up by a strong authentication framework. As mentioned in Section 4.1, the HPGC portal uses LDAP[18] as its primary authentication mechanism. LDAP is an industry standard tool for storing directory information. It is often used for storing user information and access to this information is password based. LDAP is a protocol that allows querying and modifying the directory information using clients that can be installed on compute resources to authenticate login requests. CCT and LONI have two different LDAP implementations, which are used to store user information and provide authentication services and hence LDAP was the obvious choice for portal authentication. LDAP is a widely used protocol and High Performance

Computing centers generally use LDAP to store user information and hence this solution is applicable to a large set of VOs.

4.2.1 Details of LDAP Implementation

The core GridSphere and GridPortlets codes were modified to be able to use the CCT and LONI LDAP directories to authenticate users. Both LDAP directories require the use of Secure Sockets Layer (SSL). The SSLConnection class uses the Novell LDAP API to provide a generic method called SSLBind that receives a LDAP host, a login DN and a password string to authenticate users using a LDAP bind call. The class uses a Java SSL keystore to store the SSL certificates of the LONI and CCT Certificate Authorities. These CAs are used to sign the LDAP service certificates so that all communication with the LDAP servers can be encrypted. The code for the sslBind method is:

```
public boolean sslBind(String ldapHost ,
String loginDN , String password) {
    int ldapPort = LDAPConnection.DEFAULT_SSL_PORT;
    int ldapVersion = LDAPConnection.LDAP_V3;
    String path = "/etc/portal.keystore";
    LDAPSocketFactory ssf;
    try {
        // Dynamically set JSSE as a security provider
        Security.addProvider(new com.sun.net.ssl.
        internal.ssl.Provider());
        // Dynamically set the property that JSSE uses
```

```

// to identify
// the keystore that holds trusted root certificates
System.setProperty("javax.net.ssl.trustStore", path);
ssf = new LDAPJSSESecureSocketFactory();
// Set the socket factory as the default for
all future connections
LDAPConnection.setSocketFactory(ssf);
// Note: the socket factory can also be
passed in as a parameter
// to the constructor to set it for this
connection only.
LDAPConnection lc = new LDAPConnection();
// connect to the server
lc.connect( ldapHost , ldapPort );
// authenticate to the server
lc.bind( ldapVersion , loginDN , password.
getBytes("UTF8") );
le = lc.read(loginDN);
// at this point you are connected with a
secure connection
System.out.println(
    "Successful SSL bind with server.");

```

```

        lc.disconnect();

        return true;
    }

    catch( LDAPException e ) {

        System.out.println( "Error: " + e.toString() );

        return false;
    }

    catch( UnsupportedEncodingException e ) {

        System.out.println( "Error: " + e.toString() );

        return false;
    }
}

```

The `getLdapUserInfo` method uses the `LDAPEntry` object from a successful LDAP bind to receive the user's information such as the name of the organization, full name, email address etc and stores this into a `Gridsphere User` object. The code that implements this method is:

```

public boolean getLdapUserInfo(User user) {

    if(le == null)

        return false;

    LDAPAttribute at1 = le.getAttribute("givenName");

    String gName = at1.getStringValue();

    user.setGivenName(gName);
}

```

```

System.out.println("Name being set to " + gName);

user.setFamilyName(
    le.getAttribute("sn").getStringValue());

user.setEmailAddress(
    le.getAttribute("mail").getStringValue());

user.setFullName(
    le.getAttribute("cn").getStringValue());

return true;
}

```

The LDAPAuthModule class that is part of GridPortlets source was modified to accommodate the specifics of the CCT and LONI LDAP directories, as follows:

```

public void checkAuthentication(User user , String password)
throws AuthenticationException {

    int orgMask = 0;

    boolean authflag = false;

    SSLConnection cctLdapSSLConnection = new SSLConnection();

    boolean authSuccess = cctLdapSSLConnection.sslBind(
        "ldap01.sys.loni.org", "uid=" + user.getUserName() +
        ", " + "ou=People ,ou=Admins ,dc=loni ,dc=org", password);

    if(authSuccess) {

```

```

        authflag = true;

        orgMask = orgMask + 1;
    }

    authSuccess = cctLdapSSLConnection.sslBind(
        "ldap01.sys.loni.org", "uid=" + user.getUserName() + ","
        + "ou=People,ou=lsu,ou=users,dc=loni,dc=org", password);

    if(authSuccess) {
        authflag = true;
        orgMask = orgMask + 2;
    }

    authSuccess = cctLdapSSLConnection.sslBind(
        "ldap01.sys.loni.org", "uid=" + user.getUserName() + ","
        + "ou=People,ou=ull,ou=users,dc=loni,dc=org", password);

    if(authSuccess) {
        authflag = true;
        orgMask += 4;
    }

    authSuccess = cctLdapSSLConnection.sslBind

```

```

("ldap01.sys.loni.org", "uid=" + user.getUserName() + ", "
+ "ou=People ,ou=uno ,ou=users ,dc=loni ,dc=org", password);

if(authSuccess) {

    authflag = true;

    orgMask += 8;

}

authSuccess = cctLdapSSLConnection.sslBind

("ldap01.sys.loni.org", "uid=" + user.getUserName() + ", "
+"ou=People ,ou=tulane ,ou=users ,dc=loni ,dc=org", password);

if(authSuccess) {

    authflag = true;

    orgMask += 16;

}

authSuccess = cctLdapSSLConnection.sslBind(

"ldap01.sys.loni.org", "uid=" + user.getUserName() + ", "
+"ou=People ,ou=su ,ou=users ,dc=loni ,dc=org", password);

if(authSuccess) {

    authflag = true;

    orgMask += 32;

}

```

```

authSuccess = cctLdapSSLConnection.sslBind
("ldap01.sys.loni.org", "uid=" + user.getUserName() + ","
+"ou=People ,ou=latech ,ou=users ,dc=loni ,dc=org", password);
if(authSuccess) {
    authflag = true;
    orgMask += 64;
}

authSuccess = cctLdapSSLConnection.sslBind
("ldap.cct.lsu.edu", "uid=" + user.getUserName() + ","
+ "ou=internal ,ou=People ,dc=cct ,dc=lsu ,dc=edu", password);
if(authSuccess) {
    authflag=true;
    orgMask += 128;
}

if(authflag == false)
    throw new AuthenticationException("User:" +
    user.getUserName() +
    "Unable to bind under provided name and password");
if(orgMask < 128) user.setOrganization("LONI");
if(orgMask == 128) user.setOrganization("CCT");

```

```
        if(orgMask > 128) user.setOrganization("HPC");  
        cctLdapSSLConnection.getLdapUserInfo(user);  
    }
```

The checkAuthentication method shown above tries the different CCT and LDAP servers using different types of Distinguished Names (DN). A DN is a name that uniquely defines a directory entry within an LDAP database and can be used to locate it within the directory tree.[44] Based on which of these bind requests return successfully, an organization mask is incremented by a specific power of 2. This is equivalent to setting particular bits to 1 or 0 in a binary number. The final value of the mask after all the tests provides information about which organizations the user belongs to. The LONI VO is made up of multiple organizations and hence the user DNs consist of different organizations. These are all considered as the LONI VO and the user organization is set to LONI.

The GridSphere LoginService was modified so that a user trying to login is not rejected if his login name is not found in the GridSphere database. The LDAPAuthService is invoked with a temporary user object which is discarded upon failure and stored into the GridSphere database along with all the information retrieved from LDAP. This reduces the number of queries to the LDAP server to only one at login. The password is not stored in the GridSphere database to minimize security risks. The User needs to explicitly change his organization if he desires to do so because user information is retrieved from LDAP only if the user is not found in the GridSphere database. Fig 4.2 shows the sequence of messages involved in authenticating a user and storing the User object in the database. The following code fragments capture the changes made to the LoginServiceImpl class that allow handling LDAP users.

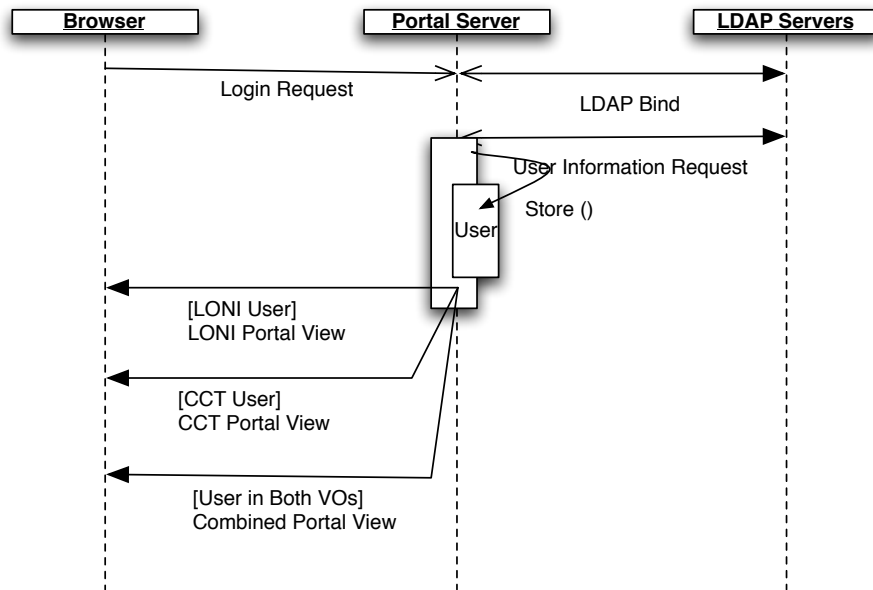


FIGURE 4.2: Authentication Message Sequence

```

User user = null; boolean ldapUsed = false;
// if using client certificate , then don't use login modules
if (certificate == null) {
    if ((loginName == null) || (loginPassword == null)) {
        throw new AuthorizationException(getLocalizedText
            (req, "LOGIN_AUTH_BLANK"));
    }
    // first get user
    user = activeLoginModule.getLoggedInUser(loginName);
} else {
    log.debug("Using certificate for login :" + certificate);
    List userList = userManagerService.getUsersByAttribute
  
```

```

        ("certificate", certificate , null);

        if (!userList.isEmpty()) {
            user = (User)userList.get(0);
        }
    }

if(user==null) {
    SportletUserImpl tempUser = new SportletUserImpl();
    tempUser.setUsername(loginName);
    user = tempUser;
    ldapUsed = true;
}

...

// second invoke the appropriate auth module
List modules = this.getActiveAuthModules();
Collections.sort(modules);
AuthenticationException authEx = null;
Iterator it = modules.iterator();
log.debug("in login: Active modules are: ");
boolean success = false;
while (it.hasNext()) {
    success = false;

    LoginAuthModule mod = (LoginAuthModule) it.next();

```

```

log.debug(mod.getModuleName());

try {
    mod.checkAuthentication(user, loginPassword);
    success = true;
} catch (AuthenticationException e) {
    String errMsg = mod.getModuleError(e.getMessage(),
        req.getLocale());
    if (errMsg != null) {
        authEx = new AuthenticationException(errMsg);
    } else {
        authEx = e;
    }
}

if (success) break;
}

if (!success) {
    user=null;
    log.debug("Failed all LDAP, throwing Auth Exception.");
    throw authEx;
}

else {
    if(ldapUsed) {

```

```

        userManagerService . saveUser ( user );

        Set groups = portalConfigService . getPortalConfigSettings ( )
        . getDefaultGroups ( );

        Iterator itg = groups . iterator ( );

        while ( itg . hasNext ( ) ) {

            PortletGroup group = ( PortletGroup ) itg . next ( );

            this . groupService . addUserToGroup ( user , group );

        }

        roleService . addUserToRole ( user , PortletRole . USER );

    }

    return user ;

```

4.3 Implementation of Portlets Using a VO Centric View

4.3.1 GridPortlets

GridPortlets are a set of portlets that allow users to perform grid tasks on a set of grid resources. These resources are stored in a registry that is read from a XML resource description file and stored in memory as a database. Each resource is assigned a Distinguished Name (DN) that is used to identify it and also establish the hierarchical relationship between the types of resources. For example, a hardware resource such as `bluedawg.loni.org` provides a software resource such as GRAM. The XML description does not capture the fact that resources belong to VOs because the top level element is a hardware resource and other service resources belong to the hardware resource. The hardware resource itself is not contained in a resource that can be used to identify a VO nor does any of the hardware resource properties have a VO parameter. In order to add this

information to the DNs, the label field of the hardware resource was used to prefix the VO name with the resource name since the DN contains the label. This is close to the ideal situation which would have the VO as the root element of the DN instead of as a prefix. The VO prefix in the DN allows querying for resources that belong to a particular VO and this feature is exploited to return the suitable set of resources by adding the getResources method call that accepts a VO as an argument and returns the appropriate set of resources. Below is a code snippet used to modify the ResourceRegistryService that provides the described functionality.

```
public List getResources(String VO) {  
    return getNeededResources(Resource.class.getName(), VO);  
}  
...  
public List getNeededResources(String classname, String VO) {  
    synchronized (lock) {  
        List result = null;  
        if (!VO.equals("CCT") && !VO.equals("LONI")) {  
            VO = "HPC";  
        }  
        try {  
            String query = "";  
            if (VO.equals("HPC")) {  
                query = "from " + classname;  
            } else {
```

```

        query = "from " + classname + " r where r.Dn like
                '%" + VO + "%'";
    }
    log.debug("getNeededResources query " + query);
    result = pm.restoreList(query);
} catch (PersistenceManagerException e) {
    log.error("FATAL: could not retrieve list of "
            + classname + " resources!" + e);
    result = new ArrayList(0);
}
return result;
}
}

```

The JobSubmissionService and the FileBrowserService have been modified to take advantage of the new functions in the ResourceRegistryService that accept a VO as an argument. New functions have been added to these that accept a VO argument and return Job and File resources belonging to a particular VO. The render methods then call these functions to get a list of resources that are used to fill in the UI components that display the resources to choose from:

```

public List getJobResources(JobType type , String VO) {
    // Translate to a default type if this is our base type
    if (JobType.INSTANCE.equals(type)) {

```

```

        try {
            type = (JobType) getDefaultTaskType(type);
        } catch (TaskException e) {
            return new Vector(0);
        }
    }

    // Get the service for the given job type
    JobSubmissionService service
        = (JobSubmissionService) registry .
        getTaskSubmissionService(type);

    if (service == null) {
        return new Vector(0);
    }

    return service.getJobResources(type, VO);
}

...

public List getJobResources(JobType type, User user, String VO) {
    // Translate to a default type if this is our base type
    if (JobType.INSTANCE.equals(type)) {
        try {
            type = (JobType) getDefaultTaskType(type);
        } catch (TaskException e) {

```

```

        return new Vector(0);
    }
}

// Get the service for the given job type
JobSubmissionService service
    = (JobSubmissionService) registry .
    getTaskSubmissionService(type);

if (service == null) {
    return new Vector(0);
}

return service .getJobResources(type , user , VO);
}

```

Similarly in the FileBrowserService:

```

public List getFileResources(String VO) {
    log.debug("Returning all file resources");
    List fileResources = new Vector(10);
    log.debug("Adding file resources to list from " +
        fileBrowserServices.size() +
        "file browser services");
    Iterator fbs = fileBrowserServices.values().iterator();
    while (fbs.hasNext()) {

```

```

FileBrowserService fileBrowserService =
(FileBrowserService) fbs.next();

List nextList = fileBrowserService.
getFileResources (VO);

log.debug("Adding " + nextList.size()
+ " file resources to list from " +
fileBrowserService.getClass().getName());

fileResources.addAll(nextList);
}

return fileResources;
}

```

Finally the UI components make use of these service methods to provide the appropriate view. The JobSpec component, for instance, has:

```

protected List getJobResourceList() {

String VO = getUser().getOrganization();

if (!VO.equals("LONI") && !VO.equals("CCT")) {

VO = "HPC";

}

return jobSubmissionService.getJobResources

(jobType, getUser(), VO);

}

```

The FileBrowser component uses:

```
public void setDisplayModeFlag() {  
    log.debug("Setting display mode flag");  
    displayModeFlag = Boolean.FALSE;  
    try {  
        LogicalFileBrowserService logicalFileBrowserService =  
            (LogicalFileBrowserService) getPortletService  
                (LogicalFileBrowserService.class);  
        List fileResources = logicalFileBrowserService.  
            getFileResources(getUser(), getUser().getOrganization());  
        if (fileResources.size() > 0) {  
            log.debug("There are " + fileResources.size()  
                + " logical file resources");  
            displayModeFlag = Boolean.TRUE;  
        }  
    } catch (Exception e) {  
        log.error(  
            "Unable to get logical file browser service", e);  
    }  
}
```

Similar modifications have been made to the ResourceBrowser and ResourceRegistry portlets to accomodate using VOs.

4.3.2 Other Portlets

All other portlets that have been developed in this thesis work make use of the organization info that is available from the User object in portlets using the Gridsphere API or using the portlet.xml user organization parameter in JSR-168 portlets. The AllocationPortlet, which is a JSR-168 portlet has two methods that are used to obtain the user name and user organization which are used to determine the VO:

```
private String getUsername(PortletRequest request) {  
    Map userInfo = (Map) request.getAttribute  
        (PortletRequest.USER_INFO);  
    if(userInfo == null) return "Guest";  
    String username = (String) userInfo.get("user.name");  
    return username;  
}  
  
private String getOrganization(PortletRequest request) {  
    Map userInfo = (Map) request.getAttribute(  
        PortletRequest.USER_INFO);  
    if(userInfo == null) return "ALL";  
    String userOrg = (String) userInfo.get  
        ("user.organization");  
    return userOrg;  
}
```

```
}
```

The view rendering is handled in the Java Server Pages (JSP) depending on the user's VO as returned by the above two functions. JSP allows server-side Java code to be embedded in plain HTML pages, thus allowing access to objects and methods to determine the user's VO.

4.4 Challenges in Creating a Multi-VO Portal

Most software engineering endeavors are faced with numerous challenges and building the multi-VO portal had its own set of challenges. This section summarizes the problems that were faced and the solutions that were developed to solve those problems.

4.4.1 Users and VOs

In order to successfully incorporate VO-based views in the portal, it is necessary to identify the VO of the user. Although GridSphere provides user information, this is limited to the organization of the user and not the VO. Adding the VO field to the User object would require modifying the GridSphere database that holds user information. This was avoided by computing the VO from the organization whenever the VO information was needed.

4.4.2 Grid Credential Management

The other major challenge was in managing credentials for users from different VOs having certificates from different certificate authorities (CA). In order to solve this problem, the acceptance policy of the myproxy server was changed to accept certificates issued by both the CCT and LONI CAs. A wide acceptance policy will accommodate any other VOs that need to be integrated into the HPGC portal.

4.4.3 Resource Management

The biggest challenge proved to be adding VO information to resources in the GridPortlets resource registry. Once again, the goal was to do this without having to modify the GridSphere database and the Resource Object. Hence the VO information was added to the resource label field that is part of the DN of the resource and all its child resources. This way it is possible to query the resource registry database for resources belonging to a particular VO.

5 Design and Components

5.1 Layout

“A portal layout is a template that defines how a set of selected portlets should appear on a page.”[22] Each page can be divided into rows and columns and portlets can be placed in them. Every user can create his/her own layout in a portal environment. This is an important personalization feature in a Portal. The Guest User layout of HPGC Portal contains Weekly News Portlet that informs users of upcoming LONI and CCT events. Currently this portlets is used to announce scheduled and unscheduled outages and maintenance notifications. There is an About HPGC Portlet that gives users a brief description what the HPGC portal is and the collaborators. There is a login Portlet that enables user to login to the Portal using their LDAP credentials. It also shows the load, queue and job information of both LONI and CCT Grid. The idea behind designing this layout is that the users can get the most important information about the grid resources without logging into the portal. This page is a public page which can be viewed by anyone who is interested to know about CCT and LONI grids. After logging into the Portal, the user has flexibility

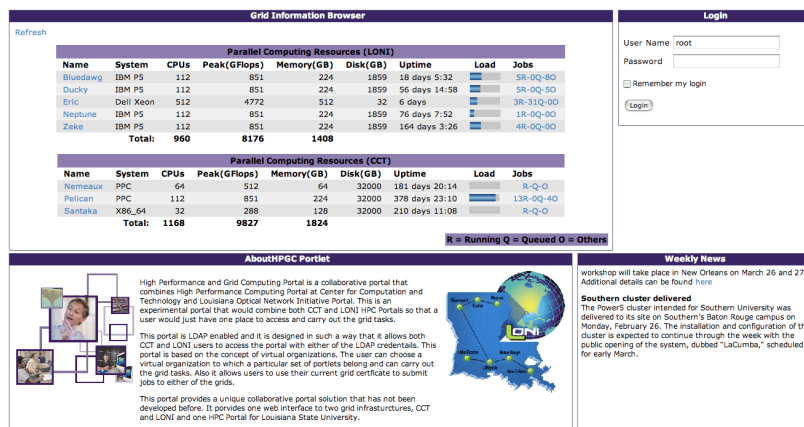


FIGURE 5.1: Guest User Layout of HPGC Portal

to design his/her own layout by creating new pages and placing the portlets they are authorized to

access on these pages. Figure 5.1 shows the Guest layout of the HPGC portal.

5.1.1 Themes

Themes are used to define the over all look and feel of a Portal environment. The theme defines color schemes, font sizes and types. Different themes can be developed from the default theme by changing the colors and other text elements. A user can choose to apply any theme from the set of defined themes that he/she is authorized to access. This gives a personalized look and feel to the Portal pages. The HPGC Portal has two sets of themes, a purple color scheme and a default blue color scheme. The administrator can define the default theme for the Guest layout. A user does not have permission to change the Guest layout.

5.2 Portlets

5.2.1 Portlets Summary

Table 5.1 summarizes the work that has been done on each of the portlets in the HPGC portlet to provide the LDAP and VO based view functionality.

TABLE 5.1: Summary of work

Name	Modifications
Profile Settings	None
GPIR Browser	Major
Grid Portlets	Major
Allocations Portlet	Developed from scratch
MRTG Portlet	Developed from scratch
RancidConfig Portlet	Developed from scratch
AboutHPGC Portlet	Minor
Weekly News Portlet	Developed from scratch

5.2.2 Profile Settings Portlet

This is an out-of-the-box GridSphere portlet that is used to display and edit user information, as seen in Figure 5.2. This information depends on the credentials that were used to login the first

time and does not automatically change there after. The user can manually edit the information and save it to the local database. This edit will not change information in LDAP. Also the organization that is retrieved defines the VO to which user belongs. This, however, can be manually changed will alter the affiliation. The three options that a user has are CCT, LONI and HPC. HPC is the affiliation for the users that belong to both CCT and LONI and would like to have access to both sets of portlets.

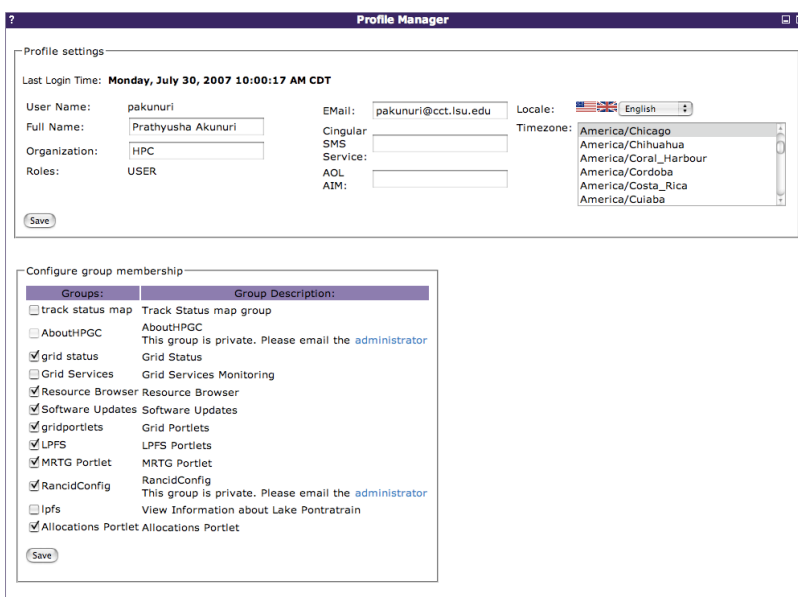


FIGURE 5.2: Profile Settings Portlet

5.2.3 GPIR Browser Portlet

GPIR stands for GridPort Information Repository. This portlet is developed by GridPort Toolkit team. The GPIR browser portlet provides static and dynamic information about resources. This information is retrieved by the portlet from a GPIR service (a web service). This Portlet has three views: a systems view, a machine view and a job view. The systems view summarizes the resource information of all the machines that belong to a VO. The machine view gives a more detailed information on the about each individual machine. This information is static. The job view gives

user the current job status on each individual machine.[17]

The GPIR Browser portlet has been modified to provide views based on the VO to which a user belong. A combined CCT and LONI user views information about both the grids in a single portlet, whereas the other two sets of users view just the grid information about the VO to which they belong. The three views are shown in Figures 5.3 and 5.4.

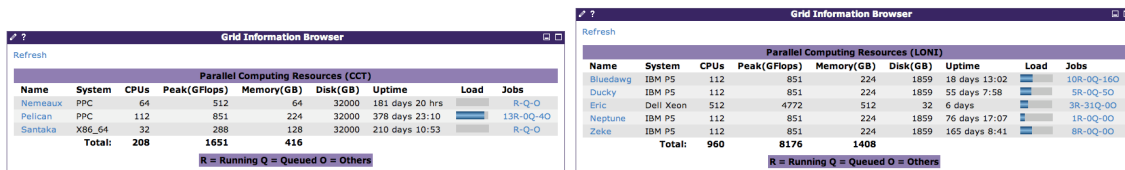


FIGURE 5.3: (i) GPIR Browser Portlet - CCT (ii) GPIR Browser Portlet - LONI

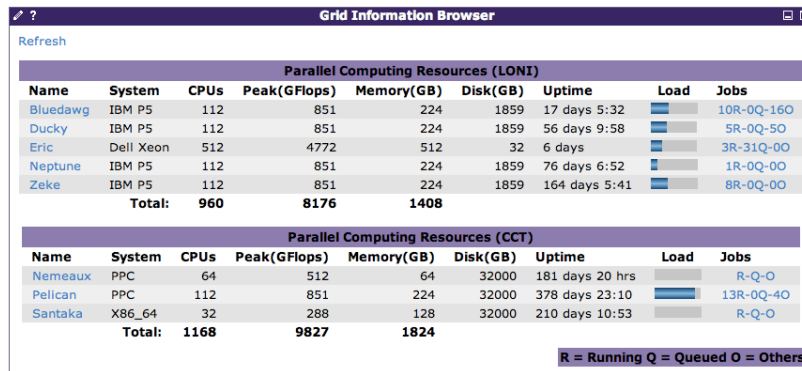


FIGURE 5.4: GPIR Browser Portlet - HPC

5.2.4 GridPortlets

As discussed in the previous chapter, GridPortlets have been modified to implement VO centric view.

The Resource Registry portlet, as shown in Figure 5.5, is accessed only by the administrator to add or edit the grid resources belonging to each of the VOs.

The credential manager portlet, as shown in Figure 5.6, appears the same to all users. This portlet is used to retrieve the credentials from the my-proxy sever configured with the portal. This

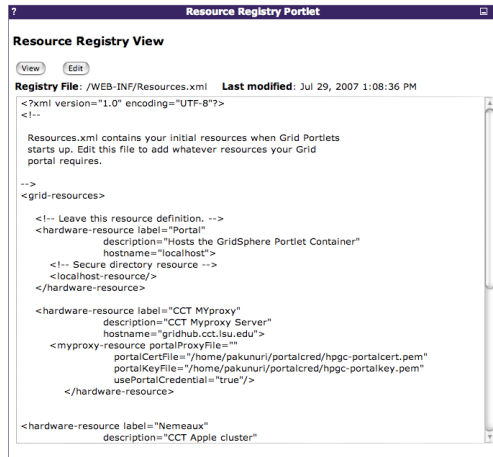


FIGURE 5.5: Resource Registry Portlet

credential expires every time the user logs out of the portal but the my-proxy credential remains active for seven days by default. This credential is used by other GridPortlets such as the job submission portlet and file transfer portlet.

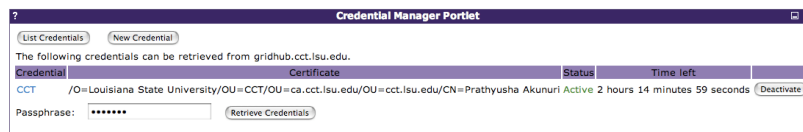


FIGURE 5.6: Grid Credentials Portlet

The Grid File Browser portlet changes its view depending on the VO to which the user belongs. This is a very useful portlet as it provides a easy GUI for transferring files across grid. This portlet allows the HPC user to move files from CCT Grid to LONI and vice-versa. This portlet allows the user to browse through the files on any grid resource, copy, move, delete, upload, download, view files and directories or make a new directory. Also this portlet provides a list of all the file management activities that are carried by user on the grid. Figures 5.7 shows exclusive CCT and LONI views whereas Figure 5.8 shows the view for a HPC user.

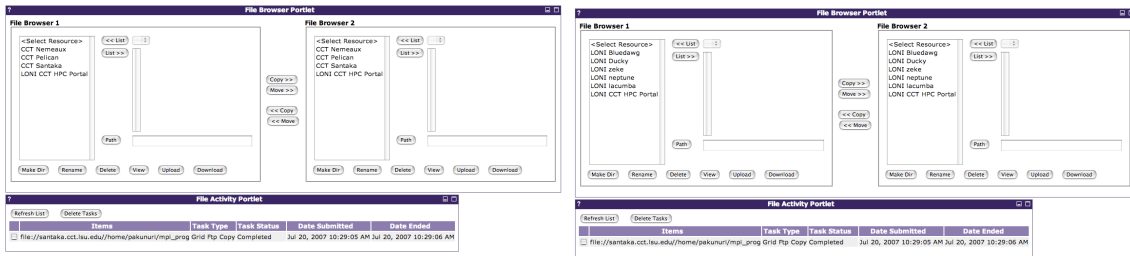


FIGURE 5.7: (i) Grid File Browser Portlet - CCT (ii) Grid File Browser Portlet - LONI

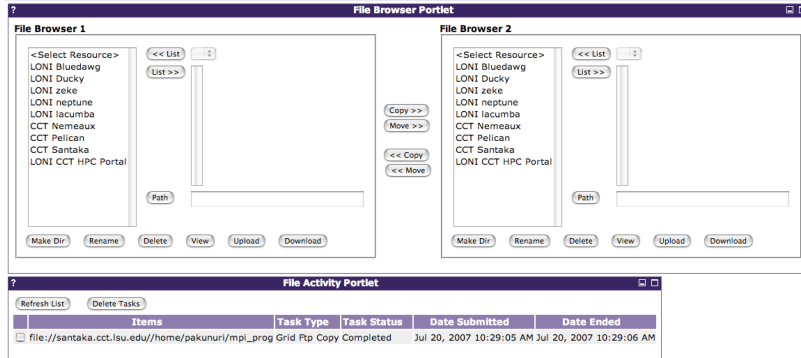


FIGURE 5.8: Grid File Browser Portlet - HPC

The most important portlet in GridPortlets is the job submission portlet. This portlet changes its view depending on the user's VO. This allows users to submit their jobs via a GUI rather than using a complicated command line interface. This portlet provides a complete list of jobs that have been submitted by the user in the past. It allows the user to submit a previous job while changing variables such as number of CPUs required, memory required and choosing from the available queues and schedulers, etc. The three views of this portlet are shown in Figures 5.9 and 5.10.

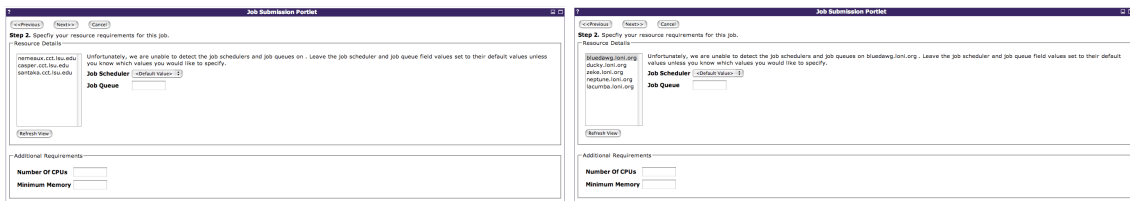


FIGURE 5.9: (i) Job Management Portlet - CCT (ii) Job Management Portlet - LONI

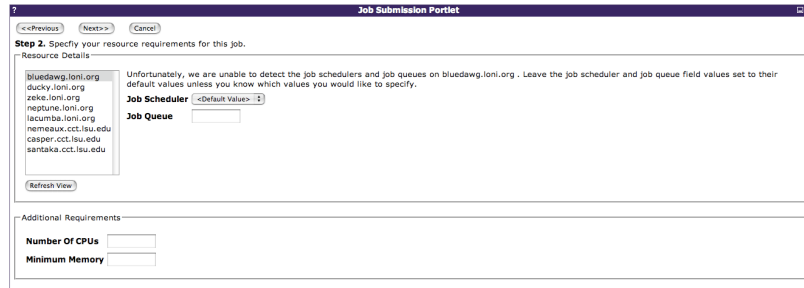


FIGURE 5.10: Job Management Portlet - HPC

5.2.5 Allocations Portlet

This is a JSR-168 compliant portlet developed to access allocation information from the Gold database. It gives users their allocation information such as the project name, allocation deposited, balance available, start date and the end date of the allocation. If a user does not have any active allocations, the portlet displays a link to request allocations for the appropriate VO. This portlet has three different views depending the user class. The three views are as shown in Figures 5.11 and 5.12.

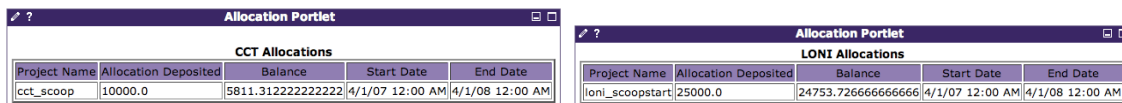


FIGURE 5.11: (i) Allocation Portlet - CCT (ii) Allocation Portlet - LONI

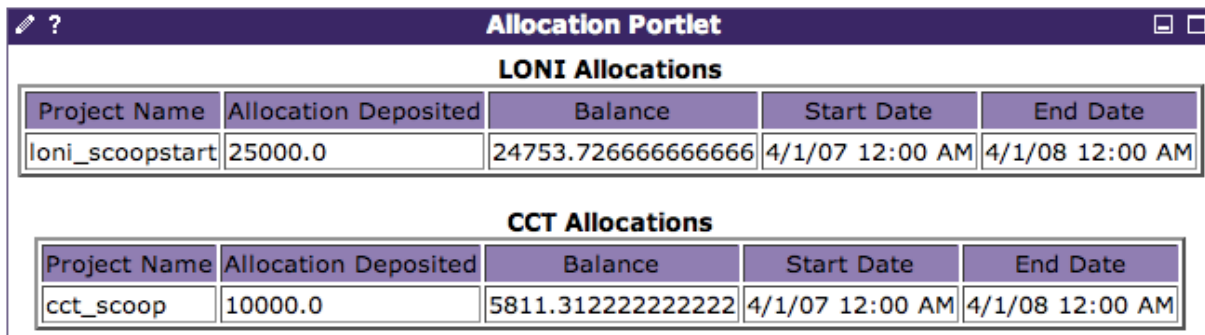


FIGURE 5.12: Allocation Portlet - HPC

5.2.6 MRTG Portlet

This is a GridSphere portlet that interfaces with LONI MRTG interface to display the traffic across all LONI routers. This portlet is exclusively developed for LONI and is accessed only by LONI users; CCT users do not see it. This portlet has two drop down menus as shown in Figure 5.13, one to select one of the six LONI partner schools and the second to choose the interface from LONI to one of the routers belonging to that particular school. It displays daily, weekly, monthly and annual network statistics of incoming and outgoing traffic of that router. This is a very useful portlet for users to determine the network conditions and predict the best time to carry out tasks that require high bandwidth.

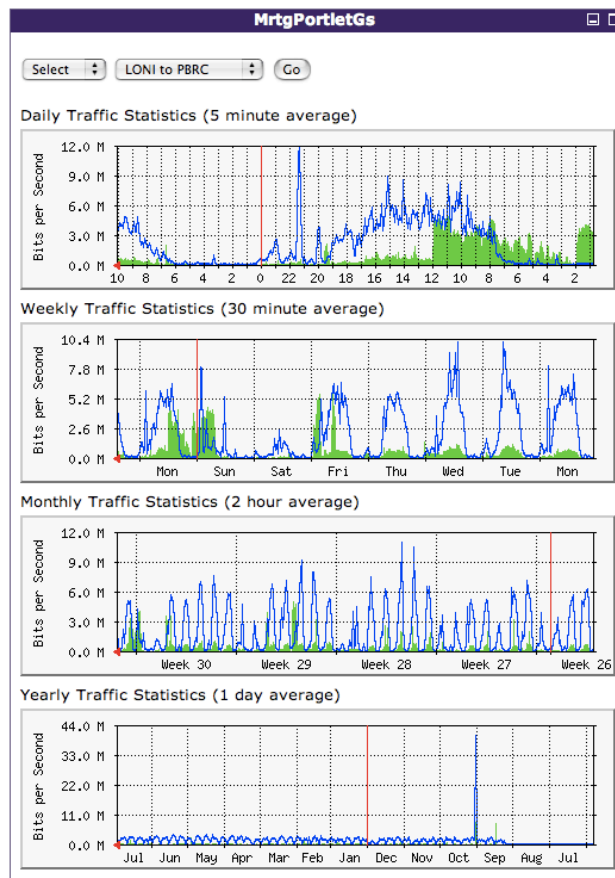


FIGURE 5.13: LONI MRTG Portlet

5.2.7 RancidConfig Portlet

This is custom JSR-168 compliant portlet developed exclusively for LONI network administrators.

This is a private portlet that can be viewed only by a specific group, the network administrators.

This portlet allows the users to choose one of the routers and view the configuration of that router.

It also allows users to download the configuration to the local machine. Work is in progress to add

search and compare functions to the existing portlet. Router configurations are backed up using

CVS every night. It is proposed to implement CVS browser functionality into this portlet so that

the administrators have access to all the versions of the router configurations. The current state of

the portlet is shown in Figure 5.14.

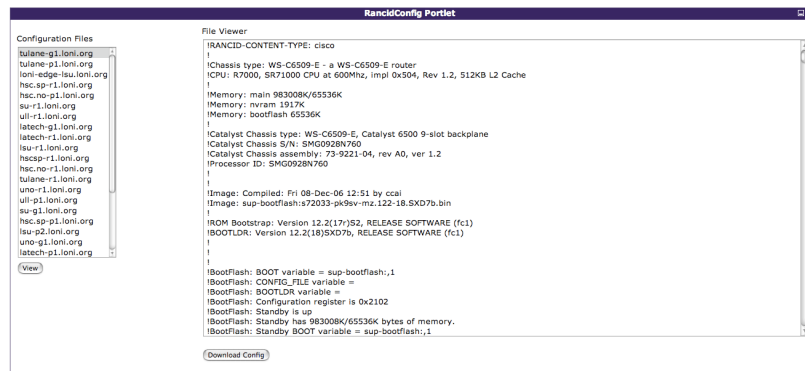


FIGURE 5.14: LONI RancidConfig Portlet

5.2.8 AboutHPGC Portlet

This is an information portlet deployed on the guest page of the HPGC Portal, as shown in Figure

5.15. This portlet gives a brief description of the HPGC Portal and the partners associated with it.

It is a static portlet developed from one of the out-of-the-box GridSphere template portlets.

5.2.9 Weekly News Update Portlet

This is a custom JSR-168 compliant portlet developed to display important news and outages as

shown in Figure 5.16. This portlet shows news items from files that are in a specific format.

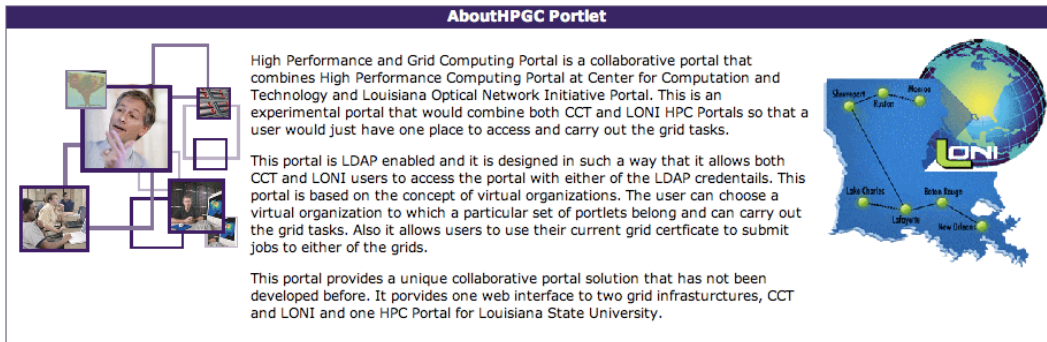


FIGURE 5.15: AboutHPGC Portlet

Every news item has a start date and an expiration date and only currently active news items are displayed. This portlet displays news items in vertical scrolling motion. High priority news items gets displayed first in the order. We have developed a PHP interface to add and edit the news items.¹



FIGURE 5.16: Weekly News Portlet

¹“Short for Hypertext Pre-processor) is a server-side, HTML embedded scripting language used to create dynamic Web content. PHP scripts can be embedded in HTML code and use similar syntax to the Perl and C programming languages.” [27]

6 Conclusions and Future Work

6.1 Advantages of Integrating CCT and LONI HPC Portals

As discussed in Chapter 2, there are many advantages in integrating High Performance Computing Portals. Through this thesis work, we successfully integrated the CCT HPC and LONI Grid portals. The users belonging to either or both organizations will have the following advantages in using the HPGC portal for their grid related tasks.

- Users of both CCT and LONI HPC will have to access just one portal instead of two separate portals in order to carry out their grid tasks with an easy to use GUI.
- All the required portlets are available in one place, to let these users carry out their grid tasks without any effort across both CCT and LONI grid machines.
- Once a user has a CCT or LONI user account, he/she automatically has access to the portal, since it is LDAP-enabled and supports both CCT and LONI LDAPs. There is no additional step for requesting a login for portal.
- The GridPortlets in the HPGC portal allow users to copy and move files from CCT HPC machines to LONI machines. This can save a lot of time for users carrying out their tasks across both the grids simultaneously. The HPGC Portal also allows users to submit jobs on both the grids with the same credential that is retrieved in the portal.
- The HPGC portal provides information about the resources, allocations, software installed, scheduled outages and other useful information about both the grids which avoids browsing through various web pages on multiple web sites to get this information.

6.2 Related Work

As part of the research, a survey of other related high performance computing portals, such as TeraGrid and SURAGrid, has been conducted.

6.2.1 TeraGrid Portal

The TeraGrid Portal (shown in Figure 6.1) was developed using the GridSphere Portal Framework. It has a combination of out-of-the-box and custom portlets deployed in the portal. The TeraGrid

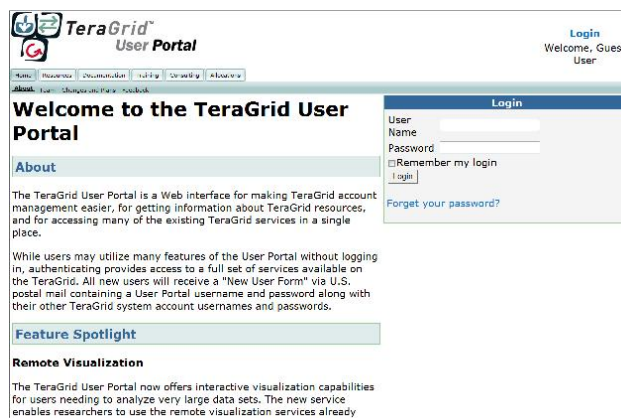


FIGURE 6.1: TeraGrid Portal Login Page [23]

Portal also uses the GPIR Resource browser portlet developed by GridPort to provide the load, queue, memory and other system related information about all the TeraGrid resources. It has a custom HPC queue prediction portlet (see Figure 6.2) that helps user predict the wait time on each of the grid resources based on the number of processors needed for the job to run and approximate runtime. The Teragrid portal also has portlets like remote visualization, portlet showing user about his/her accounts on all Teragrid resources, etc. It also has a ticketing system integrated within the portal that users can use to report problems. It also has an allocation request page displayed in the portal. Also it provides documentation on portal as well as resource information. The TeraGrid portal can incorporate some of the portlets in the HPGC portal such as GridPortlets that allow users

to submit jobs to machines easily. Also the LDAP authentication might be useful to tie in the portal login to the authentication mechanism of the various TeraGrid sites. The HPGC portal can make use of the batch queue prediction and the ticketing system portlets that have been integrated into the TeraGrid portal.

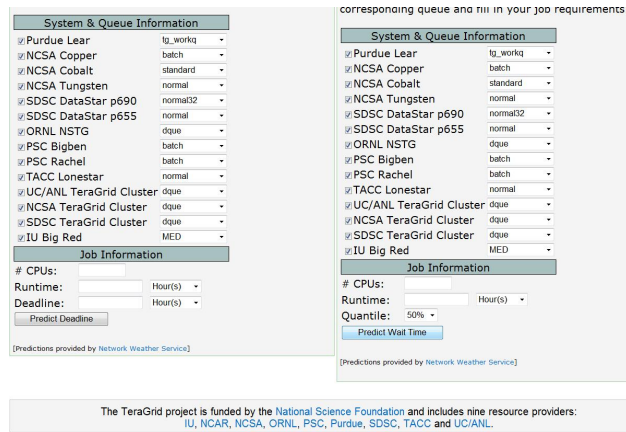


FIGURE 6.2: TeraGrid HPC Queue Prediction Portlet [23]

6.2.2 SuraGrid Portal

The SURAGrid Portal is a completely functional Portal with a large user base from different SURA partners. It was developed using the Grid Portal Framework. The portlets deployed were developed as part of the GridPort Toolkit.[17] The portal has grid based portlets (see Figure 6.4) such as

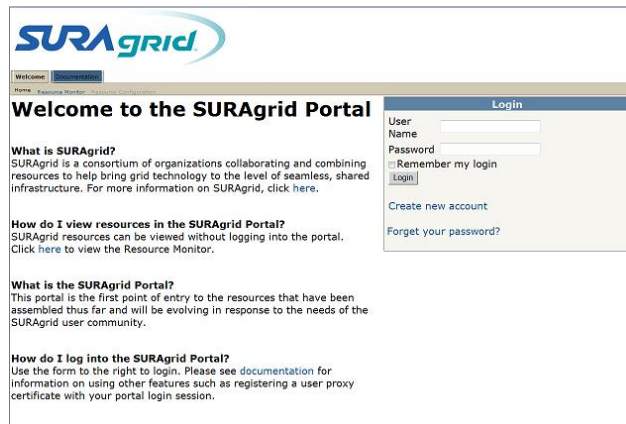


FIGURE 6.3: SURAGrid Portal Login Page [25]

proxy management, resource browser and configuration, job submission and file management, developed by the GridPort team. Users request a login by filling in a form. Then the SURAGrid team then verifies the user’s authenticity and creates a login (see Figure 6.3). Complete user documentation that describes the portal as well as basic information about the SURAGrid resources are provided as part of the portal. The HPGC portal in it’s current state is more advanced than the SURAGrid portal and it can provide a number of features that will be useful to SURAGrid portal users such as LDAP authentication against the SURAGrid LDAP and partner site LDAPs, job submission and file management from the GridPortlets which are more advanced than the ones provided with GridPort for example the SURAGrid job submission portlet provides a very simple form interface which is not very useful for selecting queues etc. The allocation portlet will be useful once SURAGrid adopts some form of allocation management for the compute resources.

Name	Institution	System	CPUs	Peak Gflops	Memory Gbytes	Disk Gbytes	Test Status	Current Load	Current Jobs
acStrocks	Georgia State University	ACS Dell Dimension 3000 Rooks Cluster	6	37	6	240	Click Here		R-Q-Q
AGL72_LM	University of Michigan	Michigan SURAGrid Gatekeeper	16		32	80	Click Here		R-Q-Q
Bandera	The University of Texas at Austin	TACC DGC Cluster	8	48	8	144	Click Here		R-Q-Q
Calstab	Texas A&M University	Calstab Optplex Cluster	306	1971	306	3500	Click Here		R-Q-Q
CSMlabde	University of Wisconsin at Lafayette	IBM BladeCenter H820 Cluster	12	87	20	25	Click Here		R-Q-Q
CSC	George Mason University	GMU CSC Center Prod	14	20	22	400	Click Here		R-Q-Q
Grid04	University of Virginia	Dell Rooks Linux Cluster	40	240	60	400	Click Here		R-Q-Q
Grid1	University of Virginia	HP Linux Cluster	3	26	3	325	Click Here		R-Q-Q
Janus	Georgia State University	GSU IBM Powerpc+, p575	138	866.1	256	4000	Click Here		R-Q-Q
ksu-csgr01	Kennesaw State University	Kennesaw Dell PowerEdge Xeon Cluster i96_i4	14	70	21	490	Click Here		R-Q-Q
Medusa	University of Alabama at Birmingham	Dell PowerEdge Cluster	32	18	16	100	Click Here		R-Q-Q
Mileva	Old Dominion University	Sun Fire X2100 Dual Core Opteron Cluster	8	24	8	25	Click Here		R-Q-Q
Pelican	Louisiana State University	IBM SURAGrid Frame at LSU	368	2797	960	10000	Click Here	148.0Q	
Scopio	University of Alabama at Huntsville	ADCIRC-2006 Cluster	8	48	8	100	Click Here		R-Q-Q
Taroon	Texas A&M University	Taroon Opteron Cluster	256	718	384	3600	Click Here		R-Q-Q
Weland	Texas Tech University	AMD Athlon MP 2000+	64	69	64	780	Click Here		R-Q-Q
Total:			1285	7019.1	2176	32609			

FIGURE 6.4: SURAGrid Portal after user Logs in [25]

6.2.3 Comparison

Table 6.1 shows the comparison of various features of the SURAGrid and Teragrid portals with the HPGC portal. This table gives an idea of the few important requirements of an HPGC portal and how these have been addressed in the various portals. The table shows that the HPGC portal uses LDAP for user management which appears to be the best solution in terms of scalability and

convenience for users. The job submission and file management portlets are also the most advanced in the HPGC portal as they are part of the GridPortlets which have been modified to incorporate VO-based views. Resource Monitoring in all three portals makes use of the GPIR service. The HPGC portal GPIR portlet has been modified to display up-time and VO specific resources. The HPGC and TeraGrid portals also provide allocation portlets that allow users to view the status of their allocations on the compute resources. The HPGC portal also provides the MRTG and RancidConfig portlets that allow users to monitor network and router statistics. The NWS portlet which is part of the TeraGrid portal provides valuable batch queue prediction and is a portlet that will be incorporated into the HPGC portal in future.

TABLE 6.1: Comparison of HPC Portals

	HPGC Portal	TeraGrid Portal	SURAGrid Portal
User Management	LDAP	Custom	GridSphere Database
Job Submission	GridPortlets(modified)	None	GridPort
File Management	GridPortlets(modified)	None	File Listing
Resource Monitoring	GPIR(modified) and GridLab Testbed Portlet	GPIR	GPIR(modified)
Allocation Management	Custom Portlet	Custom Portlet	None
Network Monitoring	Rancid and MRTG Portlets	None	None
Batch Queue Prediction	Work in Planning(NWS)	NWS	None

6.3 Other Related Solutions

As part of the thesis work, research was conducted to investigate other solutions that provide user and certificate management such as PURSE[28]. It was decided to use LDAP as the solution because the current functionality provided by PURSE is limited by the fact that the MyProxy server can only store one certificate per user. This limits the ability to identify users that belong to multiple VOs. Since the HPGC portal uses LDAP for authentication and it is possible to bind to multiple LDAP servers, it becomes easy to identify users belonging to multiple VOs.

6.4 Conclusion

As part of this thesis work, we have successfully developed the HPGC portal that will help the LSU grid community in carrying out their grid tasks more efficiently using this GUI-based easy-to-use portal. The VO-based implementation of the portal views is a unique solution that can be scaled to add more VOs without much effort. This can be a starting point for major collaborations. Three major challenges were overcome and solutions were developed that can be used to provide VO-based services to portal users.

The integration of multiple HPC Portals not only helps bring together grid communities but it is also cost effective in terms of hardware and maintenance. It helps the developers and administrators of the portal to concentrate on a single infrastructure and make it more reliable and highly available.

6.5 Future Work

The HPGC portal was designed to integrate the CCT and LONI VOs. Hence some of the code in the HPGC portal uses the CCT and LONI elements such as the LDAP servers, DNs etc. Although the code has been written such that these are easy to modify, future work might be to put these configuration parameters into XML files that the portal can read and thus make it easier to change the configuration without having to recompile the source. Also certain changes may be considered to the GridSphere core so as to be able to provide better support for handling VOs such as modifying the User object to include VOs and also the resource registry. Integrating notification mechanism in GPIR browser portlet to send out SMS, email or instant message as per user's preference to notify availability of large number of processors, change in the load, etc. Another important change would be to add a VO field to the User object, so that we would not need to use the organization to determine the VO. This will give the users flexibility to be in multiple VOs and VOs will be able

to represent application and project groups such as LONI, CCT, LSU, HPC, SCOOP, UCoMS etc.

Some additional features can be added to the HPGC portal that would be helpful to the users such as integrating a ticketing system with the portal can be very useful, as users would not need to go to a different website to report the problems and also will be able to view all their tickets and statuses in one place. It is suggested that GridSphere implements the LDAP module that is developed as part of this thesis work. This makes it easy for future upgrades to the HPGC portal and other portals that utilized these features. As part of this work we also realized that having roles associated with portlets allows better access control than having roles associated with groups. Hence adding this feature will make GridSphere better suited for a VO-based portal solution.

Bibliography

- [1] Wikipedia Definition of Grid Computing, http://en.wikipedia.org/wiki/Grid_computing
- [2] <http://www-fp.mcs.anl.gov/foster/Articles/WhatIsTheGrid.pdf>
- [3] <http://www.ibm.com/developerworks/grid/library/gr-portlets/>
- [4] <http://grid.ncsa.uiuc.edu/myproxy/>
- [5] http://en.wikipedia.org/wiki/High-performance_computing
- [6] <http://www.motive.co.nz/glossary/portal.php>
- [7] <http://publib.boulder.ibm.com/infocenter/wpdoc/v510/index.jsp>
- [8] <http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/portal>
- [9] <https://www.sdn.sap.com/irj/sdn/netweaver>
- [10] <http://www.eweek.com/article2/0,1895,1372096,00.asp>
- [11] <http://www.liferay.com/web/guest/home>
- [12] <http://portals.apache.org/pluto/arch.html>
- [13] <http://portals.apache.org/jetspeed-2/>
- [14] <http://www.globus.org/toolkit/about.html>
- [15] <http://www.clusterresources.com/pages/products/torque-resource-manager.php>
- [16] <https://allocations.loni.org/about.php>
- [17] <http://gridport.net/main/gpir-browser-2/index.html>
- [18] <http://www.gracion.com/server/whatldap.html>
- [19] <http://oss.oetiker.ch/mrtg/>
- [20] <http://www.shrubbery.net/rancid/>
- [21] <http://www.gridisphere.org/gridisphere/gridisphere>
- [22] <http://www.novell.com/documentation/extend52/Docs/help/Director/books/pgPortalLayouts.html>
- [23] <https://portal.teragrid.org/gridisphere/gridisphere>
- [24] http://grids.ucs.indiana.edu/ptliupages/publications/Teragrid_Big_Red_Portal-camera-ready.pdf

- [25] <https://gridportal.sura.org/gridsphere/gridsphere>
- [26] <http://www.bl.uk/about/strategic/glossary.html>
- [27] http://www.google.com/search?hl=en&defl=en&q=define:PHP&sa=X&oi=glossary_definition&ct=title
- [28] <http://www.extreme.indiana.edu/portals/purse-portlets/>
- [29] Ian Foster, Carl Kesselman and Steven Tuecke, “Anatomy of the Grid”, ANL,ISI
- [30] Wolfgang Gentzsch, “Ask The Expert: What exactly is Grid Computing?”, <http://www2.cio.com/ask/expert/2002/questions/question1566.html>
- [31] “GridLab WebPages: WP - 1 GAT”, <http://www.gridlab.org/WorkPackages/wp-1/>
- [32] Chongjie Zhang, Ian Kelley and Gabrielle Allen, Grid Portal Solutions: A Comparison of GridPortlets and OGCE, Concurrency and Computation: Practice and Experience, Volume 19, Issue 12, p. 1739-1748, 2007.
- [33] Michael Russell, Gabrielle Allen, Ian Foster, Ed Seidel, Jason Novotny, John Shalf, Gregor Laszewski and Greg Daues, The Astrophysics Simulation Collaboratory: A Science Portal Enabling Community Software Development, Journal on Cluster Computing, Volume 5, Issue 3, Pages 297-304, (2002).
- [34] Chongjie Zhang, Chirag Dekate, Gabrielle Allen, Ian Kelley and Jon MacLaren, An Application Portal for Collaborative Coastal Modeling, Concurrency and Computation: Practice and Experience, Volume 19, Issue 12, p. 1571-1581, 2007.
- [35] Ruxandra Bondarescu, Gabrielle Allen, Greg Daues, Ian Kelley, Michael Russell, Ed Seidel, John Shalf and Malcolm Tobias, The Astrophysics Simulation Collaboratory Portal: a Framework for Effective Distributed Research, Future Generation Computer Systems, Volume 21, Issue 2, Pages 259-270, (2005).
- [36] Gabrielle Allen, Kelly Davis, N. Dolkas, N. D. Doulamis, Tom Goodale, Thilo Kielmann, Andre Merzky, Jarek Nabrzyski, J. Pukacki, and Thomas Radke, Enabling applications on the grid: A Gridlab overview, International Journal of High Performance Computing Applications, Volume 17, Number 4, Pages 449-466, (2003).
- [37] Gabrielle Allen, Tom Goodale, Hartmut Kaiser, Thilo Kielmann, Archit Kulshrestha, Andre Merzky, Rob V. van Nieuwpoort, “A Day in the Life of a Grid-Enabled Application: Counting on the Grid”, GGF-12 Grid Applications Programming Interfaces
- [38] K. Davis and T. Goodale, “D1.2 Technical Specification” ID: Gridlab-1-D1.2-0002.TechnicalSpecification.
- [39] “Grid Resource Management System”, <http://www.gridlab.org/WorkPackages/wp-9/index.html>.
- [40] S.Bradner, “RFC 2119: Key words for use in RFCs to Indicate Requirement Levels,” <http://www.ietf.org/rfc/rfc2119.txt>.

- [41] J.Rumbaugh, I. Jacobson, G. Booch, “The Unified Modeling Language Reference Manual”, Addison-Wesley, Reading, Massachusetts, 1999.
- [42] Jeffrey, E. F. Friedl, “Mastering Regular Expressions”, O’Reilly & Associates; 2nd edition (July 15, 2002).
- [43] Henry Spencer, “regex — POSIX 1003.2 regular expressions”, Unix Manpages, chapter 7.
- [44] Monash University Public Key Infrastructure - Definitions and Acronyms, <http://www.its.monash.edu.au/staff/security/staff-only/certs//theory/pki-definitions.html>.

Vita

Prathyusha Akunuri-Venkata is a native of South India. She completed her Bachelor of Science in Electrical and Electronics Engineering from JNTU, Hyderabad, India. She enrolled in Master of Science in Electrical Engineering program at Louisiana State University in 2003. She worked as Graduate Assistant for Center for Computation and Technology at LSU. She was a member of the Grid Computing and Portal groups and was involved in various projects. In June 2006, she accepted full-time employment with Louisiana Optical Network Initiative (LONI) as an IT-Analyst while continuing her graduate studies. She plans to pursue a Master of Business Administration degree while continuing her current employment.