5-2006

Architectural Tradeoffs for Unifying Campus Grid Resources

Amy Apon
Clemson University, aapon@clemson.edu

Bart Taylor
Acxiom Corporation

Architectural Tradeoffs for Unifying Campus Grid Resources[1]

Bart Taylor and Amy Apon

Acxiom Corporation
Bart.Taylor@Acxiom.com

University of Arkansas
Amy.Apon@uark.edu

Abstract. Most universities have a powerful collection of computing resources on campus for use in areas from high performance computing to general access student labs. However, these resources are rarely used to their full potential. Grid computing offers a way to unify these resources and to better utilize the capability they provide. The complexity of some grid tools makes learning to use them a daunting task for users not familiar with using the command line. Combining these tools together into a single web portal interface provides campus faculty and students with an easy way to access the campus resources. This paper presents some of the grid and portal tools that are currently available and tradeoffs in their selection and use. The successful implementation of a subset of these tools at the University of Arkansas and the functionality they provide are discussed in detail.

1 Introduction

The University of Arkansas has an assortment of resources ranging from small three or four node heterogeneous test clusters to a 128 node supercomputer. Faculty members and research students use several of these resources on a regular basis and would benefit from an approach to unifying the interfaces. Job submission, for example, is generally done on the machine where the job will be executed. Allowing jobs to be submitted to resources remotely in a scriptable way could save time and reduce the possibility of making errors when submitting them individually. Multiple courses also use these resources as tools to teach parallel programming and grid computing concepts. For example, a large class that teaches MPI programming could benefit from a campus grid by permitting students to determine how busy a resource is and to submit their jobs to a machine with a smaller load. Centralizing access to these resources and presenting a single web-based interface for common tasks such as submitting jobs would provide users with easy access to their jobs or data on any of the systems from a web browser.

Creation of a centralized resource was prompted by desires and requests to make using campus resources easier for both students and faculty. These desires form the basis upon which this project was undertaken and have been continually referenced as design decisions were made. One benefit of the organization at the University of Ar-

[1] Support for this project is provided in part by NSF Grant DUE #0410966.

kansas is that ownership and administration of most of the cluster resources on campus is limited to a few individuals. This means that consensus is much easier to obtain as to what can or should be done not only to the structure as a whole but also to the individual resources. Listed here are some of the requests and any specific design characteristics that have been agreed upon:

1. Single Sign-On

   A user should be able to sign on one time to a centralized environment and submit any number of jobs to any resource of their choosing without the need to connect to each resource individually for authentication. For example, a single authentication could provide access to an environment where resources can be monitored and valid Grid Security Infrastructure (GSI) credentials [1][2] can be obtained and used to submit jobs or transfer files.

2. Scheduler Independent

   A user should be able to submit a job to a resource without any knowledge of the local scheduler. A user should, for example, be able to submit a job to a resource using Load Share Facility (LSF) [1] as its local scheduler and immediately submit the same job with the same syntax to a different resource using Sun Grid Engine (SGE) [2] or Portable Batch System (PBS) [3] and never detect a difference in the behavior of the system.

3. Single Proxy

   A user should be able to obtain grid credentials from a proxy server, which will minimize the annoyance of having to login to particular resource to obtain grid credentials and submit jobs.

4. Resource Information Available

   A user should be able to obtain general and current utilization and job queue information for any resource connected to this environment that chooses to provide it. This will allow users to decide which resource is capable of handling a job or will be able to complete the job first.

5. Maintainable and Extendable

   The solution should be approached so that the result is easily maintainable and extendable internally as well as extendable by eventually allowing non-UA grid members. For example, the solution could allow members of the Southeastern Universities Research Association's test bed grid (SURAgrid) [4] to access the resources.

6. Resource Independent

   Each individual resource administrator should make the final decision as to whether specific users have access to submit jobs or otherwise use the resource. Simply obtaining credentials for the grid will not grant access to every resource by default.

7. Metascheduling

   A user should be able to submit a job without concern for where the job is executed. While this requirement could produce several benefits, it is beyond the scope of this research. An additional layer can be added later to facilitate this functionality.

The remainder of this paper is organized as follows: Section 2 surveys available grid tools and the components needed to make them operational. Section 3 surveys portal frameworks as well as three portlet packages that provide grid functionality. Section 4 analyzes the tradeoffs of an implementation of a subset of these grid tools at the University of Arkansas. Section 5 describes the selection and installation of a por-

tal framework. The three packages of grid portlets are installed and compared for functionality. Section 6 presents conclusions. The final result is a portal interface to the University of Arkansas grid resources.

2 Grid Tools

While some grid tools are available for the Windows and Macintosh platforms, Linux is widely accepted as the standard for grid applications. Tools surveyed here for building a grid include Linux cluster installation packages, local schedulers, Globus Toolkit, Condor, and Certificate Authorities.

2.1 Cluster Installation Packages

A popular choice for administrators wanting to add a cluster resource to a grid is Rocks [5]. Rocks is based on Red Hat Linux. With Rocks, the front-end of the cluster is installed first with the base operating system and any additional rolls, optional software and its configuration, desired. After the initial setup, rebooting a compute node reimages the node from the image on the head node.

Open Source Cluster Application Resource (OSCAR) [8] is another cluster installation tool. OSCAR, however, does not include the operating system by default. Once the head node is properly installed, all Red Hat Package Manager (RPM) files from the installation disks need to be copied to a specific folder in the root directory in order to create the image for the compute nodes. OSCAR is similar to Rocks, but allows for a little more flexibility in the installation which makes the process more complex. Rocks provides a very simple single unit installation that takes little effort to install but provides less flexibility.

There are several application packages that are needed when adding a resource to a grid and are not directly grid related. The Java SDK [9] is a requirement for virtually all of the tools since most of the grid software has been developed in Java. JDK 1.4.2 is generally required as opposed to 1.5 for compatibility reasons. Because grid services are an extension to web services, a servlet container, specifically Apache Tomcat [10], is another requirement. Other basic tools include, but are not limited to, Apache Ant, Apache Maven, GNU make [11], and a JDBC compliant database like MySQL [126] or PostGreSQL [137] with its adapter.

Many problems can be avoided by realizing that a specific version of a software package may be required for a successful installation of a specific tool. For example, two versions of Ant may be required if an old version of the Globus Toolkit is installed as well as the GridPort [14] portal package. Multiple installations of these packages are possible, but careful attention should be paid to how the environment variables are set.

2.2 Local Schedulers

A local scheduler is necessary for the actual queuing and handling of jobs submitted to any resource. Local schedulers determine the order of execution of jobs in the queue, the specific nodes on which a job will execute in the case of a cluster, load balancing, file staging, and other administrative tasks. Choices for a scheduler include some mentioned above and also PBS, PBS Pro [15], SGE, LSF, LoadLeveler [16], Computing Center Software (CCS) [17], and Condor [18]. Some grid tools, such as the Globus Toolkit, interact directly with the underlying local scheduler and may require a specific scheduler (e.g. PBS, LSF, or Condor in this specific case) for which adapters have been written. Third party adapters may also be available (e.g. currently only for SGE in this case). The open source solutions are popular to keep costs down. The commercial version of some of these local schedulers, such as Platform's LSF and Altair's PBS Pro, offer improved and supplemental functionality.

Description	PBS	LSF	Condor
Submit to Queue	qsub	bsub	condor_submit
Status of Queue	qstat	bjobs	condor_q
Delete from Queue	qdel	bkill	condor_rm
Suspend job inQueue	qhold	bstop	condor_hold
Release suspended job	qrls	bresume	condor_release
Modify Queue order	-	btop	condor_prio
Queue history	-	bhist	condor_history
Move job to another Queue	qmove	-	-

Fig. 1. Comparison of common local scheduler commands

Each scheduler has its own implementation of the features it supports. A grid tool such as the Globus Toolkit must interface with these various implementations. Administrative settings such as the scheduling algorithm do not affect Globus, but differences such as the name of the command to submit a job to the queue or check the status of a job are of great importance. Figure 1 shows several of the basic functions in PBS, LSF, and Condor. Globus uses adapters installed on a resource for a specific scheduler to translate a generic set of commands into commands for that scheduler.

2.3 Globus Toolkit

The Globus Toolkit [19] has become widely accepted as the standard tool for building a grid architecture and boasts success stories including GriPhyN [20], Open Science Grid [21], TeraGrid [22], and NEES grid [23]. The Globus Toolkit provides identification, authentication, authorization, and security for connecting the resources of the grid, as shown in Figure 2. The infrastructure allows for strict local control of a given resource but still allows users from anywhere to perform a variety of tasks related to resource and service discovery, file management, system and job monitoring, and job submission. It maintains security with standard X.509 [23] certificates and offers its

own certificate authority, SimpleCA, if one is not already provided at the location. Globus Toolkit also provides a proxy server, MyProxy Server, to reduce the number of certificate requests for users with accounts on multiple resources. Globus Toolkit is a toolkit containing many components that provide the necessary functionality for two or more resources to be connected in a grid scenario. True functionality is achieved when the toolkit is used as a foundation for a collaborative environment.

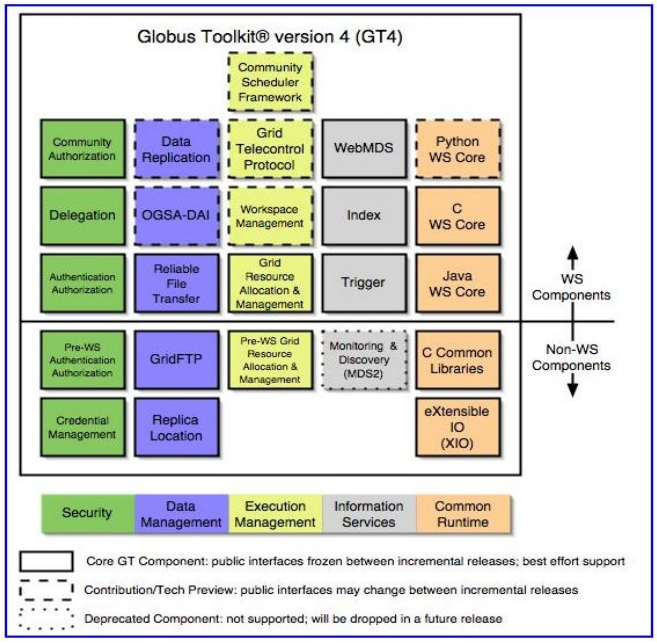

Fig. 2. Globus Toolkit Version 4 architecture [20]

The Globus Toolkit supports a standard interface called Grid Resource Allocation Manager (GRAM) that allows any remote client to submit a job to the local system without any knowledge of the underlying components. GRAM adapters are configured to the underlying local scheduler. This is a useful tool for allowing access to users who may not have a way to get information on the scheduler in use otherwise. It also offers a way to add resources to a centralized interface such as a portal without making the centralized interface more complicated each time a new resource is added. WS-GRAM provides reliable job submission and cancellation through web services.

The Globus Toolkit also includes support for data movement operations, including GridFTP a server called globus-gridftp-server, and a command-line client called globus-url-copy. GridFTP provides standard file movement to and from remote servers and the local machine as well as third-party transfers from remote host to remote host. Grid Security Infrastructure (GSI) is used on both control and data channels for security. Multiple data channels are used for parallel transfers, and the striping feature allows multiple endpoints at the sender, receiver, or both to transfer files faster than a single machine is capable. The provided client tool can translate between the

FTP, HTTP, HTTPS, and POSIX file I/O protocols on the client side. The latest server implementation released with Globus Toolkit 4.0.1 has been shown to support up to 1800 clients.

Reliable File Transfer (RFT) extends the GridFTP functionality to provide better recovery from failure, including client-side failure. RFT provides subscription to changes in state or polling for state information. RFT behaves like a job scheduler for file transfers. A job description detailing which files need to be transferred with sources and destinations is submitted by a user. The job is stored in a database in case of system failure, and the transfers are executed. If the RFT service fails, the jobs are resumed when RFT is restarted. This is also helpful for users whose connection may not be maintained for the duration of the transfer, such as laptop users. The job is handed off to a scheduler that performs the transfers on the user's behalf without the user being directly involved after the submission. Users can request the status of the transfers or use the Web Services Resource Framework tools provided with the Globus Toolkit to subscribe to notifications of state change.

An emerging mechanism for security in the Globus Toolkit is GridShib [25], which is a blending of the Globus Toolkit and Shibboleth. With Shibboleth [26], all attributes about a user are stored at the user's home institution and distributed in response to requests from remote services through interaction with an Identity Provider. GridShib is comprised of two plugins, one each for Globus and Shibboleth, that allows a Globus grid service provider to request user attributes from an Identity Provider. The GridShib approach offers finer grain control of who has access to which resources while maintaining all the core functionality of a standard Globus installation.

2.4 Condor

Condor is a useful batch processing utility for queuing and handling many serial jobs that are run simultaneously on many individual computers. These computers can be dedicated Condor machines or regular workstations that serve as compute nodes when they are in an idle state. The ability for the Condor Manager to return to the submit machine only workstations that are not currently being used makes it an attractive solution for scaling an existing resource quickly. Hardware that already exists on the network can be easily added to a Condor pool. This is useful if a large supply of machines that get little usage already exists or if the funding for a dedicated resource is not available. Authentication can be achieved using X.509 certificates just as the Globus Toolkit does, and a Condor CA is also provided.

Figure 3 shows the structure of Condor job submission. Compute machines initially alert the Condor Manager to their existence and node information such as processor speed and available memory. When a user submits a job, the submit machine advertises the job requirements specified by the user to the Condor Manager, which returns a set of available machines that meet these requirements. The submit machine then communicates directly with the compute machines.

Condor also supports a mechanism called flocking that offers another pool of execute machines if the local pool is busy. If job cannot be run on the local pool it flocks the job to the remote pool if the remote pool has enough available resources. This is useful in the situation where two institutions or departments each have their own Condor

pool but agree to share resources. The group with the smaller pool can still execute a large number of processes.

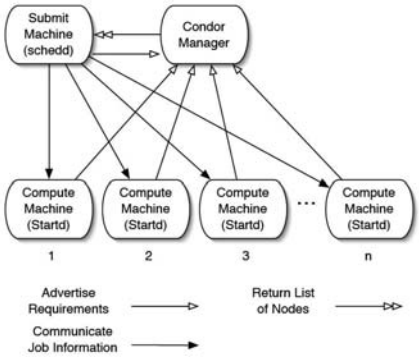

Fig. 3. Condor job handling

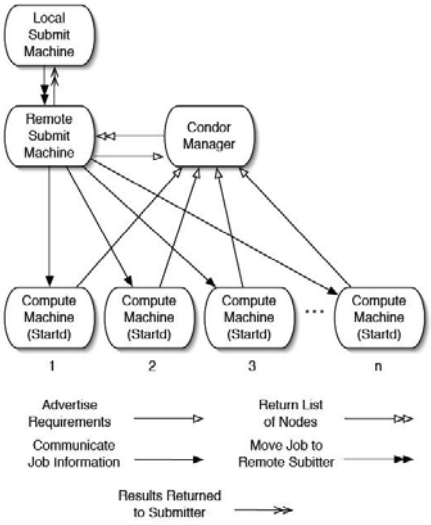

Fig. 4. Condor-C job handling

Condor requires many network ports to be open per compute node to communicate with the manager that passed it the job. Under a normal flocking scenario across a firewall, the number of ports becomes can become unmanageable before the size of a Condor pool is really useful. The Condor-C [27] approach reduces the number of required open network ports of this by sending the entire job to a submit machine on the remote resource pool and allowing it to handle the interaction with each compute node. The remote submit machine will return the results to the local submit machine for retrieval when the job completes. This capability is shown in Figure 4.

Condor-G [28] is an extension of Condor that works with the Globus Toolkit. Condor blends the benefits of the Globus infrastructure with Condor's approach to batch processing. Instead of using Globus repeatedly to start many jobs, a single Condor job request can be sent to a Condor-G server that starts all of the jobs on Globus resources by translating the Condor job into the GRAM format that Globus understands. This is accomplished by modifying the Condor job description, called a ClassAd, to specify the type of Globus resource, the remote job manager, and the local scheduler on the remote machine. Globus only needs to be installed on the resource targeted for execution. It does not need to be installed on the submission resource. The submission machine uses GridFTP for staging files to the Globus resource. Condor-G does not replace Globus. Condor-G uses Globus for its security and grid capabilities. It attaches Condor's idea of batching jobs to the Globus framework by translating jobs and passing them to Globus.

2.5 Certificate Authorities

Certificates are the default method of security in grid computing. They are based on the Public Key Infrastructure (PKI) with asymmetric public and private keys. A Certificate Authority (CA) generates these certificates that contain, among other information, the public key and unique Distinguished Name for the subject of the certificate. Many institutions and organizations already have a working Certificate Authority. For those who do not already have one, but do have a Globus Toolkit installation, SimpleCA may be used. SimpleCA is a wrapper for the OpenSSL CA. The CA provides a mechanism for signing grid credentials in the form of X.509 certificates. Resources can be configured to trust certificates from multiple CAs, thus allowing access to the resource from as many institutions as are desired. For institutions where all users will obtain certificates from the same CA, such as a university campus, SimpleCA may be sufficient. For more complex architectures, such as with virtual organizations, commercial Certificate Authorities are available.

A useful tool to ease the complexity of security for grid users is a grid certificate proxy server. A proxy certificate is a copy of the credentials that uses a different encryption key and generally has a much shorter life span than the original credentials. The shorter life span helps ensure that if a third-party manages to decrypt the certificate and attempts to use the credentials, they will not be valid for long. The Globus Toolkit provides such a server by default called the MyProxy Server. This allows a user of the grid who submits jobs from accounts on multiple machines at different times to keep only one set of valid grid credentials instead of needing credentials on every machine. Once a valid set of credentials is obtained, the user can easily store them in the proxy server. When the user needs to update a proxy for any account on any machine on the grid, it can be done through the proxy server. Figure 5 shows the steps of proxy management and the life expectancy for the credentials at each step.

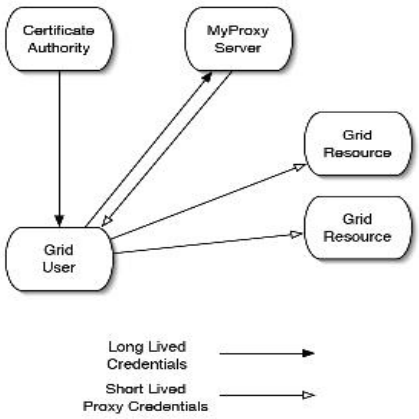

Fig. 5. Credential Management with a MyProxy server

For example, a user joins a university and is given accounts on resources A, B, and C. The user only needs to request one set of credentials for one of these accounts.

Once these credentials are granted, the user can place them in a MyProxy Server. When a proxy is required for submitting from any of the resources, A, B, or C, a request can be made to the MyProxy Server, and a valid proxy will be granted from the server. This also helps reduce the workload incurred by the Certificate Authority administrator. When a user has an account on any given resource the user will only need to generate a one certificate. The request for a certificate to be signed usually occurs when the user initially joins the grid.

3 Portal Tools

The idea of web portals has been around for many years. They are used every day by a wide range of users at enterprise intranets, e-commerce sites like Amazon, and news and information sites like Yahoo. These sites provide a central location where users can access a pool of services and information that the portal provides. Grid portals operate on this same basic principle. The portal just provides the framework in which these services are housed and accessed. Portlets are designed as pluggable modules that can be added or removed from the portal framework and provide a desired service or functionality to the overall portal environment. Many portlets designed are freely available for download and can easily be added to a compatible framework. This section describes both portal frameworks and portlets.

3.1 Portal Frameworks

The trend among most of the available portal packages today is a separation of the content from the portal framework. Within the last few years, a Java portlet specification, JSR-168 [29], has been adopted as the default standard for portlet development. Because of the development of this standard, co-led by Sun with strong industry backing and support, the portals can be tailored to support a single interface to the content portlets that they will be used to house. Many portals have chosen to offer this standard interface in addition to the proprietary interface already implemented in their design. In many cases, the proprietary interface is more stable and offers more advanced features to portlet developers. Portal frameworks considered for the UA Portal include WebSphere [30], Jetspeed [31], GridSphere [32], uPortal [33], and Grid Engine Portal (GEP) [34].

3.1.1 WebSphere

WebSphere Portal is an IBM product that provides an out-of-the-box web portal for enterprise settings. It provides authentication through a Tivoli LDAP server, user customizable pages stored in a DB2 database, access from desktops and Personal Data Assistants, single sign-on using encrypted cookies to access WebSphere protected resources with a credential vault for back-end systems, and local and remote portlets. Administrators can install and interact with remote portlets as if they are installed locally, and any change or update to the remote portlet is reflected to all users. Many

web portlets for enterprise use such as a people finder, document manager, email portlet, contact list, and search center are supplied. Although the WebSphere framework is geared toward an enterprise setting and not specifically directed toward grid computing, an existing implementation of WebSphere could be extended to support grid applications by adding portlets that make use of grid resources and support the WebSphere API or the JSR-168 standard.

3.1.2 Jetspeed

Jetspeed is an open source Apache project built as an Enterprise Information Portal, a gateway to a business' information, data, and knowledge base for employees or customers, and has been in development since 1999. From a web browser a user can access content from network resources such as applications and databases that are added to the portal. Jetspeed also works with templating applications such as WebMacro and Velocity to help separate content presentation and Java code in portlets. Template engines allow simple referencing of objects defined in Java code. Jetspeed-2 introduces improvements such as JSR-168 compliance and multi-threading. Authentication and authorization are achieved through the Java Authentication and Authorization Service (JAAS) , which is a Java implementation of the Pluggable Authentication Module (PAM). JAAS and PAM are easy-to-use modules that allow applications to remain independent of the actual authentication process. Role Based Access Control (RBAC) is also used to grant access to specific sections of the portal. RBAC uses the concept of a hierarchy of user permissions where the top level of the hierarchy has broader permissions, and it is implemented in a markup language. Apache Ant, Apache Maven, Java SDK 1.4.2, and the Tomcat servlet container are prerequisites for Jetspeed-2.

3.1.3 GridSphere

GridSphere is an open source project that is part of GridLab [35], a European Union funded venture into developing application tools and middleware for grid environments. GridSphere provides compatibility with the JSR-168 standard for portlet API and near full compatibility with the WebSphere 4.2 API. It also supports portlet development using Java Server Faces (JSF). A set of core portlets that handle user authentication, user customization, and administration features such as user, group, and portlet management are installed by default. It also supports localization, adaptation of language and content in 10 languages. All portal configurations are done by editing XML files. RBAC is implemented by default in the portal. It separates user accounts into guests, users, administrators, and super users. GridSphere builds on the Web Application Repository (WAR) deployment model, so support of third-party web applications is implicit. Only an empty file whose name corresponds to the WAR file for the application needs to be created to tell the container to load it. Portlets can be dynamically started, stopped, and reloaded without restarting the hosting container. GridSphere provides an automated file and template generator through an Ant command to facilitate developing custom JSR-168 portlets. Java SDK 1.4.2,

Tomcat 4 or 5, and Jakarta Ant 1.6+ are prerequisites for installing the GridSphere framework. A single Ant call is needed to install and deploy the portal.

3.1.4 uPortal

The Java in Administration Special Interest Group (JA-SIG) [36] is leading the way in the development of a university-wide Portal called uPortal, which is free for post-secondary institutions and targeted specifically at the needs of universities. Five member universities are working on its development. Currently, uPortal supports the implementation of channels, not portlets. However, as of version 2.3, portlets can be used via a conversion mechanism and the Pluto [37] portlet container, which is an Apache project. Version 3.0 is slated as the version where portlets will become the default content module and includes support for channels through a Channel-to-Portlet adapter. The uPortal implementation is beneficial for universities that only desire the framework to provide classes, interfaces, XSL stylesheets, and XML data files for their Java programmers to utilize when building a custom campus portal.

3.1.5 Grid Engine Portal

Grid Engine Portal (GEP), developed by Sun, was formerly known as Sun Technical Computing Portal. This portal is built on Sun's Grid Engine and SunONE Portal Server [38]. Given this specific set of software, Sun is able to provide a variety of services including single sign-on, fine-grained security policy rules, discovery of available back-end resources, and load-balancing. GEP is also based on the idea of channels. By default, it provides channels allowing the ability to create projects, view applications available from the portal, and submit and monitor jobs with email alerts of completion. Its downside is the lack of independence from other Sun specific software packages. Sun does claim, however, that it is only loosely tied to these packages.

3.2 Portlets

Portlets serve as the content that is plugged in to a portal framework, and this content can conform to the JSR-168 specification. While the portlets being considered here are directly related to the field of grid computing, they could just as easily be unrelated to grids and more akin to information portlets in popular web portals such as MSN and Yahoo. Several open source portlets have already been developed for the purpose of unifying grid resources. Development time can be better spent in creating portlets that offer new tools to the user since it is not necessary to redevelop existing code to get the basic functionalities. Discussed here are a few readily available portlet packages that supply much of the necessities for a useful grid portal such as job submission and file management.

3.2.1 GridPortlets

GridPortlets are released as a supplement to the GridSphere framework. They are JSR-168 compliant and are therefore compatible with other portals as well. In the GridPortlets version 1.1.3 support is provided for credential management with a MyProxy server, remote job execution via GRAM interface, remote file management via GridFTP, and information on current resources registered with the portal. By default the portlets are only compatible with Globus Toolkit 2.4. Support for GT3 and GT4 is under development in the gt3portlets and gt4portlets projects, respectively, but these projects have not yet been released for general use. They can be checked out of GridSphere's CVS repository if desired. Configuring the portlets involves editing the build.properties file in the root directory of the package and the webapp/WEB-INF/Resources.xml file. The build.properties file contains basic installation information such as necessary paths and debugging flags. The Resources.xml file contains any resources to be used from the portal and supplies sample XML for defining resource services. It can also be edited at run-time through a portlet installed with the package. A single Ant command will install and deploy the portlets.

3.2.2 OGCE

The Open Grid Computing Environment (OGCE) [39] develops open source portlets with the intention for them to be used under multiple portal frameworks. GridSphere and uPortal are specifically mentioned with instructions for installation. The version discussed here is OGCE2-RC5. In addition to the JSR-168 standard, they also support Velocity and Java Server Pages (JSP) based portlets by providing conversion tools to the JSR-168 standard. The group of 'Core Grid Portlets' they provide supplies portlets for proxy management with a MyProxy server, remote command execution via GRAM interface, remote file management using GridFTP. Grid Portal Information Repository (GPIR) [40] information services are provided using a GridPort [41] portlet which is discussed in the following section. All of these portlets work with Globus Toolkit 2.4, GT3.0, GT3.2 and GT4 and allow dynamic version selection from the portal. Basic configuration of the portlets is accomplished by editing the grid.installation.properties and project.properties files in the root directory of the package. The grid.installation.properties file defines on which hosts the MyProxy, GRAM, and GridFTP services are available. The project.properties file defines path information as well as portal container specific information. After this is completed, a simple Maven command will install and deploy the portlets automatically.

3.2.3 GridPort

GridPort, short for the Grid Portal Toolkit, is available through the University of Texas, in conjunction with the Texas Advanced Computing Center (TACC). GridPort 3.5 supports authentication via MyProxy server, command execution and batch jobs through a GRAM interface, file management through GridFTP, GPIR information services, job submission through the Community Scheduler Framework (CSF) and

Job Sequencer, which executes a linear sequence of these tasks that can be performed by GridPort. Support is only available for Globus Toolkit 3 resources from the GridPort 3 portlets. GridPort 4 was recently released and includes support for the Globus Toolkit 4 and Condor job submission. GridPort offers a prepackaged demonstration portal or individual portlets for download. There are configuration files in each of the portlet root directories and the overall demo portal root directory named project.properties that can be used before installation. Maven is used to automate the install process.

The GridPort Information Repository (GPIR) portlet is a front-end to the actual service. The system is based on remote providers that gather data about a system on a set schedule and send this data to an ingester service that collects the information for multiple resources. The portlet is designed to pull information out of this repository, a Hypersonic SQL database, and present it in a visual manner that can be adjusted before the portlet is deployed. To get functionality from this portlet, the remote provider must be installed on the resources to be monitored, and an ingester needs to be set up on a machine that is accessible by the portal or on the portal itself. The remote provider installation also installs SOAPLite by default for the user performing the install. It is recommended that this user have a separate account.

4 UA Grid

A grid was constructed at the University of Arkansas using Globus Toolkit 4.0.1 as the base software. Globus Toolkit version 4 was selected as it was the newest release of Globus available at the time that was compatible with the portlets most of interest. A tradeoff of this choice is that some portlets are not compatible with GT4. The single implementation requirement imposed for each resource that provides job submission is that it must support a GRAM interface. The only requirement imposed for availability of file management from the portal is an installation of a GridFTP server on each resource. Section 4 describes the selection of UA grid components and their installation, and command line testing.

4.1 UA Grid Component Selection and Installation

The components of the UAGrid that have been installed include security components, the Prospero cluster, the UAGridPortal host for the portal, and the Kite Condor cluster. The base implementation decisions and process for the security components or the grid, Prospero and the UAGridPortal computers are described here.

4.1.1 UA Grid Security Components

Obtaining grid credentials in the form of a X.509 certificate is a necessary step to using default installations of Globus, so SimpleCA was installed on a local Linux machine, and each resource accepts grid certificates signed by this CA. A MyProxy server is installed on a local Linux machine for housing these credentials.

Figure 6 shows how a user could interact with the UA Grid. From an account on the Prospero cluster, the user requests credentials from the CA as shown in step 1. After their return in step 2, the user places them into the MyProxy server as shown in step 3. From the portal interface, step 4 shows a request that is made for a proxy certificate, and that certificate is used to execute tasks on the grid resources in step 5.

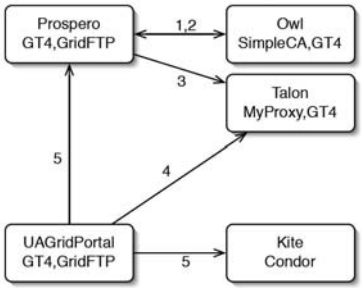

Fig. 6. User interaction with security components of the UAGrid

4.1.2 Prospero Cluster

Prospero is a 32 node dual-processor cluster that serves as a research tool for both faculty and students at the University of Arkansas. The decision was made to install the operating system and software stack with a fresh installation. The Rocks 4 cluster installation package is used on most clusters on campus, so it was the best choice for familiarity of administration. The standard Rocks install was performed with the PBS roll for the local scheduler and the Condor roll for later use by research students working with it. Originally, the SGE local scheduler was installed, but this resulted in a version inconsistency that caused all of the nodes to continually reboot. This in turn reimaged the node with the same flaw which did not solve the problem. This setup was eventually discarded and replaced with a reinstallation of the operating system with PBS as the scheduler. The Jakarta Ant and Java SDK packages required for installing the Globus Toolkit 4.0.1 were installed with Rocks by default.

A user named 'globus' was created to own the GT4 installation. The source for GT4 was unpacked and installed following the instructions provided by Globus. A host certificate was requested, signed by the local CA, and placed in the /etc/grid-security directory. Copies were made of the hostcert.pem and hostkey.pem files and saved as containercert.pem and containerkey.pem. The globus user then generated a request for a user certificate, and the signed certificate files were placed in the .globus directory of the globus user's home directory. An entry was made in the /etc/grid-security/grid-mapfile that corresponded to the certificate and username for the globus user based on the instructions in the Globus installation instructions. The GridFTP server was also started as a background process. GRAM was installed and configured with PBS as the local scheduler by following the instructions in Chapter 11 of the Globus installation guide. The GridFTP server was tested against the UAGridPortal's GridFTP server once its installation was completed. GRAM was also tested from UAGridPortal and from the MyProxy server.

4.1.3 UAGridPortal

The approach to building UAGridPortal was taken in much the same way as the installation of Prospero. Because it is a single dual-processor machine and not a cluster, Fedora Core 3 was chosen for the base installation. The necessary prerequisite software for the Globus Toolkit was installed, followed by a binary installation of the toolkit itself.

The decision of which portal framework to choose was influenced by the desire to utilize previously developed standards-compliant portlets. The effort involved in developing basic portlets such as a job submission portlet can be redirected toward creating necessary portlets with new functionality since several job submission portlets already exist. This meant selecting a JSR-168 compliant portal that could use some of the more popular portlet packages. GridSphere was chosen as the framework for the portal for this reason, as well as the demonstrated successful implementation of the portal for LSU and SURA and its well-developed user interface. The package is also very simple to install, maintain, and modify. Its prerequisites were obtained, and GridSphere was then built and deployed to Tomcat. Once the Tomcat startup script was run, the GridSphere portal was web accessible.

The UAGridPortal Fedora Core 3 installation was a 'custom' installation, which allowed the choice of which packages were preferred at install time. A hostname was chosen, and a static IP address and entry in the local DNS were obtained for it prior to starting the install. Several of the basic packages that are necessary for the Globus Toolkit and GridSphere were obtained from their respective websites. The Java SDK version 1.4.2 was obtained from Sun, and the JAVA_HOME environment variable was set after unpacking it in /usr/local/bin/j2sdk1.4.2_09. Apache's Ant 1.5.1 and Maven 1.0.2 packages were obtained from the Jakarta website and unpacked in the /usr/local/bin/jakarta-ant-1.5.1 and /usr/local/bin/maven01.0.2 directory as well. The corresponding ANT_HOME and MAVEN_HOME environment variables were set to these directories. Apache Tomcat was initially installed in the /usr/local/bin/jakarta-tomcat-5.5.4 directory as well, but some of the later installations looked for it in the home directory of the portal user. Since no other user on the system needs access to Tomcat, the install was moved to the portal user's account, and CATALINA_HOME was set to the root directory of the package.

At this point, the server was ready for the Globus Toolkit 4.0.1 installation. A user named 'globus' was created to own the Globus installation. The Fedora Core 3 binary was downloaded from the Globus site, and installed as the globus user to /usr/local/globus following the instructions provided on the Globus installation manual. The GLOBUS_LOCATION environment variable was set to point to this location. A host certificate was requested by this installation, and the local CA signed the host certificates which were placed in /etc/grid-security. The files hostcert.pem and hostkey.pem were copied to containercert.pem and containerkey.pem respectively. The globus user then requested a user certificate, which was then signed by the CA and stored in the globus user's ~/.globus directory. An entry was made in the grid-mapfile according to the instructions in the Globus installation manual. An instance of the GridFTP server was also started and run in the background.

Before the GridSphere 2.0.4 package was installed, the user 'portal' was created for the explicit purpose of owning the portal installation. This home directory is where the Tomcat installation resides. It was also discovered that some of the packages to

be installed required a different version of Apache Ant, so version 1.6.1 was obtained and installed to the portal user's home directory. The portal user's environment was configured to use this new installation directory. GridSphere was unpacked, installed and configured, and the GRIDSPHERE_HOME environment variable was set to /home/portal/gridsphere-2.0.4. The portlet packages by GridSphere, OGCE, and GridPort, were installed and configured.

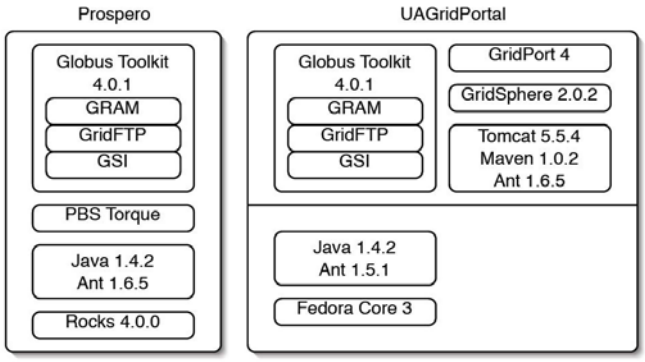

Fig. 7. Prospero and UAGridPortal software stacks

Figure 7 shows the software stacks for Prospero and UAGridPortal from the operating system at the bottom to the user interface at the top. Prospero offers a GridFTP and GRAM interface. UAGridPortal offers both of these as well as a portal interface.

4.2 Command Line Testing

All of the grid tools that have been installed on University of Arkansas resources are designed to be used from a command line. The grid was tested using this command line interface. The following sections describe tasks such as job submission, third-party file transfers, and credential management, and how they can be accomplished using the command line tools provided with the Globus Toolkit 4.0.1.

4.2.1 Credential Management with Proxy Certificates

Regardless of the preferred interface, once a user is given his first account on the grid, he must perform a certificate request from the new account. This is accomplished by logging to a grid resource and using the grid-cert-request command. This command creates three files in a subdirectory of the user's home directory: usercert.pem, usercert_request.pem, and userkey.pem. The command also gives instructions to email usercert_request.pem to a specified administrator email address for signing by the CA. The command causes a prompt to appear for a name, which will become the Common Name in the certificate granted, followed by a request and verification for a passphrase for the certificate. Once this is complete, information for mailing the certificate is displayed. The usercert.pem file is a placeholder that will be overwritten when the requested certificate is signed and returned. The certificate is usually returned with liberal permissions that must be changed to be granted a proxy. The

maximum allowable permissions are read and write permission for the owner and read permission for the group and other users. This can be achieved using the "chmod 644 usercert.pem" command. Once an account has valid credentials, there two options: 1) a proxy can be created using the credentials in this account on this resource, or 2) the credentials can be placed in a MyProxy Server repository for use from any resource on the grid that recognizes the CA that signed the certificate.

A user can create a proxy from the certificate by using the grid-proxy-init command on a resource where the grid credentials are stored. The command displays a prompt for the passphrase used in the grid-cert-request command, and then the expiration date of the proxy is displayed. If problems occur when using this command, the -debug option can be specified to provide additional information. The -verify option will force a check of the certificate for which the proxy is being created. Once this is accomplished the user is ready to use the grid tools that require valid credentials.

To store a credential in a MyProxy Server repository the user runs the myproxy-init command from a computer where the grid credentials are stored. The server must be set up and running, and either the MYPROXY_SERVER environment variable must be set or a fully qualified address of a grid resource that holds the user's certificates must be known. The -s option allows the name of the server to be specified in the command, and the -l option states the username for the proxy in the repository. The myproxy-init command needs to be run every time the credentials in the repository expire. By default the expiration time of the credential in the repository is seven days and this can be set with the myproxy-init command's -c option, in increments of an hour. Once a credential is stored in the MyProxy repository, the myproxy-logon command can be used to obtain a proxy credential.

4.2.2 Running Remote Commands with GRAM

Jobs are submitted to Web Services GRAM (WS-GRAM) with the globusrun-ws command. Once a proxy certificate is created, a user can specify the executable and arguments as well as the resource that should be used and many other job specific options. A simple job can be run with all of the options specified on the command line.

Specifying the -F option tells GRAM exactly which factory should be used for executing the specified job. An example of this is shown in Figure 8. The Hello World program is submitted to the ManagedJobFactoryService on the specified machine using port 8443. Leaving off the -F option will default to the localhost on the same port using the same factory for submission.

```
[bhtaylo@prospero ~]$ globusrun-ws -submit -F https://prospero.uark.edu:8443/wsrf/services/Managed
JobFactoryService -c /home/bhtaylo/hello
Submitting job...Done.
Job ID: uuid:14082438-5493-11da-a568-0002e3004b18
Termination time: 11/14/2005 22:16 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
[bhtaylo@prospero ~]$
```

Fig. 8. Job submission with globusrun-ws using the –F flag

The length of these commands can grow very quickly and become burdensome to the submitter, so an alternative is provided in the form of a job description file. This is an XML based file specified with the -f option on the command line.

This is only a subset of what can be done using GRAM, but it services the needs for most of the users on the University of Arkansas campus. Any testing or usage outside of these basic submission examples should work as described in the WS-GRAM user's guide.

The standard file management tool provided with GT4 is GridFTP. GridFTP allows not only normal transfers from remote hosts to the local machine and vice versa but also third-part transfers from one remote machine to another remote machine. A server is provided as well as a client, and the command-line client is suitable for scripting. However, no interactive client is provided. GridFTP operates on the same principle as FTP in regard to using both a command and data channel, but security is used on both. Multiple data channels can be opened for parallel transfers. Other features include reusable data channels to reduce the overhead associated with closing and reopening channels for many small files and partial file transfers.

The server is started by the globus-gridftp-server command. Typical options include -p for setting the port and –S to run the server runs as a detached background process. Many of the command-line options can be applied in a configuration file specified in the command or located in the $GLOBUS_LOCATION/etc/gridftp.conf file or /etc/grid-security/gridftp.conf. Only the first of these three options found was used in the University of Arkansas implementation.

The transfers are generally accomplished using the globus-copy-url command. Its arguments are in the order of source then destination. A multitude of options are available including recursive copying, specifying a list of files to transfer in a file, and the size and rate of the current transfer. Figure 9 shows a third-party transfer of a file from one remote machine to another.

```
[bhtaylo@talon bhtaylo]$ globus-url-copy gsiftp://uagridportal.csce.uark.edu/hom
e/bhtaylo/testFile.2 gsiftp://prospero.uark.edu/home/bhtaylo/testFile.2
```

Fig. 9. Globus-url-copy from remote host to remote host

4.2.3 Example Use Case

The amount of functionality available presents many different possibilities for how users might interact with the grid. A typical usage will begin with the user logging into one of the Globus resources. If the credentials in the MyProxy Server have expired, which is displayed when running the myproxy-logon command, then the grid-proxy-init -debug -verify command must be run on the resource holding the credentials the user requested from the CA, and the myproxy-init command should be run to put the credentials back into the MyProxy Server. On the original resource the user logged into, the myproxy-logon command will grant a valid proxy certificate.

Suppose the user needs to run a simulation based on the data files in the home directory on the current machine, but the current machine is not capable of running the simulation. The data files need to be moved to the target resource. The user may move all of the files in the datafiles directory on the local machine to the same directory on the supercomputer and start the simulation as:

```
% globus-url-copy
file:///home/user/simulation/datafiles/gsiftp://campus.supercomputer.edu/home/user/simulation/datafiles/
%globusrun-ws -submit -F
https://campus.supercomputer.edu:8443/wsrf/services/ManagedJobFactoryService
/home/user/simulation/simulation-binary
```

The simulation may run for several hours and terminate. The globus-url-copy command can also be used to copy the results back to the original host if desired.

5 UA Portal Design

Usage of the command line interface can quickly become complicated and almost burdensome, hence the desire of some users for a portal type interface. There is a tradeoff in using a portal, however. Since the simplicity of a graphical interface often removes the advanced functionalities of shell based commands, many users still desire the increased power that the command line offers, or simply prefer the interface to which they are accustomed.

Once the University of Arkansas' baseline resources had been installed and tested using the command line interface, three packages of portlets were installed to test the system and compare functionality. GridPortlets, developed by the GridSphere team, was a natural choice. With the portal framework chosen it was likely to install properly and also provides essential features such as obtaining proxy grid credentials, job submission, and file management. The OGCE portlet package was also installed and tested because it offers the same functionality with its own implementations as well as some other functions such as a queue viewer and resource monitoring. GridPort 3 was installed and tested because it supplies a demo portal that contains a few portlets as well. Its installation was a little less straightforward and is explained in detail along with the installation details of the rest of the portlets in this section. The release of GridPort 4 resolved these issues and was much easier to install. The portlet packages rely on a MyProxy server for obtaining credentials. The portlets were configured to use the MyProxy server and Prospero for job submission and GridFTP usage.

5.1 GridSphere and GridPortlets

GridSphere distributes GridPortlets along with the portal framework to provide grid functionality. GridSphere was extracted in the portal user's home directory, and the "ant install" command was run in the extracted folder to complete its installation. The GridPortlets package was extracted in the $GRIDSPHERE_HOME/projects folder. All that was left to have a functioning grid portal was to run the command "ant install" from the $GRIDSPHERE_HOME/projects/gridportlets directory and to start the Tomcat server. Starting the Tomcat server was done by running the script startup.sh in the $CATALINA_HOME/bin directory. If the Tomcat server had been running, it would have been restarted by running the shutdown.sh script in the same folder, followed by the startup.sh script. A web browser was then pointed to the URL http://uagridportal.csce.uark.edu:8080/gridsphere where the portal is deployed. The default password for the root account was used to log into the portal.

The basic configuration of the GridPortlets involved editing a file located in the $GRIDSPHERE_HOME/projects/gridportlets/webapps/WEB-INF directory. In order for local resources to be added to the portal, the Resources.xml file was modified to include them. Samples of GRAM, MyProxy, and GridFTP resources available in this file were used as examples. The file can also be edited through the portal. If changes

are made through the portal they will be lost if a reinstall of the portal is done without first backing up the file. The MyProxy server requires credentials to use, so a proxy file was specified in its entry in the Resources.xml file. Once the configurations were made, the %ant deploy command was executed from the $GRIDSPHERE_HOME/projects/gridportlets directory. This updated the files deployed to the Tomcat server. Tomcat was restarted once again.

5.2 OGCE portlets

The OGCE set of portlets supports a few more options in addition to the same functionality that GridPortlets provides. The package was extracted in the same $GRIDSPHERE_HOME/projects directory as the GridPortlets. Inside the ogce2 subdirectory that this creates, the grid.installation.properties file was updated to include the resources that needed to be added. The project.properties file was also updated to contain the proper Tomcat directory, deployed GridSphere directory in Tomcat, and the IP address of the portal machine. It should be noted here that the default OGCE settings require Tomcat to be installed in the same account to which the portal is being built and deployed. This can be changed by further editing of these same files.

The “maven gridSphereDeployAll” command was executed in the same directory that these files are located. This deployed all of the portlets to the Tomcat server. Tomcat was then restarted again. To use the new portlets, an administrator account was used to create a new group using the provided interface under the Administration tab and 'Groups' pane. After clicking 'Create new group', the administrator gave the group the name OGCE and a description, and then selected all of the OGCE portlets that were just deployed. The layout was then given a name, and 'Create Template Layout' was selected. An OGCE tab containing all of the portlets that were added appeared next to the GridSphere tab. If this had not occurred automatically, the user would have been added to the new group manually.

5.3 GridPort 3 and GridPort 4

The final set of portlets installed was developed by GridPort. A demonstration portal is provided as well as individual downloads for each portlet. The demo portal for GridPort 3 was extracted in the $GRIDSPHERE_HOME/projects folder as were the first two packages. This is a variance from the GridPort instructions. Several installation issues arose with the individual portlets, so it was simpler to install the demo portal. GridPort by default deploys the GridSphere framework and its own portlets. Since GridSphere had already been deployed, this was undesirable. Instead of running the “maven deploy” command the “maven deploy-portlets” command was chosen to only install the portlets. A problem still arose with the installation of the gpcommon portlet, so the prerequisite to install it was removed in the maven.xml file. Once the “maven deploy-portlets” command was run, all of the portlets defined in the project.properties file on the first line are installed. The installation of the Proxy Manager portlet was the same as the OGCE portlet, so a new portlet was not listed.

Tomcat was again restarted, and a new group was created to include these new portlets.

The release of GridPort 4 fixed many of the issues in the previous version. The latest release was extracted in the portal user's home directory, and the "maven gp:deploy" command deployed the demonstration portal to Tomcat. If GridSphere had not yet been installed, the "maven gp:install" command would have installed and deployed it first and then completed the GridPort install.

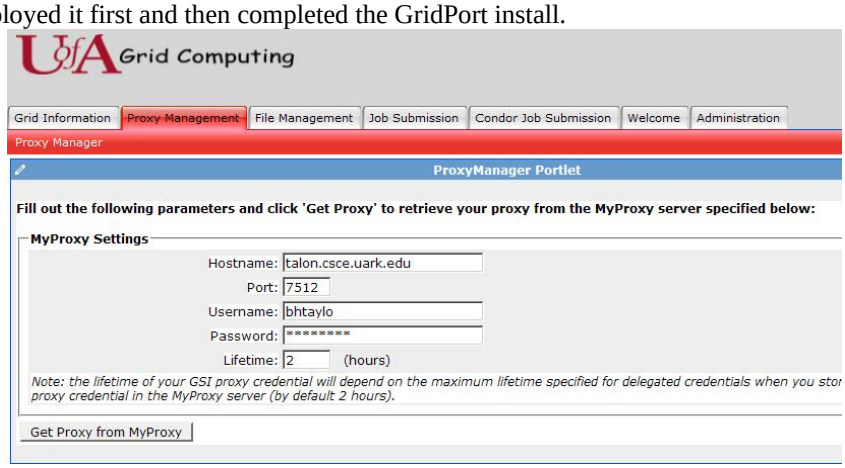

Fig. 10. GridPort ProxyManager Portlet

Figure 10 shows the Proxy Manager Portlet. Figure 11 shows the File Management Portal. Files can be uploaded or downloaded through the browser, or moved between machines that are registered with the portal. Figure 12 shows the Job Submission Portal and the fields available. The resource to which the job is submitted must be entered as a GRAM resource in a configuration file in order to show up in the Host list. A GPIR Browser Portlet was also configured but not shown here.

6 Conclusions

Since grids are by nature complex with varying architectures, software stacks, ownerships, and administrators, it is difficult to foresee an out-of-the-box grid solution that can simply be switched on. The installation and administration of these tools will always maintain some degree of complexity since no two grids are likely to be identical without some planning. However, grid tools are becoming widely used, and the support supplied by documentation and the user community is adequate for most cases. The increasing amount of effort devoted to research in the area of grid computing and development of tools to facilitate it have produced amazing results in the latest generation of grid tools.

With many of the major functionality holes filled and the overall stability of grid tools such as Condor, Globus Toolkit, and Shibboleth, many administrators are finding grids to be a reasonable part of their architecture. The Globus Toolkit now offers a wide variety of tools including credential management with SimpleCA and

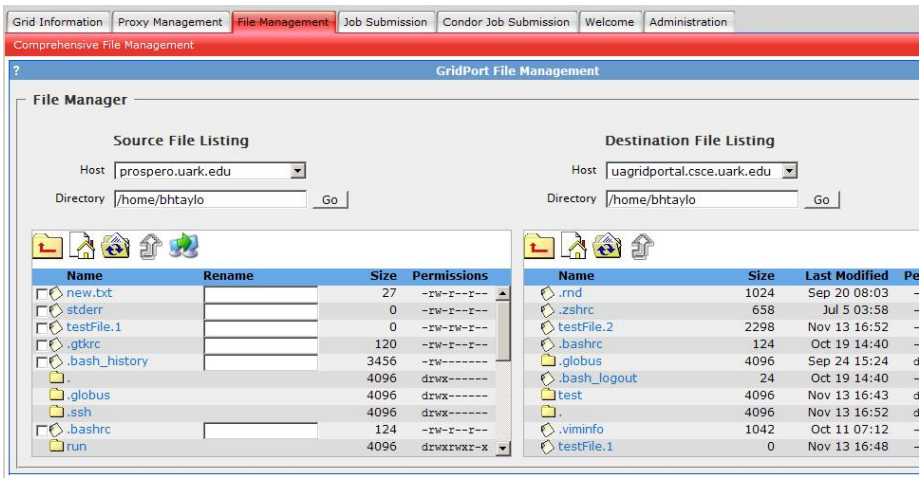

Fig. 11. GridPort File Management Portlet

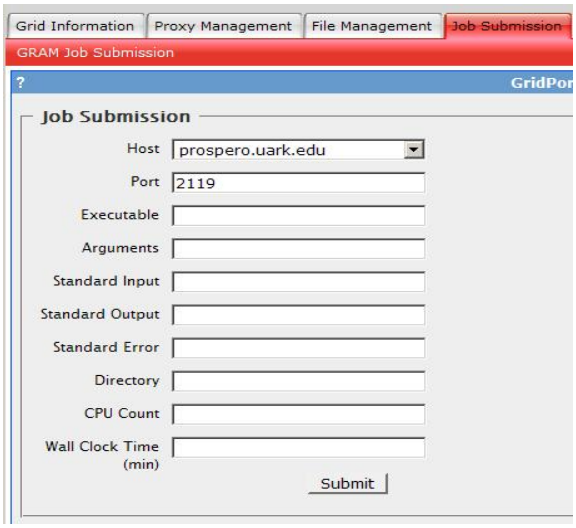

Fig. 12. GridPort GRAM Job Submission Portlet

MyProxy, file management with GridFTP, job submission and management with GRAM, resource monitoring with WSRF tools, and more all within a single installation. The volume of documentation for the installation and management of Globus is growing, and many administrators are finding it fairly understandable. Condor is also an attractive solution for creating a sizable grid from idle machines already in existence on a network. The addition of Condor-C has improved the prospect of collaboration between institutions with Condor pools by greatly reducing network congestion problems. Condor-G has begun unifying Condor and Globus grids in order to increase collaboration and computing power. GridShib has done the same with Shibboleth and Globus, and their approach of fine-grain security is appealing to campuses

since faculty and students can be granted access by updating the attributes in an Identity Provider. With the functionality, stability, and availability of current tools, building and maintaining working grids has become a viable option for many organizations.

Web portals have been around for quite some time, but the idea of a framework that allows content to be plugged-in is relatively new. Many of the well known portal frameworks have moved toward a pluggable framework from the original design of integrated content and subsequently redesigned their framework with this in mind. This move left the design of the module to be plugged-in up to the developers of each portal and led to several different implementations. The recent creation of the JSR-168 standard for Java portlets has greatly simplified the process of portlet development since there is no longer the concern of which portals will support the portlet. As the portals integrate support for this standard, portlet developers can write quality portlets and determine which framework best supports their needs. If, at a later time, this decision is no longer valid and another framework is more suitable, the change to a new framework will not be devastating in terms of a complete redesign.

Also new is the concept of accessing grid resources from portal interfaces. Some work has already been done in this area by groups such as OGCE, GridPort, and GridSphere, but the field is still in the process of developing. As grid technology improves and more grids are developed, this field will become more important to simply interfacing with the grid for users. At the moment, many grid tools are still in development with drastic implementation changes occurring at each release. This can make development of the corresponding portal interfaces an enormous challenge since each new interface can mean a total redesign of a previously working tool. Developers of these interfaces have a significant amount of work to do after each release, and this will continue as long as significant updates continue to the underlying grid tools.

University campuses generally have several computing resources. These can range from a supercomputer down to a simple lab of networks personal computers, but resources are generally available for the research that takes place on campus. The emergence of grid computing as a mechanism to unite all of these resources and harness their computing power effectively has brought many universities into the grid computing field. The ability of the Globus Toolkit to provide the necessary glue to pull the resources together and of Condor to utilize idle time on machines in the general access labs on campus that are not in use at all times gives universities new high-performance computing resources without any additional cost. Collaboration with other institutions, educational or otherwise, is now available at a scale previously only possible with significant financial investment. While many universities are undoubtedly testing and adopting these new tools, relatively few have advertised working portal interfaces to these resources for students or faculty, but the field is still new. As the technology improves and universities make their grids available to students, faculty, and researchers, usage of grid portals will increase accordingly.

Many of the computing resources at the University of Arkansas either had no grid tools installed or had grid tools installed on them, but these had not been used together in the sense of grid computing. The main goal of this research was to pull these resources together so that they could be better utilized. The installation of the Globus Toolkit on several resources including security components, Prospero, UA-

GridPortal, and others allows this connection in a nonintrusive way. Since the installation of Globus is done using a separate user and interfaces to the local scheduler, the resource can continue to be used as it always has been as well as through the interfaces that Globus provides. This high level interface layer that Globus provides has proven that it is possible to install the toolkit on virtually every resource on campus without ill effect if the administrator wishes to add the resource to the campus grid. This installation in no way removes an administrator's ability to control who has access to the resource since access control is ultimately controlled with the grid-mapfile.

The command-line interface to these tools has been shown to work without incident. Many tasks involved in effective use of the grid were discussed with example executions for most. Among these were credential management by requesting credentials with grid-cert-request, creating proxy credentials with grid-proxy-init, and using the MyProxy Server with myproxy-init and myproxy-logon. File management was shown through GridFTP using the globus-url-copy command for standard and third-party transfers. Job submission to the GRAM interface using the globusrun-ws command was also demonstrated. The basic functionality that these commands provide make the grid interface a useful tool for utilizing many of the resources on campus.

While the command-line interface works well, the portal interface to these tools does not quite reach the same level. Many portal frameworks were discussed, and the installation of the GridSphere framework with the many components it requires works quite well. The web interface to the portal also works flawlessly. Deployment of the GridPort, GridPortlets, and OGCE portlets were also deployed with the varying degrees of success. With the final deployment of GridSphere and GridPort 4, proxy management, file management, and the resource monitoring services work without a problem. Although job submission to Prospero was successful from the command line, it was never achieved through any of the portlet packages. As the portlet developers work out the final details of these portlets, these issues will be resolved, and the University of Arkansas will have a fully functional portal interface to its campus grid.

Several campus resources have been discussed, but many more resources are still available on campus. Integrating these resources into the campus grid will help to further unite the campus' computing power and give researchers a stronger system with which to work. Adding these resources to the portal and eventually granting access to students will give the university a more robust environment for teaching students about grid concepts and facilitating this research. Upgrades should be made to the grid portlets when made available to help solve the issues with job submission. Another area of improvement is to integrate the portal authentication with the campus LDAP server. The portal was designed with this in mind, but no implementation is currently available. As soon as the issues related to job submission have been resolved, the portal will be ready for testing. This could be done in a classroom setting to help discover any remaining issues or needs that have not been met. After a successful testing period, the campus’ Computing Services department will need to be contacted about implementing the solution on a production quality machine. A campus-wide Certificate Authority and MyProxy server will also be needed.

Bibliography

[1] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," presented at IFIP International Conference on Network and Parallel Computing, 2005.
[2] The Globus Security Teams, Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective, The Globus Alliance, 2005.
[3] Platform Computing, October 2005, http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF.
[4] Sun N1 Grid Engine 6, October 2005, http://www.sun.com/software/gridware/index.xml.
[5] OpenPBS, October 2005, http://www.openpbs.org.
[6] SURAgrid, October 2005, http://www1.sura.org/SURAgrid.html.
[7] P. M. Papadopoulos, M. J. Katz, and G. Bruno, "NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters," presented at Cluster 2001: IEEE International Conference on Cluster Computing, Newport Beach, California, 2001.
[8] T. Naughton and S. L. Scott, "The Penguin in the Pail - OSCAR Cluster Installation Tool," presented at 6th World Multi Conference on Systemic, Cybernetics and Informatics, Orlando, Florida, 2002.
[9] Java 2 Platform, Standard Edition (J2SE), October, 2005, http://java.sun.com/j2se.
[10] Apache, October 2005, http://tomcat.apache.org.
[11] GNU Project, October 2005, http://www.gnu.org/software/make.
[12] MySQL, October 2005, http://www.mysql.com.
[13] PostGreSQL, October 2005, http://www.postgresql.org.
[14] The GridPort Toolkit, October 2005, http://gridport.net.
[15] Altair PBS Professional, October 2005, http://www.altair.com/software/pbspro.htm.
[16] IBM LoadLeveler, October 2005, http://www-03.ibm.com/servers/eserver/clusters/software/loadleveler.html.
[17] Computing Center Software, October 2005, http://wwwcs.upb.de/pc2/projects/ccs.
[18] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," Concurrency and Computation: Practice and Experience, vol. 17, pp. 323-356, February 2004.
[19] The Globus Toolkit, October 2005, http://globus.org/toolkit/about.html.
[20] Grid Physics Network, October 2005, http://www.griphyn.org.
[21] Open Science Grid, October 2005, http://www.opensciencegrid.org.
[22] TeraGrid, October 2005, http://www.teragrid.org.
[23] NEESgrid, October 2005, http://www.neesgrid.org.
[24] R. Housely, W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," RFC 2459, 1999.
[25] V. Welch, T. Barton, K. Keahey, and F. Siebenlist, "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration," presented at 4th Annual PKI R&D Workshop, Gaithersburg, MD, 2005.
[26] M. Erdos and S. Cantor, "Shibboleth-Architecture DRAFT v05," Internet2/MACE Draft Specification, 2002.
[27] Condor-C, October 2005, http://cs.wisc.edu/condor/manual/v6.7.8/5_4Condor_C.html.
[28] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid," in F. Berman, G. Fox, and A. J. G. Hey, editors, Grid Computing: Making the Global Infrastructure a Reality, John Wiley, 2003, pp. 299-336.
[29] A. Abdelnur and S. Hepper, "Java Portlet Specification Version 1.0," JSR-168, 2003.
[30] WebSphere, October 2005, http://www-306.ibm.com/software/websphere.
[31] Jetspeed-2, October 2005, http://portals.apache.org/jetspeed-2.
[32] J. Novotny, M. Russell, and O. Wehrens, "GridSphere: A Portal Framework for Building Collaborations," in Proceedings of 1st International Workshop on Middleware in Grid Computing, 2003.

[33] uPortal, October 2005, http://www.uportal.org.
[34] Grid Engine Portal, October 2005, http://gridengine.sunsource.net/gep/GEP_Intro.html.
[35] GridLab: A Grid Application Toolkit and Testbed, October 2005, http://gridlab.org.
[36] Java in Administration Special Interest Group, October 2005, http://www.ja-sig.org.
[37] Pluto, October 2005, http://portals.apache.org/pluto.
[38] Sun ONE Portal Server, November 2005, http://docs.sun.com/app/docs/coll/S1_PortalServer_61.
[39] Open Grid Computing Environment, October 2005, http://www.collab-ogce.org/nmi/index.html.
[40] GridPort Information Repository, November 2005, http://gridport.net/services/gpir.
[41] J. Novotny, "The Grid Portal Development Kit," Concurrency and Computation: Practice and Experience, vol. 14, pp. 1129-1144, February 2004. http://gridport.net.
[42] B. Taylor, “Architectural Tradeoffs for Unifying Campus Grid Resources,” M.S. Thesis, University of Arkansas, December, 2005.