



**GridPP**

UK Computing for Particle Physics

# **Technology Review for GridPP Portal Project**

GIDON MOONT

Document History:

First draft: 30 November 2004

High Energy Physics Group  
The Blakett Laboratory  
Imperial College London  
South Kensington Campus  
Prince Consort Road, LONDON SW7 2BW  
Tel: +44 (0)207 594 7810  
FAX: +44 (0)207 823 8830

**Contents**

<b>1 Introduction</b>	<b>3</b>
<b>2 Grids</b>	<b>4</b>
<b>3 Security</b>	<b>4</b>
<b>4 Portals</b>	<b>5</b>
<b>5 Web Servers</b>	<b>7</b>
<b>6 Web Services</b>	<b>8</b>
<b>7 Conclusions</b>	<b>9</b>

## 1 Introduction

The GridPP Portal project aims to provide easy access to the computational power provided by Grid computing. To do this, the portal will provide an interface to the LHC Computing Grid (LCG), which is synonymous with the Enabling Grids for E-science in Europe (EGEE) Grid. If the project is to remain relevant for more than the lifetime of this post, then the portal needs to have the flexibility to interface to both future versions of the LCG, and to other Grids. The technologies used therefore need to be appropriate, flexible, and sufficiently stable into the future.

The document "Use Cases and Requirements for GridPP Portal Project" lists the functionalities that the GridPP Portal should provide to those who will be using it. These are concentrated on the provision of Grid production services to enable large amounts of calculations to be done by small HEP experiments. The portal should also be able to be used to demonstrate the computational power of the LCG.

The technologies reviewed here are being examined with a view to how well they can provide resources and frameworks to the above needs.

## 2 Grids

A Grid is a distributed computing environment. Like all generalised concepts, there are many ways in which to implement it.

Historically, Grids have been based on the Globus Toolkits (GT). The LCG is based on the European Data Grid (EDG), which used GT2.x. The current LCG incorporates both EDG and GT2.4 functionalities. However, the LCG is planning to move to using gLite in the near future. Also, in a similar time frame, there will be a release of GT4. Both of these Grid technologies will represent significant shifts in the way that Grids work.

This means that the GridPP Portal, in order to have any chance of an extended lifetime, must be flexible enough to be used on Grids based on either gLite or GT4. However, for the portal to be useful in the near future, it has to be able to interface with the existing LCG.

The most accessible method to the LCG is by the LCG-2 Command Line Interface Tools. There are also some C++ APIs and also some Java APIs. However, the Java APIs are merely wrappers around the C++ APIs, and therefore have the same problem of not being cross-platform portable.

Since the LCG is partly based on GT2.4, the portal can utilise the projects which provide GT2.4 compatible software. There are various Commodity Grid (CoG) Kits including Java CoG, Python CoG, and Perl CoG. However, none of these provide the functionality of the EDG commands that are also an integral part of the LCG. Most importantly, this excludes the Replica Manager commands which are vital to storing and retrieving data on the LCG.

## 3 Security

The Grid is a valuable resource, and any access point to it needs to be secure.

Any user of the Grid is required to have an X.509 certificate, signed by a relevant Certificate Authority (CA). This means that any person wishing to use a function of the GridPP Portal that uses the Grid, would be required to have such a certificate. There may be other people who would not be using these functions, and so may not have a certificate.

If all the functions of the GridPP Portal are Grid functions, then there is a

strong argument for the method of login to be based on the users CA signed X.509 certificate. The requirements derived for the GridPP Portal in the document "Use Cases and Requirements for GridPP Portal Project" indicate this to be the case.

Such a login method would mean that the job of administering user accounts for the portal would effectively be managed by the CA. This would not exclude the portal from providing functions that were accessible without logging in by any method, such as general Grid monitoring information and access to documents.

The technologies which provide this functionality for a web server include the Apache module GridSite, as well as various integrations of Open-SSL with Apache and Tomcat. It is also possible to transfer the security layer provided by GridSite to Tomcat via the mod\_jk connector, where Tomcat acts as a worker for the Apache server, rather than acting as its own server. The emerging OGSA aims to provide Web Services using X.509 authentication.

When using a Grid, there are issues connected to how various components of the Grid will have access to a user's credentials. It is not secure to pass around a user's certificate, so there are mechanisms for creating proxy certificates. These can have the same permissions as a user's CA signed certificate, or a subset of them, yet have a limited lifetime. This lifetime is typically a few hours.

This limited lifetime is not an ideal solution. Problems occur if a job is still running when a proxy certificate expires, as any further credential check will fail. One solution to this is to give the proxies longer lifetimes. Another is to use the MyProxy service which enables components on a Grid that need to check credentials to check a central repository, should the available proxy have expired.

## 4 Portals

This post is to provide a working GridPP Portal. This portal could either be accessible through a web browser, or through a thin client, or provide components so that both were possible.

There are already several portals in existence funded by various Grid projects, and it would be wasteful to duplicate their work.

The "Use Cases and Requirements for GridPP Portal Project" document was written in order to determine what was wanted from the GridPP Portal. The required underlying capabilities are certificate handling, job submission/retrieval/logging,

and data/file handling at a high level. Unfortunately, no existing portal, or portal toolkit, provides these functionalities.

The one possible exception is GENIUS, which is not entirely open source. It is based on CGI components, and is interfaced by a web browser only.

Many of the portals in existence have concentrated on portal technology, rather than Grid usage. These portals provide useful tools for collaborative on-line, web browser interfaced, environments, but do not currently provide implemented Grid functionalities. Even where projects state aims of providing working Grid capabilities, these are aimed at GT2.4 or later Globus architectures, not at the LCG.

A lot of recent projects have concentrated on collaborative portals based around Java technologies, specifically portals that are made up of "portlets". These are self contained components which sit in an appropriate container, the whole making up a portal. An emerging JSR-168 standard is being widely adopted, allowing portlets designed to be a component of one portal to be readily reused in another. The container is normally itself a web application running on an Tomcat web server.

An example of this is GridSphere, a part of the GridLab project. There is development of some GT2.4 functionalities, but these are not yet stable. Even in this case, there is no intention of providing LCG components, and neither is there any current time frame of when components for gLite or GT4 would be developed. Other projects have implemented prototype Grid portals with functions such as retrieving a proxy from a MyProxy server and basic job submission. However, some of the security models are non-existent, such as for the Open Grid Computing Environment (OGCE) and CrossGrid, both of which want a user to provide a passphrase for a MyProxy server over http.

Another technology that is being incorporated into portals is the idea of remote web services. This involves the portal providing a thin interface to a service that is possibly shared by many access points. The interface can be through a web browser or potentially through a thin client, and the service can be provided from a number of platforms. The transport is normally based on Simple Object Access Protocol (SOAP), and the services can be provided from specific web service containers such as Axis, as well as using SOAP::Lite through Perl/CGI. The CCLRC e-Science projects of HPCPortal and DataPortal have investigated web services.

Although there are many projects investigating and prototyping Grid portals,

overall there is currently no available toolkit or template which provides a method of creating a portal for the LCG.

## 5 Web Servers

The server technology of choice, which has the added advantage of being free is Apache. Although SUN, IBM and Microsoft and other companies provide web servers, none of them are free and none of them have the level of flexibility and support offered by Apache. The only possibility other than Apache is the SUN server, due to the integration of Java technologies.

### Java

Java technology has become integrated into web technology through several different channels of Java development. Java Server Pages (JSP) is the most prominent example, though JSR-168 standard Java Portlets and Java Web Start (JWS) are also relevant to this review.

JSP allows for web pages that include Java functionality. Although the SUN server provides the ability to serve JSP pages inherently, Apache also provides software additional to the core httpd server to allow for JSP content, most prominently via the Tomcat project.

Java Portlets are bundles of files that provide a pluggable unit to be inserted into a container designed to accept them, i.e. a portal. There are several projects, all using Tomcat as the underlying Java interface, that implement portals that can contain these portlets. These include PLUTO, uPortal and JetSpeed. Other projects build upon these to provide fully functional portals with ready made components, such as GridSphere, CHEF and SAKAI. In order to encourage the re-usability of portlets, a standard now exists as to how these portlets should behave and how they should connect to a portal. This JSR-168 standard is now being adopted by all portal projects, though many of them are doing so through connectors that translate between standards.

Applets are of interest, but cannot interact with the file system on the users machine.

## Other components of Web Servers

The Common Gateway Interface (CGI) is a universal mechanism to allow programs to be accessed by a webserver. These programs then return data to the server, often as a web page to be sent back to a user. These programs can be written in any language, though Perl is very popular due to its ability to integrate well into the Apache software.

All web pages are written in a form of Hypertext Markup Language (HTML). It is important for compatibility with multiple browsers that pages conform to the one of the agreed WC3 definitions.

Cascading Style Sheets (CSS) are the method by which HTML can be given a "look and feel". This has advantages of easily enabling a global style to a site, but has a problem in that CSS suffers from poor implementation on many browsers, including the most popular. CSS1 is rendered correctly in most cases on more recent versions of the browsers, but CSS2 is still flaky.

Javascript is not a Java technology. It is useful in syntax checking forms in HTML, and can be used in conjunction with the Document Object Model (DOM) to dynamically change the HTML in a web page based on a user action.

Pre Hypertext Processing (PHP) allows for a web page written in HTML to have embedded within it some non-static components.

There are many other technologies available that can be used as part of a web server, such as Microsoft's Active Server Pages (ASP) and Macromedia's Shockwave/Flash. However, these are not relevant technologies for this report.

## 6 Web Services

To enable sharing of information between software components, the eXtensible Markup Language (XML) is a method of defining data in a document in a hierarchical way that can be understood and manipulated by different software components. XML is not so much a technology in its own, as an integral part of many other technologies which want to be able to share information.

Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information in a decentralised, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules

for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

Web Services Description Language (WSDL) is an XML format used to describe web services such as SOAP. This has the very useful aspect of allowing a web service to be described in a machine readable form. As a result, software such as Axis and Perl::Lite can read in a WSDL file, and immediately provide methods to accessing the web services described.

OASIS Web Services for Remote Portlets (WSRP) is similar to WSDL but specific that it provides services that can be easily integrated into a JSR-168 compliant portlet. This is still very much at a development stage.

Axis is an Apache project that provides web services via SOAP. It acts as a standalone server, but can also work in conjunction with a Java server such as Tomcat. It is written in Java and includes tools to convert services described by a WSDL into APIs for a client interface.

## 7 Conclusions

After reviewing the above technologies, and trying out the more promising candidates for their ease of use, the following conclusions have been reached.

In order to allow for rapid development of the portal, while allowing flexibility for the future, the portal will be designed as two separate components. One will provide a user interface, the other will provide methods to use the LCG Grid. The two will communicate through SOAP services, described by WSDL files.

The user interface will initially be written as a website, accessible to the end user through any standard browser. The pages will be written in XHTML and will be tested on at least Mozilla and Internet Explorer browsers. The website will run Gridsite, so allowing the use of Gridsite's security and powerful file management tools. Functionalities will be provided using CGI Perl, and use Perl::Lite to access the SOAP services.

Future user interfaces could include websites based on GridSphere. However, at the moment the model of reusable components and additional management tools available in a collaborative JSR-168 development environment does not represent the reality of prototype development and non-adherence to standards.

The other component, providing methods to use the LCG Grid will be written

as SOAP services in Perl using SOAP::Lite. They will be described by WSDL files, so allowing any "front end" to connect to them.

Although Axis provides some very useful powerful tools to serve SOAP web services, the current lack of an API to the LCG tools means that ultimately a service has to execute a system shell command. This is a lot easier to do in Perl than Java. It is hoped that such APIs will exist in future versions of the LCG, so allowing a move to Axis or equivalent Java based SOAP services.

The only additional piece of technology to be used is Java Web Start, with the Java CoG Kit, to provide an easy method for a user to give the portal a proxy certificate. This avoids the user having to have any access to a machine that can initiate proxy certificates.