



Development of standards-based grid portals, Part 1: Review of grid portals

Level: Intermediate

Xiaobo Yang (x.yang@dl.ac.uk), Software Developer, Consultant

Xiao Dong Wang (x.d.wang@dl.ac.uk), Senior Software Developer, Consultant

Robert Allan (r.j.allan@dl.ac.uk), Group Leader, Consultant

20 Mar 2007

Built on top of grid middleware, grid portals act as gateways to the grid because they smooth the learning curve of using grid. In the first of this three-part "[Development of standards-based grid portals](#)" series, we give an overview of grid portals, focusing on today's standards-based (JSR 168 and WSRP 1.0) second-generation grid portals. We provide examples in Part 2 that demonstrate how grid portals are developed using portlet technology. And in Part 3, we discuss the application of WSRP and the future of grid portals.

A short history lesson on the Globus Toolkit and portals

With the emergence of grid in the 1990s, distributed resources like supercomputers and experimental instruments combined to explore ever-larger problems that previously were impossible to tackle. The concept of virtual organisation (VO) has been proposed to describe the on-demand community, which can be constructed dynamically. Now such a grid system is widely adopted in research institutions and is gradually being deployed in industry. Grid systems can be divided according to their scales. A small one (campus grid) might only contain resources from several research groups, while a large one may integrate resources across an entire country, such as the U.K. National Grid Service (NGS), or even around the world, like the Enabling Grids for E-science (EGEE) project.

Before moving on to grid portals, we'll take a brief look at the Globus Toolkit (GT) for reference. GT, the de-facto standard of grid technology, has been widely deployed since V2.x. It allow you to build grid systems and applications. GT2 includes a set of scripts for executing tasks like proxy credential creation, remote job submission, and GridFTP-based file transfer. Many grid systems based on GT2 have been built around the world. For example, NGS uses GT2 to provide production services to U.K. researchers in multiple disciplines.

In 2002, the Global Grid Forum (GGF) proposed a new architecture, the Open Grid Services Architecture (OGSA), to define a common standard and open architecture for grid-based applications. OGSA extends the concept of Web services so that grid services are defined. The key attribute of a grid service is that it's *stateful*, while a traditional Web service is *stateless*.

GT3 is an OGSA implementation developed by the Globus Alliance. Although GT3 brings a new architecture and tries to use benefits of the industrial Web service standards, it has not been widely deployed due to its complexity. Nevertheless, some countries have developed GT3-based projects. For example, the ShanghaiGrid project aims to connect some supercomputers so that massive storage and computing power can be shared. The project's goal is to use the power of this grid system to provide real-time traffic-jam control and guidance in Shanghai.

The Globus Alliance soon responded to the issues raised by GT3 and released GT4, which is based on the WS-Resource Framework (WSRF). The WSRF, as originally proposed by the Globus Alliance and IBM®, is now an Organization for the Advancement of Structured Information Standards (OASIS) standard. WSRF standardises how to make Web services stateful. In fact, the Web services and the grid communities are merging.

As GT has been upgraded, projects adopting it have required upgrading accordingly. No matter which version is adopted, the resulting application software must remain user-friendly. Researchers aren't willing to change their habits, though they understand how grid can benefit them. To smooth the learning curve of using grid, two options are available for end users: 1) using a desktop client, and 2) using a Web-based grid portal. The first option is a traditional approach for software distribution. Based on grid middleware, you can develop client tools that provide

powerful functionalities for executing grid-related tasks. These tools normally require a set of ports be open for completing tasks like file transfer using GridFTP. The approach works well within particular institutions (a campus, for example), but requires a lot of administrative work and is not as flexible as the second approach. The Web is an ideal platform for distributing services, and nearly all desktops are equipped with browsers (as are devices such as mobile phones and PDAs). This makes a Web-based grid portal a natural choice for providing the same services as a desktop client. Best of all, grid portals don't require client software installation -- only a Web browser. Pervasive access is possible because users are no longer limited to computers with grid middleware. We hope that an increasing number of users will be attracted by grid systems and applications accessed transparently through Web portals.

First-generation grid portals

First-generation grid portals were typically built with the help of either Perl-based GridPort Toolkit V2 or Java™ technology-based Grid Portal Development Kit (GSDK). Both GridPort and GSDK wrapped commonly used grid functionalities so that a set of higher-level programming APIs and tools were provided. In some other projects, some commonly used tasks like proxy management and job submission were built in again and again, but there was no way to reuse these tools. In practice, tools were developed according to each project's own requirements without considering the reusability of code. The adoption of portal technology changed this and showed how common Web-based services could be developed.

Grid portal developers began to improve the reusability of their software by looking at programming frameworks. They looked to projects like Jetspeed-1, an Apache project that provided *portlets*. For example, the first release of the U.K. NGS portal and the first release of Open Grid Computing Environment (OGCE) were based on Comprehensive Collaborative Framework (CHEF). CHEF is the predecessor of Sakai, which was built on top of Jetspeed-1 to provide nonstandard portlet support. Jetspeed-1 showed a promising approach for building reusable Web components (for example, Java portlets), which could be shared by developers. Web Services for Remote Portlets (WSRP) 1.0 and Java Portlet Specification V1.0 (JSR 168) were both ratified by OASIS in 2003 to form the basis of today's second-generation grid portals.

Second-generation grid portals

The second-generation grid portals are standards-based. In particular, JSR 168 has been widely adopted by grid portal developers. JSR 168 and WSRP V1.0 aim to solve the interoperability issues in portal and portlet development and deployment. Furthermore, the second-generation grid portals are acting as service clients so that they fit in today's Service-Oriented Architecture (SOA). In this next section, we'll discuss the two portal and portlet standards, JSR 168 and WSRP, and the adoption of SOA in the portal world.

JSR 168 and WSRP 1.0

JSR 168 is a specification for standardising communications between portlets and a portlet container by defining a set of Java APIs. In addition to providing benefits like customisation, different modes and window states, this specification makes it possible for portlet developers to exchange Web components -- JSR 168-compliant portlets, for example. These portlets can be deployed in any JSR 168-compliant portlet container without modifying the source code.

Portals are now built around portlet containers, which manage the life cycle of their portlets. A typical portal framework usually provides functionalities like user account management and portlet deployment support. Hence, the burden on portal developers is greatly relieved, and developers can focus on portlet development, particularly the

Portal, portlet and portlet container from JSR 168

A *portal* is a Web-based application that provides personalised single sign-on content aggregation from different sources and hosts the presentation layer of Information Systems.

A *portlet* is a Java technology-based Web component, managed by a portlet container that processes requests and generates dynamic content.

A *portlet container* runs portlets and provides them with the required runtime environment.

the business-logic layer.

JSR 168, as its name implies, is a specification for Java portal developers. The question: How do you reuse Web contents (portlets) published using languages other than the Java programming language -- Perl and PHP, for example? The OASIS WSRP V1.0 specification was proposed to answer this question, among others.

Because WSRP separates portlets from portals, it introduces the concepts of *producer* and *consumer*. A producer is a service provider, and a consumer is a service client. Unlike the traditional data-oriented Web services, a WSRP producer provides formatted, rather than arbitrary, data to its consumers. A consumer is responsible for redirecting requests from end users to the corresponding producer service with information like input parameter name and value. The producer then handles the request and sends a response back to the consumer with markup fragments. The client-side portal can then retrieve these markup fragments from its consumer to render a full Web page in the user's browser.

WSRP realises a deploy-once, run-anywhere paradigm. Portlet vendors don't need to distribute their portlets to clients anymore. They now have full control of their portlets if they host them. This approach looks like the Java Web Start approach, which enables a software provider to always provide the latest version to its clients.

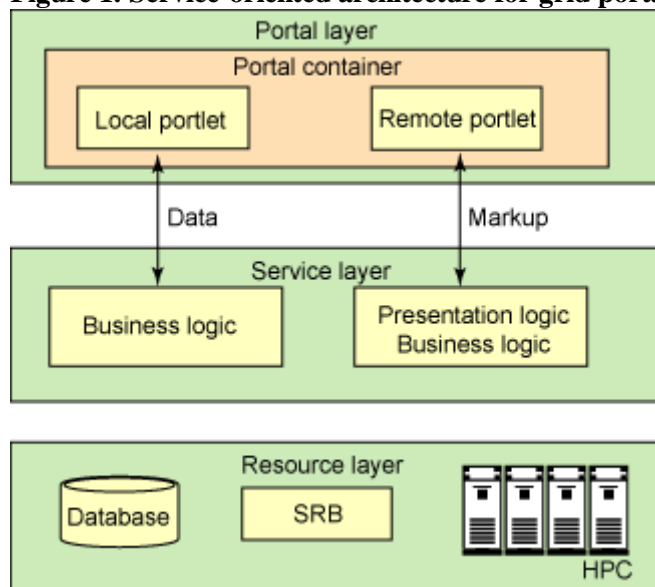
Because WSRP is based on Web services, it is automatically language- and platform-independent. In theory, consumers and producers should be able to be developed using languages like Perl and C#. Unfortunately, current WSRP implementation in open source portal frameworks is still immature. To the best of our knowledge, the only open source WSRP producer written in another language is done by the EDINA group using Perl. WSRP is supported by several leading IT companies, including IBM. One example of WSRP support on the Microsoft® .NET platform is from NetUnity, which aims at plugging .NET applications into Web portals.

JSR 286 and WSRP V2.0, successors of JSR 168 and WSRP V1.0, are scheduled to be released in 2007 and should provide improvements that will solve issues like inter-portlet communication and remote portlet life-cycle management.

Adoption of SOA in grid portals

Second-generation grid portals commonly act as front ends to services. Web services are today's de-facto standards for SOA implementation. Portlets are widely designed as Web services clients to consume remote services. With such an approach, portals fit well in SOA as a presentation layer.

Figure 1. Service-oriented architecture for grid portals



Whilst portals being treated as service clients does benefit portlet developers, with a clear separation of presentation and business logic, sometimes there is need to include presentation and business logic so that reuse of web components is possible. As shown in Figure 1 SOA is expressed as a three-tier architecture in which the three layers are *portal*, *service*, and *resource*, the same as the GridPort V4 architecture. Here the service layer has been extended

so that presentation logic may be included. Furthermore, the portal layer is no longer a container of service clients, but could itself be a service provider. This highlights presentation-oriented services proposed in WSRP and makes the portal layer able to provide Web components as services.

Grid portal examples

OGCE Release 2

OGCE is a U.S. National Science Foundation (NSF)-funded project that aims to build reusable portal components that can be integrated in a common portal container system. OGCE Release 1 was based on CHEF, which in turn was based on Jetspeed-1 with nonstandard portlet support as mentioned. Not surprisingly, the second release of OGCE adopts the JSR 168 standard and provides a set of grid portlets and services. Furthermore, OGCE Release 2 provides its own services to solve the interportlet communication issue, which is missing in the JSR 168 specification. Velocity-based portlet support is retained so Jetspeed-1 portlets can migrate to JSR 168-compliant containers. uPortal and GridSphere are supported by OGCE release 2. Based on our experience with portlet deployment, migrating OGCE portlets into JSR 168-compliant portal frameworks should not be difficult.

GridPort V4

Based on Perl, GridPort V4 is now Java technology-based and JSR 168-compliant. Similar to OGCE Release 2, a set of grid-related portlets such as the GridPort Information Repository (GPIR) browser portlet and Condor job submission portlet are provided to help users set up grid portals. GridPort V4 has a clear three-tier architecture that includes the portal layer, the service layer, and the resource layer. The portal layer interacts with end users and the service layer, which then handles interaction with the resource layer.

GridPort 4 architecture is considered a service-oriented architecture. Core grid functions -- GridFTP (for file transferring) and Grid Resource Allocation and Management service (WS-GRAM) (for job submission), for example -- are wrapped and exposed as Web services. These services can then be consumed by portlets residing in the portal layer.

NGS Portal release 2

A similar approach to OGCE development was adopted during the U.K. NGS Portal development. Release 2 of the NGS Portal is JSR 168-compliant, while Release 1 was based on CHEF. Originally, a set of grid portlets was developed for CHEF and migrated to a customised version of StringBeans, another open source portal framework. The MyProxy LoginModule was added to allow the NGS Portal to create user context automatically during logon with the grid proxy credentials retrieved from the NGS MyProxy server. Other key functions have been wrapped in reusable Java packages based on Java CoG V1.2.


During development, we realised the importance of adopting SOA, and we have investigated the application of J2EE technologies in portlet development. Enterprise JavaBeans (EJBs) have been written to implement the main business logic. The LDAP query bean has been converted to a Web service with the help of the EJB V2.1 specification, which allows the conversion of stateless session beans to Web services.


We have recently conducted extensive study on the importance of WSRP for portal development. Though the available implementation of the WSRP V1.0 specification is immature, it is practical in order to make WSRP producers and consumers function. We have developed a WSRP consumer for Sakai within which our grid portlets developed for the NGS Portal have been successfully consumed. WSRP provides a promising way to build federated portals. We discuss this trend for future portal development in the last part of this article.

Conclusion

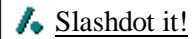
Second-generation grid portals are based on standards and fit well with SOA. Both JSR 168 and WSRP V1.0 aim to solve the interoperability issues in portal and portlet development and deployment. In this article, we've briefly covered a history of grid portals and standards. In Part 2, we show three examples to demonstrate LDAP query, grid proxy credential retrieval, and job submission to remote computing resources. This

Share this...

 [Digg this story](#)

 [Post to del.icio.us](#)

forms the basis of many generic grid portals. And in Part 3, we discuss the application of WSRP and the future of grid portals.



Resources

Learn

- The [JSR 168: Portlet Specification](#) was developed by Java Community Process (JCP) to standardise communications between a portlet and a portlet container.
- The [Web Services for Remote Portlets Specification](#) is a portlet specification developed by OASIS to standardise communications among portlet containers.
- [Open Grid Services Architecture \(OGSA\)](#) represents an evolution toward a grid system architecture based on Web services.
- [Introduction to Grid Portals](#) is a tutorial extending some of the information in this article.

Get products and technologies

- The [Globus Toolkit](#) is the de-facto standard implementation for enabling the grid.
- The [U.K. National Grid Service](#) provides coherent electronic access to computational and data grid resources for U.K. researchers.
- The [OGCE Portal Toolkit](#) aims to develop JSR 168-compatible portlets and Web services for building Web portals for science gateways.
- The [GridPort Toolkit](#) provides a set of portlet interfaces and services for rapid development of grid portals.
- The [Sakai Project](#) aims to provide a collaboration and a learning environment for education.

About the authors



Xiaobo Yang is a software developer working in the Grid Technology Group, CCLRC e-Science Centre, in the United Kingdom. He is interested in grid, collaborative virtual research environments, and various Web technologies, including grid portals, and Service-Oriented Architecture (SOA).



Xiao Dong Wang is a senior software developer in the Grid Technology Group, CCLRC e-Science Centre, in the United Kingdom. His interests include grid middleware design and application, portal user interfaces and portlet development, JSP, and JSF. He has more than six years' experience in Java programming and Web and grid service development.



Robert Allan leads the Grid Technology Group, CCLRC e-Science Centre, in the United Kingdom. His background is as a physicist and -- since the mid-1980s -- developer of high-performance computing applications using the latest technologies. He has managed several large HPC and e-science projects in the United Kingdom. He is particularly interested in making the grid widely usable.