Building Grid Services for User Portals

Dennis Gannon, Marcus Christie, Octav Chipara, Liang Fang, Matthew Farrellee, Gopi Kandaswamy, Wei Lu, Beth Plale, Aleksander Slominski, Anuraag Sarangi, Yogesh L. Simmhan
Department of Computer Science
Indiana University

1. Introduction

A Grid portal is a user's point of access to a Grid system [3,5,6,9]. It provides an environment where the user can access Grid resources and services, execute and monitor Grid applications, and collaborate with other users. A number of Grid portals exist or are currently under construction including the NEESgrid Portal [14], the Alliance Portal [22], the GENIUS Portal for the European Data Grid [19], and the IeSE (Integrated e-Science Environment) [20]. Also, several large collaborative efforts are underway to build portal construction toolkits based on OGSA Grid services, including the GridSphere project [23] in Germany and the NSF NMI Open Grid Computing Environment project in the US. Over the past nine months these projects have evolved a standard style for interacting with Grid and Web services as back-ends to the portal. The architecture that has developed is based on the idea that the portal server is a container for "user facing" Grid service clients which are designed according to the portlet model. A portlet is a server component that controls a small, user-configured window in a "pane" on the user's web browser. There is now a standard specification for portlets [21] and the portlet model is supported by most of the major vendors of portal software. It is also a standard supported by the Apache open source software foundation.

In this paper, we describe an example of our experience based on the Alliance Portal project and several Grid service examples. The first is a Grid service we call the XDirectory that provides each portal user with an indexed directory of his or her portal "context" information. Entries in this directory consist of XML documents that are annotated references to other objects such as Grid and Web service interfaces, service workflow documents, remote metadata directory entries and application logs. The portal users employ this service as private storage of remote Grid objects of interest along with their annotations. The portlet that interacts with this external Grid service is a custom-built interface that provides the user an intuitive look and feel and completely hides the fact that the interaction is with a remote service.

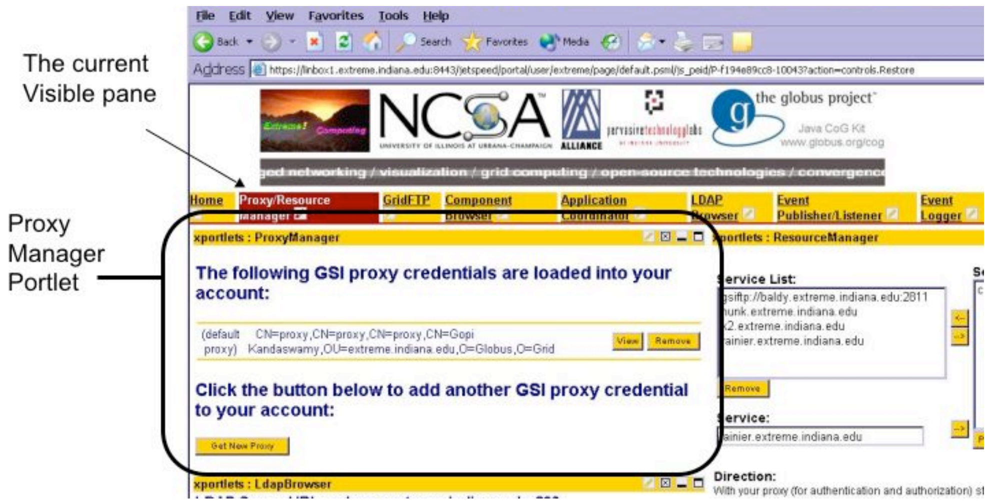
There is one difficulty with this model for portal construction. Grid and Web services are not, in general, designed to interact with humans. For the most part they are designed to interact with "service clients" and other services. The focus of this paper is to describe our experiences, both good and bad, with building services and the associated portlets that mediate the interaction between the services and the users.

Our principle conclusions drawn from these experiences are as follows. First, we have been able to build extensive and useful client interfaces to remote Grid services that can be rendered as a custom portlets. These remote service portlets make it possible for a user to interact with Grid services as a natural extension of the portal environment, without actually thinking of the services as remotely deployed. This also allows us a way to provide portal services that are consistent across multiple portal implementations and deployments. One such useful custom portlet can directly interrogate the required GridService port on any Grid service. This allows the user to interrogate any service in a standard way and this can be a powerful tool.

Second, we have found that custom interfaces are not needed for many Grid services. We have realized that it is possible to use a WSDL service description to automatically generate very simple interfaces to these services, which can then be used within the portal. However, if these services are rendered as Grid services, we believe that it is possible to accomplish much more. In particular, we feel it is possible to use the properties of a Grid service to allow us to provide powerful custom interfaces to the portal server automatically. At the end of this paper we will outline our plans for using OGSI [7,18] to accomplish this.

2. Designing a Portal for Grid Services

The portlet model for building portals is a widely accepted and used concept. It is the basis for many commercial portals[1]. It is also now a Java specification [21] which the Apache Jakarta Jetspeed Portal server will soon fully conform to. A portlet is a server-based software component that controls a small section of the user's browser display. Each portlet has a specific task, such as providing a special type of information to the user or managing some aspect of the user's Grid context as shown in Figure 1 below.



https://linbox1.extreme.indiana.edu:8443/jetspeed/portal/user/extreme/page/default.psml/js_peid/P-f194e89cc8-10043?action=controls.Restore

The following GSI proxy credentials are loaded into your account:

(default proxy) CN=proxy,CN=proxy,CN=proxy,CN=Gopi Kandaswamy,OU=extreme.indiana.edu,O=Globus,O=Grid

Click the button below to add another GSI proxy credential to your account:

gsiftp://baldy.extreme.indiana.edu:2811
hunk.extreme.indiana.edu
x2.extreme.indiana.edu
rainier.extreme.indiana.edu

rainier.extreme.indiana.edu

With your proxy (for authentication and authorization) st

Figure 1. Each portlet controls a small window in a pane on the user's browser. In this case the portlet is used by the user to see his or her current GSI proxy credentials.

A natural architecture for a Grid portal is one that has a specialized portlet interface dedicated to each of the main Grid services that are appropriate for user interaction. As shown in Figure 2, this is the architecture of our Alliance Grid Portal and it is the basis for the NSF Middleware Initiative Grid Computing Environment Portal, the NEESgrid Portal and several others.

The typical services for which our collaborations have provided custom portlet interfaces include:

- MyProxy – a server of Grid proxy credentials
- Directory Services – LDAP and the OGSI Index Service registry
- OGSA-DAI – an interactive portlet to the data access and integration service
- Various event logging and messaging services
- Group news and notes services
- AccessGrid control and viewing

[1] These include IBM WebSphere and the portal products from BEA systems and Oracle.

In the next section we describe one such portlet-service pair in detail.

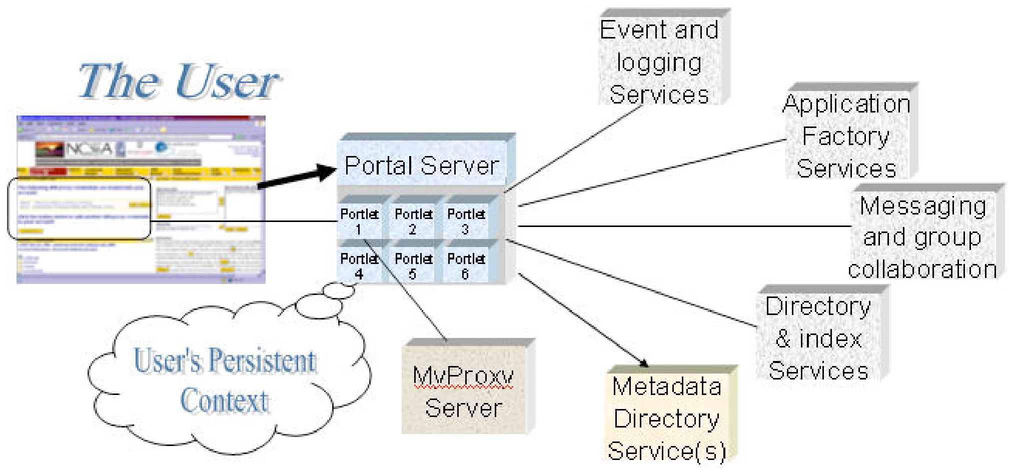

The User
Event and logging Services
Application Factory Services
Portal Server
Portlet 1
Portlet 2
Portlet 3
Portlet 4
Portlet 5
Portlet 6
Messaging and group collaboration
Directory & index Services
User's Persistent Context
MyProxy Server
Metadata Directory Service(s)

Figure 2. A portal architecture where each standard Grid service has a corresponding portlet that provides an interface between the service and the user. The portal server must also maintain a persistent "context" for each user consisting of the user's portal configuration preferences as well as references to remote objects.

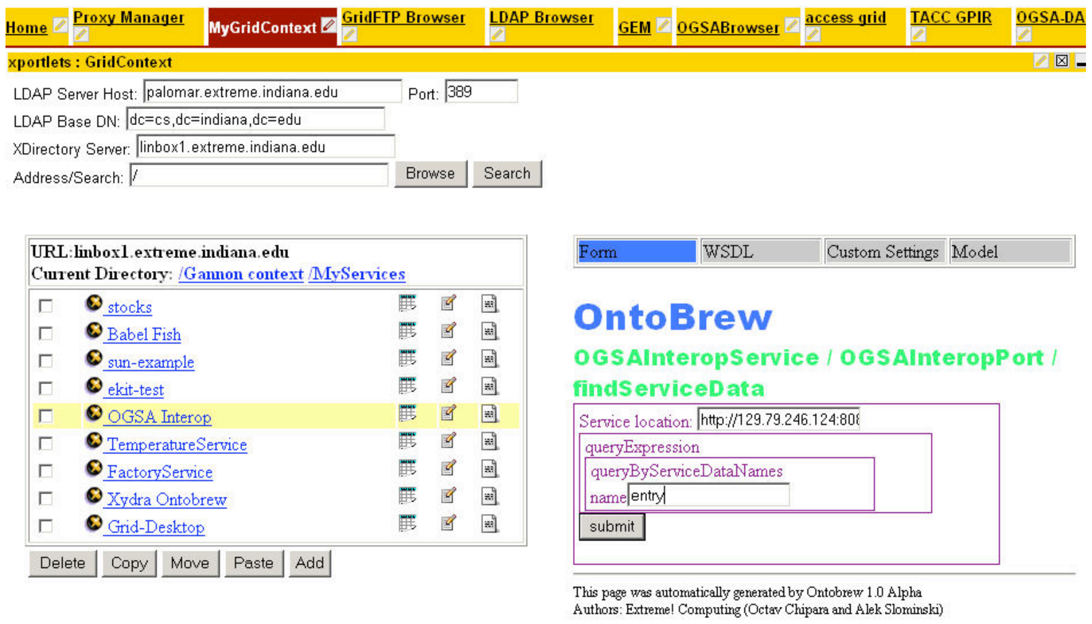
3. The XDirectory Service

One example of a portlet that serves as a front end to a Grid service is the XDirectory. This service provides the portal users a place to store annotated references to Grid objects such as Grid Service Handles (GSHs) to other services, Grid workflow scripts, and simple logs of application events. From the user's perspective, the XDirectory is organized like a hierarchical file system containing two types of nodes: directories and leaf nodes. Directory nodes contain leaf nodes and other directory nodes. Leaf nodes contain:

1. Metadata about the node – its owner, its parent in the directory tree, its creation date, etc.
2. Information about what the node refers to – for example, if this is node is a reference to a service of interest to the user, this element contains the GSH for the service and, if needed, a reference to a Handle Resolver.
3. Annotations – these are represented as small, user-editable HTML documents which may contain the user's notes about the item this node refers to, or it may also contain scripts which provide a means to interact with the service referenced by this node. For example, one can include forms or JavaScripts that allow direct interaction with certain remote services.

In terms of its implementation, the XDirectory is based on a simple relational database backend. The XDirectory service is also an XCAT Component and is compliant with the Common Component Architecture specification. A complete description of this particular hosting environment is described elsewhere [24]. It suffices to say here that the XDirectory has a standard GridService port, but it also has CCA "provides ports" implemented as standard Web services. When created, it can register a handle to the Component with a registry like LDAP or the OGSI Index Service registry. The handle uniquely identifies the Component. The string representation of the handle is the GSH of the XDirectory service while the WSDL document functions as the Grid Service Reference (GSR). The GridService port operates in the standard way and in addition, it has a service data element (SDE) called "ProvidesPortHandle" which contains a list of all the CCA Provides Ports (port-types or services) of the XDirectory service, along with the GSHs to these ports. The primary additional port implemented is the XDirectory interface, which includes the functions to add, remove and modify nodes in the database. It also contains a simple, keyword search based on indexed fields in the database.

The custom portlet for the XDirectory is shown in Figure 3 below.



xportlets : GridContext

LDAP Server Host: palomar.extreme.indiana.edu Port: 389
LDAP Base DN: dc=cs,dc=indiana,dc=edu
XDirectory Server: linbox1.extreme.indiana.edu
Address/Search: /

URL:linbox1.extreme.indiana.edu
Current Directory: /Gannon context /MyServices

stocks
Babel Fish
sun-example
ekit-test
OGSA Interop
TemperatureService
FactoryService
Xydra Ontobrew
Grid-Desktop

OntoBrew

OGSAInteropService / OGSAInteropPort / findServiceData

Service location: http://129.79.246.124:808
queryExpression
queryByServiceDataNames
name entry

This page was automatically generated by Ontobrew 1.0 Alpha
Authors: Extreme! Computing (Octav Chipara and Alek Slominski)

Figure 3. The XDirectory portlet shows the user's browsable directory tree on the left. In this case, the selected item is an entry corresponding to the service "OGSA Interop" *and the associated HTML annotation is shown on the right. In this case, the HTML annotation is actually a reference to a service called* OntoBrew, *which has been provided with the WSDL for the GT3.0 registry.* OntoBrew *has automatically generated a form to allow a user to query the service data elements of the registry.*

One service we frequently use from the portal is the application factory service. The design of this service is based on our experience with application developers who wish to provide authorized users a way to run an application that requires user input prior to execution, and gives the user a summary of the application behavior as it is running. The factory service launches an instance of an "application manager service" that controls the actual application execution. The application manager uses simple event notification mechanisms to publish important messages to the user's Grid context as a leaf node in his or her directory. This is done by having the XDirectory subscribe through a proxy for messages that are generated by the user's applications. Application event messages usually take the form of simple XML documents with annotation and, possibly, a GSH of some remote object. For example, one such message could be "The output file from *this* run of *this* application can be located via *this* query on *this* metadata directory."

In its current form, notification in OGSI is based on a simple, point-to-point, non-reliable event push model. Notifications result only from changes in Service Data of a service. However, because these application messages may not be the result of a service state change (they may be published directly by the Fortran application), the XDirectory service required a richer messaging and notification interface that operates through the use of a network of message channels and message brokering systems.

A natural question to ask is why we did not represent all the entries in the database as individual service data elements? Clearly, not every database that has a Grid service interface should have every database entry rendered as an SDE. In most cases the scale is wrong to do this. However, in this case it may be possible. Indeed, each entry in the XDirectory database corresponds to a simple XML schema. The XDirectory interface is somewhat richer than that provided by a GridService port, but there is nothing that would prevent us from mapping this interface to the query and update on the GridService port type. We plan to test this hypothesis in the year ahead as part of a larger investigation that will also involve the use of OGSA-DAI [16].

We also note that at the time of this writing we have not yet implemented the security required to make this service a private repository for the user. This will soon change[2].

4. Generating Interfaces to Grid Services

It would be ideal if for every Grid service for which it is possible or desirable to have a human interface, there was a user interface portlet installed in the portal server. Unfortunately, this is not practical or even realistic. The problem is that the portal architecture would require a user, when he or she must interact with a new Grid service, to install and configure a new portlet into his or her portal collection. This would also require that the portal manager deploy the new portlets each time a new user-facing Grid service comes on-line.

There are two solutions to this problem.

1. For many services, the user interface to the service is very simple. For example, consider a Web service that takes the name of a city as a parameter and returns the current temperature. A simple interface for this can be derived, on-the-fly, from the WSDL for the service.
2. For some services, the user interface may have requirements that go beyond what is specified in the WSDL document. For example, a Web service, when given the name of a chemical compound, may return an XML description of the geometric structure. However, a human user would probably not wish to see the XML, but rather view a 3-D rendering of the structure. In this case, the WSDL cannot tell us how do go about this rendering and we must provide means for the service to also supply the needed visualization client.

In order to address the first case, we have built a service called Xydra/OntoBrew [1], which does such an on-the-fly WSDL to web-form generation. Xydra is capable of handling WSDL with XML Schema complex types and it is extremely easy to customize the generated XHTML output by using a template. It can also be configured to use an ontology to allow a rich set of constraints and relationships between parameters. An example of how this can be used in show in Figure 3 which illustrates Xydra/OntoBrew operating on the GT3.0 Index Service GridService port. Xydra/OntoBrew is used by our application factory service which launches an instance of an "application manager" as described in the previous section. If the application manager needs further input from the user, such as application parameter values, it automatically places an entry into the XDirectory that consists of the OntoBrew generated user interface.

The second case is the more interesting. A recent OASIS specification, Web Services for Remote Portlets (WSRP) [17], describes an interesting way for a Web service to be a "remote portlet". The idea is to have the Web service return mark-up instructions to the portal to present to the user in a portlet frame. We are currently in the process of investigating how WSRP may be integrated with OGSI. However, a different approach is also possible. If a Grid service needs a special client to interact with users, the service data for that service may contain information about that client. For example, if the portal can automatically search for a standard service data element called "Interface Client", the information in that SDE can provide the portal with instructions for finding, launching and displaying the client. The client information may describe a Java applet or Java Web Start application. The client information may also be a Grid Service Handle for a client factory that can create a client service with a remote display functionality the user can interact with. We are currently exploring all of these are ideas.

[2] We expect to have authentication based on XML signature working by the time of this workshop. Our goal is complete interoperability with GT3.0 and any forthcoming Grid Services security specification.

5. Conclusions

Providing a portal that allows a user access to Grid systems is important to making these environments both manageable and productive. Our experience shows that we can build a very effective portal architecture based on a Grid service model, but we have several areas where much more work is needed. We have demonstrated that it is possible to build custom portlet interfaces for external Grid services that allow the portal user an interactive look-and-feel that completely hides the fact that the service is rendered remotely. But this is not unusual. Most sophisticated web portals use a three-tier architecture where the back-end servers may be on remote resources. The more interesting problem is how to allow the user to discover new services and integrate them into the portal environment without requiring action on the part of the portal administrator to add a new portlet or even restart the server? In this work we have described one approach based on an automatic generation of a web interface from the WSDL for the service. If the web service is also a Grid Service we have two additional advantages. First, a custom portlet can be used to interrogate any Grid Service for basic service data elements. One need only supply a GSH and a handle resolver. A more interesting approach that we are currently investigating involves creating a service data element that provides a GSH to a client service factory for instantiating a client service capable of communicating with the portal and providing a view onto the newly discovered service for the user. The WSRP specification provides a means to allow a remote service to be a remote portlet, but we feel we can go much further with Grid services.

Within the OGSI community there is frequent discussion [25] about the concept that a Grid Service can be an entity that is much lighter weight than a web service. For example, it may be a document. The concept that a document may be bound to an application, and invoked with a mouse click, is not new to computer users. Nor is the concept that a URL can automatically deliver a remote documents through a remote web server to your desktop. So it is not a great leap to imagine a GSH referring to a workflow document, or a metadata query, which when resolved results in a view to a dynamically launched, executing Grid application running on behalf of the user. This is precisely the type of functionality we feel most user want from the Grid and it is where we are going with the portal projects.

6. References

[20] Allan, R**.,** Integrated e-Science Environment for CLRC, http://esc.dl.ac.uk/IeSE

[19] Barbera, R., "The GENIUS Grid Portal," Computing in High Energy and Nuclear Physics, 24-28 March 2003, La Jolla, California.

[1] Chipara, O., Slominski, A., Xydra – An automatic form generator for Web Services, see: http://www.extreme.indiana.edu/xgws/xydra/

[2] Cubera, F. and Co-Authors, "Business Process Execution Language for Web Services, Version 1.0," http://www-106.ibm.com/developerworks/library/ws-bpel/

[3] *The Grid: Blueprint for a New Computing Infrastructure*, Ian Foster and Carl Kesselman (Eds.), Morgan-Kaufman, 1998.

[4] Foster, I., Kesselman, K., Nick, J., Tuecke, S., "The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration", in *Grid Computing: Making the Global Infrastructure a Reality*, Fox, G. Ed. Wiley, 2003.

[6] Fox, G., Berman, F., and Hey, T., *Grid Computing: Making the Global Infrastructure a Reality*, Fox, G. Ed. Wiley, 2003.

[13] Fox, G., Gannon, D., "Computational Grids", IEEE Computer Science Eng. Vol. 3, No. 4, , pp. 74-77, 2001

[5] Fox, G., Pierce, M., Gannon, D., Thomas, M., An Overview of Grid Computing Environments, Global Grid Forum GFD-I.9. http://www.gridforum.org/documents/GFD, 2003.

[7] Gannon, D., Chiu, K., Govindaraju, M., Slominski, A., “A Brief Introduction to the Open Grid Services Infrastructure”, to appear Journal of Computing and Informatics, Special Issue on Grid Computing. 2003.

[8] Gannon, D. and Co-Authors, “Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications,” Journal of Cluster Computing, 5(3): 325-336 (2002)

[15] Global Grid Forum, http://www.ggf.org

[24] Govindaraju, M., et. al., “Merging the CCA Component Model with the OGSI Framework”, Proceedings 3rd International Symposium on Cluster Computing and the Grid, pp. 182-199, May, 2003, Tokyo.

[23] GridLab, The GridSphere Portal http://www.gridsphere.org/gridsphere/gridsphere

[12] The Grid Physics Network, http://www.griphyn.org/

[10] Johnston, “Implementing Production Grids”, in *Grid Computing: Making the Global Infrastructure a Reality*, Fox, G. Ed. Wiley, 2003.

[9] Krishnan, S. and Co-Authors, “The XCAT Science Portal”, Proceedings SC2001, Nov. 2001, Denver.

[17] Kropp, A., Leue, C., Thompson, R., Web Services for Remote Portlets (WSRP), OASIS http://www.oasis-open.org

[22] NCSA Alliance Scientific Portal Project, http://www.extreme.indiana.edu/alliance

[14] Network for Earthquake Engineering Simulation Grid (NEESgrid), http://www.neesgrid.org

[16] Open Grid Service Architecture Data Access and Integration, http://www.ogsa-dai.org.uk

[11] The Particle Physics Data Grid, http://www.ppdg.net/

[21] JSR168 Portlet Specification. http://jcp.org/aboutJava/communityprocess/review/jsr168/

[18] Tuecke, S., et. al., “The Open Grid Service Infrastructure (OGSI)”, http://www.gridforum.org/ogsi-wg

[25] Vanderbilt, Peter, “Re: Grid-SOA and service state”, GGF OGSI-WG