A General (Gentle?) Introduction to (Grid) Portals/Gateways

Anurag Shankar <ashankar@indiana.edu>
TeraGrid Gateways Team, Indiana University
November 2006

Table of Contents

1. Introduction

This document provides a general overview of portal technologies, portal standards and best practices, web services, and currently available open source portals and open source grid portals. Being somewhat TeraGrid-centric, it leaves out a lot, particularly European and other non-US grid projects and technologies. This is entirely the author's fault.

2. Definitions

The definitions of portals and gateways are somewhat contentious. Many people use the words "portals" and "gateways" synonymously. In this document however we will adopt the TeraGrid convention and define them as two distinct but related concepts.

2.1. Portal:

A portal is a web-based application that is customizable by the end-user, both in the look and feel of the portal and in the available content and applications which the portal contains. A portal, furthermore, is an aggregator of content and applications or a single point of entry to a set of tools and/or applications defined by the user or by the portal service provider.

2.2. Gateway:

A gateway is a portal that is dedicated to providing services to a specific user community. When the gateway serves a science community (physicists, chemists, etc.) we call it a "Science Gateway" (at least in the TeraGrid parlance).

Examples of production science gateways include the NEESGrid portal, LEAD portal, BIRN portal, GEONgrid portal, Chronos system portal, RENCI bioportal, SCOOP ADCIRC interface, Purdue Nanohub, IU CIMA X-Ray Crystallography portal, ServoGrid portal, Belfast Gene Grid portal, MyGrid Bioinformatics portal, UT Flood Modeling portal, and ORNL Neutron Science Gateway, etc.

Most existing science gateways are designed to either allow access to/analysis of data or to expose canned applications to users.

3. Relevant Technologies/Concepts

3.1. Servlets:

Servlets are modular Java programs that perform a specific action and create HTML output (akin to CGI scripts. Unlike CGIs, however,

- servlets do not run in a separate process (from the server),
- servlets stay in memory between requests (have session awareness built in),
- servlets are often faster,
- servlets can be run in a restrictive sandbox for testing.
- servlets are not part of any particular web standard. They are part of an
- extended servlet API included now in standard Java distributions.
- servlets can be used to build (non-standards-based) portals.
- servlets require a web container (such as Tomcat) in order to run.
- a servlet can be accessed by a web client (usually a browser) directly via a URL.

3.2. Portlets

Portlets are standards-based, portable, special servlets that produce fragments of markup (HTML) code that can be aggregated into a portal page. Created using an extended Java portlet API, portlets are strung together to build a complex, complete portal. Each portlet typically performs one or more simple actions. However, portlets can also be arbitrarily complex.

On a typical portal page created using portlets, each frame or click may be tied to a different portlet. A portlet.xml file (called a deployment descriptor) at the server end defines precisely what and where the portlets are; another file called web.xml defines special servlets that load portlets.

As mentioned earlier portlets only generate markup fragments, not complete documents. A portal is thus required to aggregate these markup fragments into a complete web page. Also, portlets are not directly bound to a URL (like servlets); web clients interact with portlets through a portlet container (defined below).

Portlet standards are determined via the Sun Java Community Process (JCP).

3.3. Similarities and Differences Between Servlets and Portlets:

Similarities:

- Servlets and portlets are web based components that utilize Java for their implementation.
- Portlets are managed by a portlet container similar to a servlet container.
- Both of these components generate content, which can be static or dynamic.
- Both portlets and servlets have a life-cycle that is controlled by the container.
- The client/server model is used for both servlets and portlets.
- The packaging and deployment are essentially the same.
- The manner in which the classes are loaded and the class loaders that perform the work are also the same.
- Life-cycle management is similar.
- The Request and Response semantics are also similar.

Differences:

- Servlets can provide complete web pages, whereas portlets only provide fragments. These fragments are then aggregated to form a complete web page by the portal.
- Portlets aren't allowed to generated HTML code that contains tags such as base, body, frame, frameset, head, html, or title. The iframe tag can be used with caution.
- The user cannot access a portlet directly using a URL in the way that a servlet is accessed. Instead, the URL points to the page containing all of the portlets on one page.
- Communication between the web client and the portlets is performed through the portal.
- Portlets can be provided with buttons or controls to manipulate the portlet’s window states or portlet modes.
- Multiple instances of a single portlet can be placed onto the same page.
- Portlets support persistent configuration and customization.
- Portlets also support user profile information.
- Portlets support two scopes within the session; application scope and portlet scope.

There are several things that servlets are allowed to do, but portlets aren't:

- Portlet aren't allowed to set the character set encoding of the response.
- Portlet also aren't allowed to set the HTTP headers on the response.
- Portlet cannot manipulate the URL of the client request to the portal.

3.4. Web Container:

A web container (WC) is a server that runs Java servlets and allows access (to servlets) via HTTP. The most widely used WC is Apache Tomcat (current version is 5.5.17). Tomcat can run as a stand-alone app and listen for HTTP requests on port 80 or work in concert with a real web server (such as Apache httpd; communication between Tomcat and Apache occurs via a "connector" in this case).

Other web containers include Borland Enterprise Server, JBOSS, IBM Websphere, Resin, BEA WebLogic, Apple WebObjects, etc.

3.5. Portlet Container:

A portlet container is software running as a Java app inside a web container that allows a portlet to run. Or, in reverse order, a portlet uses a portlet container which in turn uses a web container. Portlet containers are typically also standards-based, meaning they can run standards-based portlets.

So, Portlet container = Software running inside a Web Container that understands portlets

Examples of popular open source portlet containers are Gridsphere, Apache Pluto, jPortlet, JBOSS Portal, Jetspeed2, Liferay, etc. Commercial ones include Oracle AS Java Portlet Container, Sun Java System Portal Server, IBM WebSphere, BEA WebLogic, etc.

Note: At the Portals and Portlets 2006 workshop in Edinburgh, we learned that Sun intends to release their Portal Server 7.0 as open source in the near future.

3.6. Simple Object Access Protocol (SOAP):

SOAP is an open, lightweight standard for exchanging XML based messages over the Internet. HTTP is most commonly the transport mechanism for SOAP. However, new frameworks such as Blocks Extensible Exchange Protocol (BEEP) from IETF or HTTP-R from IBM can also be used.

SOAP implementations are available for Java, perl, python, and other languages.

SOAP also forms the basis for the now highly popular Web Services technology (described below).

3.7. Security Assertion Markup Language (SAML):

SAML is a framework for exchanging authentication and authorization information securely between two partners over the network using SOAP. SAML is agnostic of each partner's security systems or underlying apps/platforms. It standardizes the exchange of credentials in an XML format called "assertions".

3.8. Web Services (WS):

[Note: Hereafter, the acronym "WS" is used to refer to both the singular and plural form of the term "Web Services". The context should make it obvious which.]

WS are open standards-based (XML, SOAP, HTTP, etc.) modular, distributed, dynamic web apps that are self-described, published, located, or invoked over the Internet. A collection of different WS might be used to create a more complicated app (for example via a portal/gateway).

A WS usually consists of a server end and a client end. The transport mechanism between the server and the client can be any arbitrary XML messaging service, for example SOAP, HTTP GET/PORT, or XML-RPC. Ultimately, the the client needs to send an XML query to which the server must respond with an XML result.

Standard WS APIs exist today for most programming languages (Java, perl, c, etc.), making WS programming easy. Businesses like Google and Amazon.comeven provide their own WS API to allow programmatic access to their services.

3.8.1. WS Pros

- WS provide interoperability between various software applications running on disparate platforms/operating systems.
- WS use open standards and protocols. Protocols and data formats are text-based where possible, making it easy for developers to comprehend.
- By utilizing HTTP, WS can work through many common firewall security measures without requiring changes to the firewall filtering rules. Other forms of RPC may more often be blocked.
- WS allow software and services from different companies (even legacy ones) and locations to be combined easily to provide an integrated service.
- WS allow the reuse of services and components within an infrastructure.
- WS are loosely coupled thereby facilitating a distributed approach to application integration.

3.8.2. WS Cons

- Many WS standards are currently nonexistent or still in their infancy (compared to more mature distributed computing open standards such as CORBA). This is likely to be a temporary disadvantage however as most in the industry have committed to the OASIS standards.
- Due to the text-based formats (XML, SOAP, etc.) WS may suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off between performance and portability.

3.9. Web Services Definition Language (WSDL):

WSDL is a special XML grammar created to allow a WS to self-describe itself to its consumers (clients). A WS' WSDL file defines the communication protocol used, the methods (that can be invoked remotely) available, data transfer syntax, and the location of the service endpoints. WS are advertized using UDDI.

3.10. Universal Description, Discovery, and Integration (UDDI):

UDDI is a specification used for companies to dynamically find and use WS. It specifies the XML format in which data is stored and an API for searching existing data using SOAP.

UDDI is also a fully operational implementation of the spec and a registry for businesses worldwide to list their WS (using WSDL) on the Internet.

Note: Internet Explorer now supports UDDI name resolution via the Real Names Keyword System. Just type "uddi" in the Internet Explorer address bar, followed by the name of the target company. For example, type "uddi ariba".

3.11. Web Services for Remote Portlets (WSRP):

WSRP is a new, presentation-oriented specification aimed at portals/portlets. It is finding wide acceptance in the industry as a tool for exposing the end-user to WS and to reap the benefits of SOA.

In normal WS, the SOAP response to a WS request contains only the data desired. The (remote) client is responsible for rendering this data however it wants to (by creating the appropriate HTML that displays the data, for example). This takes programming.

WSRP on the other hand adds the ability at the WS server end to return pre-formatted markup (in other words HTML). Portal creation then becomes a simple aggregation exercise requiring little programming (at least in principle!).

While WS provide the ability to reuse back-end services, WSRP allows the reuse of the entire user interface!

The WSRP standard was released by OASIS (described later) in 2003. The Apache project provides WSRP4J, a reference implementation of WSRP 1.0 spec. (Xiabo Yang and Rob Allan at the UK e-Science Lab at Daresbury have recently taken a comprehensive look at WSRP/WSRP4J as a portal technology. To see their PPT, see the reference to "Portals and Portlets 2006" in the bibliography.) The new spec, namely WSRP 2.0, is currently out as a draft.

3.12. Service Oriented Architecture (SOA):

SOA is a style of enterprise architecture that enables the creation of applications built by combining loosely coupled and interoperable services. These services inter-operate based on a formal definition (or contract) which is independent from the underlying platform and programming language. The interface definition encapsulates (hides) the language-specific service implementation. A SOA is independent of development technology (such as Java, .NET etc) and is therefore vendor independent. The software components become very reusable because the interface is standards-compliant (eg. WSDL) and is independent from the underlying implementation. So, for example, a C# (C Sharp) service could be used by a service consumer developed in Java. The "service" part of SOA is usually WS.

4. How Do These Work Together?

Consider a case where you want to establish a simple WS that serves the current time in Bloomington. Here is the sequence of events that follow:

a) you create a simple app (say a shell script called "bltime") that, when invoked, outputs the temperature in some format ("12:45:05 PM", for example),

b) you create the WS description using WSDL and publish it using a web server (say Apache) or a web container (Tomcat) or the two together,

c) User's WS client connects to the WS URL (which is really URL to the WS' WSDL file) and retrieves the WSDL via a HTTP GET,

d) the WSDL informs the client about what methods are available (only one in this case - bltime), how to invoke them, etc.,

e) the WS client sends a HTTP POST request containing a SOAP message to invoke the right method (bltime),

f) the WS at the server end unwraps the SOAP message and invokes bltime,

g) the WS at the server end packages the Bloomington time in a SOAP message,

h) the WS at the server end sends to the client this SOAP message as the response to the POST request in step e) above.

i) the WS client app extracts the Bloomington time from the SOAP response and uses or displays it however it wants to (or extracts the WSRP rendering info and uses it for display),

If you wanted to secure your WS, there might also be a SAML exchange of credentials before the WS returns a result.

To integrate the result into a portal page, you might write a little portlet that uses a WS client to essentially do steps c-g.

Here is a more typical scenario in detail on how a WS-enabled app works:

- The app calls a client stub (usually from a stub library) which invokes a WS locally,
- the client stub converts the local invocation to a SOAP request (this process is usually called "unmarshalling" or "deserializing"),
- a service stub at the WS end receives the request,
- the service stub invokes the methods described in the WSDL that actually carry out the work,
- the service stub envelops the results in a SOAP response,
- the response is sent back to client end over HTTP,
- the client stub receives the response,
- the client stub turns it into something the client app understands.

Portals constitute only one category of potential WS consumers. *ANY* app whatsoever can be WS-enabled, including legacy apps (via a WS wrapper).

4.1. The Java Commodity Grid (COG) Toolkit:

COG is a Java toolkit from ANL that provides a Globus toolkit-independent abstraction layer to access Globus services programmatically.

4.2. Where Does the Grid Fit In?

Portlets and WS have been described in a generic way thus far. A majority of existing portals and WS out there have nothing to do with grids.

When portlets or WS include grid functionality (using say the COG kit), they become "Grid Portlets" and "Grid Services" (more on Grid Services later).

5. Relevant Standards

5.1. OASIS Standards

OASIS stands for "Organization for the Advancement of Structured Information Standards". It is a global consortium dedicated to developing open WS
standards. Relevant OASIS WS standards are:

- SAML: see 2.7.
- UDDI: see 2.10.

- WSDM (WS Distributed Management): a specification for WS distributed management of management-enabled resources such as network management devices, consumer electronic devices (televisions, DVD players, PDAs). Each resource is represented as a WS.
- WS-Notification (WS-N): a specification that standardizes the way WS interact using "notifications" or "events".
- WS-Reliability: a SOAP based specification for reliable messaging in WS (supplements SOAP over HTTP, the commonly used method).
- WSRP (WS for Remote Portlets): see 2.10.
- WSRF (WS Resource Framework): a specification that provides methods that stateless WS can implement to become stateful.
- WS-Security: a specification to enhance SOAP to provide message integrity and confidentiality. The specified mechanisms can accommodate a wide variety of security models and encryption technologies (such as X.509, Kerberos, etc.).

etc.

5.2. W3C Standards

W3C is the "World Wide Web Consortium" and is responsible for the following relevant standards:

- SOAP: see 2.5.
- WS-Addressing: a specification that provides transport-protocol-neutral mechanisms for addressing WS.
- WSDL: see 2.8.

5.3. Not-Quite-Yet WS Standards

These are either purely drafts by vendor(s), out as RFCs, or currently being under consideration by standards organizations:

- BPEL(Business Processing Execution Language)
- WS-AtomicTransaction
- WS-BusinessActivity
- WS-CAF (Composite Application Framework)
- WS-CallBack
- WS-CDL (Choreography Description Language)
- WS-Coordination
- WS-Discovery
- WSEL (Endpoint Language)
- WS-Enumeration
- WS-Eventing
- WS-Federation
- WSFL (Flow Language)
- WS-I
- WS-I18N (Internationalization)
- WS-Interoperability
- WS-Management
- WS-MessageData
- WS-MessageDelivery
- WS-MetaDataExchange
- WS-Policy
- WS-PolicyAssertions
- WS-PolicyAttachments
- WS-Provisioning
- WS-ReliableMessaging

- WS-SecureConversation
- WS-SecurityPolicy
- WS-Transfer
- WS-Trust
- XML-Encryption
- XML-Signature

etc.

5.4. Global (Open) Grid Forum (GGF/OGF) Standards

GGF (now OGF) is an international forum of users, developers, and vendors from industry and research. Its purpose is to create open standards for grid computing in research and industry and to enable deployments of interoperable grids. The main, all-encompassing grid standard GGF is responsible for is the OGSA, described next.

5.4.1. Open Grid Services Architecture (OGSA):

OGSA is an emerging open, component-based, server agnostic, service oriented architecture and a standard that extends WS to the grid world. A "Grid Service" represents extensions to WS for grids. In OGSA, everything is a WS.

OGSA is creating specifications for the following Grid Services:

- Context services (policy, VO)
- Data services (access, integration, transfer, replication)
- Execution management services (execution, workflow mgmt, workload mgmt, execution planning, job mgmt)
- Information services (monitoring, event mgmt, discovery, logging)
- Infrastructure services (WSRF, WS-N, WSDM, naming)
- Resource management services (reservation, configuration, deployment, provisioning)
- Security services (authN, authZ, integrity, boundary traversal)
- Self-management services (heterogeneity mgmt, optimization, SLA, QoS)

5.4.2. OGSA and OGSI:

How the WS listed above are implemented is addressed by the Open Grid Services Infrastructure (OGSI), a companion standard that defines what a Grid Service is and provides interoperability between grids designed using OGSA. Read the OGSI spec for an excruciatingly detailed (and dense) view of how Grid Services should work.

Naturally, OGSA's core infrastructure is built upon OGSI.

5.4.3. OGSA WGs:

OGSA is a work in progress but the following GGF working groups (among many more) are hard at work on OGSA:

- OGSA (Architecture): established Sept. 2002, define OGSA architecture
- OGSA-AuthZ: define specs for interoperable authorization components for OGSA
- OGSA-ByteIO: define a minimal WS interface for Byte-IO. Byte-IO is a byte stream I/O protocol.
- OGSA-Data: All things data.
- OGSA-DAI (Data Access and Integration): a project to produce a framework to access and integrate data (relational, XML, files) on the grid

- OGSA-DMIS (Distributed Monitoring and Information Systems)
- OGSA-EMS (Executive Management Services): instantiating and managing tasks
- OGSA-BES (Basic Execution System) produce a minimal subset of EMS recommendations
- OGSA-HPCP (HPC Profile): create a profile of how existing specifications should be combined to address batch job scheduling of scientific/technical applications (HPC use case)
- OGSA-Naming: work on two specifications (RNS and WSNR) to realize a three level name space for OGSA and to produce a WS-Naming naming specification based on WS-Addressing
- OGSA-RSS (Resource Selection Service): provide specifications for protocols and interfaces that generate a list of resources
- JSDL (Job Services Description Language): a language for describing the requirements of jobs for submission

etc.

5.4.4. OGSA/OGSI and the Globus Toolkit:

GT3 includes a complete implementation of OGSI. GT4 adds OGSA capabilities using WSRF. Together with WS-N, the WSRF specifications describe how to implement OGSA capabilities using WS.

5.4.5. OGSA and XCAT:

OGSA is derived from various previous efforts, one of which is Dennis Gannon's XCAT project, an implementation of OGSI + the Common Component Architecture (CCA) for a Globus-based grid. (CCA is a 2000 DOE project involving several national labs aimed at defining a standard component architecture for HPC.)

5.5. Distributed Management Task Force (DMTF) Standards

DMTF is an industry organization leading the development of management standards and the promotion of interoperability for enterprise and Internet environments. Relevant DMTF standards are CIM and WBEM.

5.5.1. Common Information Model (CIM):

CIM provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. CIM's common definitions enable vendors to exchange semantically rich management information between systems throughout the network. CIM is both a spec and a schema. There are several open source implementations of CIM and many vendors (IBM, Sun, Intel, Microsoft, Oracle, etc.) have incorporated CIM into their hardware/software.

5.5.2. Web-Based Enterprise Management (WBEM):

WBEM is a set of management and Internet standard technologies developed to unify the management of distributed computing environments.

5.5.3. How does OGSA relate to CIM/WBEM?

GGF and DMTF have worked together to incorporate CIM and WBEM into OGSA for management of resources and services.

5.6. Where does the Web Services Interoperability (WS-I) Organization Fit In?

WS-I is a consortium of (vendors and end users) concerned with figuring out how to ensure that WS products actually work together and inter-operate. WS-I is not a standards body. They have created what are called

WS-I profiles. Profiles clarify any ambiguities that exist in the specifications of standards. WS-I has gone through SOAP, UDDI, and WSDL and created profiles for each.

5.7. What is CIMA?

CIMA stands for "Common Instrument Middleware Architecture". It is a NSF Middleware Initiative (NMI) project aimed at "grid enabling" instruments as real-time data sources to improve accessibility of instruments and to facilitate their integration into the grid. The CIMA middleware is based on current grid implementation standards and accessible through platform independent standards such as the OGSA and the CCA. A variety of instrument
and controller types will be supported.

The CIMA project PI is Rick McMullen, director of IU's Knowledge Acquisition and Projection Lab (one of the Pervasive Technology Labs). Kia Huffman is the project lead.

5.8. What is Open Knowledge Initiative (OKI) and its Relationship to Sakai?

OKI is a collaborative project (started at MIT) that develops and promotes specifications that describe how the components of a software environment communicate with each other and with other enterprise systems. OKI specifications enable interoperability by defining standards for SOA. Through this work OKI seeks to open new market opportunities across a wide range of software application domains.

To this end, OKI has developed and published the Open Service Interface Definitions (OSIDs). The OSIDs define important components of a SOA as they provide general software contracts between service consumers and service providers. This enables applications to be constructed independently of any particular service environment, and eases integration.

OSIDs are software contracts only and therefore are compatible with most other technologies and specifications, such as SOAP, WSDL, etc. They can be used with existing technology, or open source/vended solutions.

OSID bindings are provided in Java, PHP, and soon Objective C and C#.

Sakai 2.0 is built using these OSIDs.

6. Portlet Standards

6.1. JSR 168:

JSR 168 is a Java portlet specification standard developed by the JCP. It defines a "contract" between the portlet container and portlets. (JSR stands for "Java Specification Request". It's a standard way to "request" additional functionality. As of yesterday, there are 927 JSRs.)

JSR 168 v1.0 specification came out in Oct. 2003 and defines

- the runtime environment for portlets (i.e. the portlet container),
- the API (a "contract") between the container and the portlets,
- mechanisms to store transient and persistent data,
- mechanism used by a portlet to include a servlet or JavaServer Pages (JSP),
- how a portlet should be packaged for easy deployment,
- binary portlet portability among JSR 168 portlets,
- how to run JSR 168 portlets as remote portlets using the WSRP protocol.

(Some developers feel that JSR 168 is too narrow a standard since it does not provide any guidance needed to code real world apps. For example, JSR 168 does not address issues such as navigation, cross-portlet context, common portlet services, etc. The new spec, namely JSR 286, improves the situation somewhat but still leaves a great deal of the machinery to create real portals entirely up to the developer to choose as they please.)

6.2. JSR 286:

JSR 286 is the new portlet standard (v2.0) and has not yet been released. It adds/improves support for J2EE 1.4, WSRP, intra-portlet communication, etc. to the older (JSR 168) specification. It is expected to be finished in May 2007 (a public draft is expected in Dec. 2006).

6.3. What does JSR 168 Compliance Really Mean?

It means that a portlet is written using the JSR 168 standard Java API. It also means that a portlet container that is itself JSR 168 compliant can host these portlets without modifications. This is what makes portlets truly portable (the "port" part of the portlet); they can be exchanged easily among JSR 168 portlet containers.

7. Portal Best Practices

The following apply to portals generally:

- Universal access (anytime, anywhere, seamless)
- Personalization
- Standards-based content access
- Platform, programming language independence
- Uniform deployment of and access to portal services (a service API)
- A design that uses reusable portal components (portlets)
- Ability to do rapid portal development (by using and aggregating portlets)
- Ability to upgrade underlying technologies easily
- Integration with existing functionality
- Easy to develop new functionality
- Adoption of open standards (JSR 168/286)
- Development of easily sharable components (portlets)
- Creation of a portlet marketplace to facilitate exchange
- Use of webstart to launch common Java-based tools
- Use of commodity and open source software (Java, XML, HTTP, etc.)
- Ability to interface with existing software
- Close cooperation among major players
- Airtight security
- Support for multiple authentication and authorization systems
- Single sign-on capability
- Extensibility
- Scalability

8. Grid Portal Best Practices

The following best practices (in addition to those listed in 5 above) apply specifically to grid portals:

- Use of commonly used grid software (GT, COG kit, MyProxy, etc.)
- Use of an abstraction layer (such as Java COG kit) to access grid services
- Ability to run client apps and portal services on separate hosts
- Easy and intuitive UI for authentication/authorization, file transfer, job management, system status, help, etc.

- Support for grid authN (X.509)
- Ability to create easily customized JSR 168 portlets for batch job submission, file transfer, grid information services, data transfer and management services, workflows, etc.
- Data discovery - search-able metadata directories
- Data access visualize, publish, download, curate

9. Sources of GP Best Practices

Current grid portal best practices are based largely on work by the GGF Grid Computing Environment Research Group (GCE-RG) and on past portal projects such as GridPort/HotPage, Alliance Portal, DOE Components, NEESGrid, GRaDS, etc. Lessons learned were that:

- Everyone was spending a significant effort reinventing the wheel (namely secure logins, remote file manipulation, command execution, access to info servers, etc.).
- Portals and components were highly customized, thus making it difficult to quickly create a new portal.
- Components could not be shared among groups developing portals.
- Sharing UI components or making services inter-operate was difficult or impossible.
- There existed no well defined interfaces to portal services.

10. Commercial Enterprise Portal Software

IBM Websphere, Plumtree, SAP Enterprise Portal, BEA WebLogic, PeopleSoft Enterprise Portal, Oracle Application Server, Sun ES portal server, Novell extenNd, CA CleverPath, Microsoft SharePoint, Tibco PortalBuilder, Vignette Application Portal, etc.

11. Open Source Portal Software

11.1. Non standards-based

- PHP based: Mambo, Drupal, Joomla, Midgard, Xaraya, etc. (Note that Purdue's excellent Nanohub is a non-standards based portal that uses Mambo.)

- Java/XML based: jPortlet, Apache Lenya, etc.

11.2. Standards-based (J2SE/J2EE/XML/JSR 168)

11.2.1. Sakai:

Sakai is a community project led by U. Michigan and IU (with participation from many academic and commercial partners now; see sakaiproject.org). Its goal is to develop a new collaboration and learning environment (CLE) for higher education. Sakai supplies the application framework, tools, and components that allow the creation of any collaborative system but has been used primarily in building course management systems (CMS). (IU's current OnCourse CL CMS is based on Sakai v2 - so are several others in academia.)

Sakai includes an Enterprise Services-based portal, a complete Course Management System with sophisticated assessment tools, a Research Support Collaboration System, a Workflow Engine, and a Technology Portability Profile, etc.

- Sakai is based on OKI.
- Sakai runs under Tomcat web container (but takes control of it completely).
- Sakai provides JSR 168 compliant portlets that can run in Gridsphere, uPortal, or Pluto.
- Sakai is currently exploring WSRP to solve some of the sticky problems that crop up in creating a full fledged portal application.

11.2.2. Jakarta Pluto:

A portlet container from the Apache project. Runs under Tomcat web container. Runs JSR 168 portlets.

11.2.3. Gridsphere (GS):

GS is a portlet container from the European Gridlabs project. It runs in the Apache Tomcat web container and is JSR 168 compliant. It seems by far to be the most widely used portlet container currently.

For more info on GS, go to the bibliography at the end.

11.2.4. Others:

Exo, Jakarta JetSpeed 2, Liferay, JBoss/Nukes, uPortal, jPortlet, OpenPortal, InfoGlue, Kosmos, Stringbeans, etc.

11.3. Open Source Grid Portals

Many of the open source portals described above can be used to create grid portals with requisite programming resources. JSR 168 grid portlets could in principle be developed that run in any JSR 168 compliant portlet container. However, creating grid portal software is still a major undertaking and is left to hard core software developers.

We will discuss only the three major grid portals that exist today. They are:

- The Open Grid Computing Environment
- The GridPort Toolkit
- Gridsphere GridPortlets

[It turns out that the first two are really the same now, meaning there are in fact only two frameworks left (excluding any other European efforts that the author is unaware of).]

11.3.1. Open Grid Computing Environment (OGCE)

OGCE (www.ogce.org) is an open source portal developers' consortium established in Fall 2003. OGCE provides a framework for web-based grid portals and science gateways. Current OGCE portals include the TeraGrid User portal, LEAD science gateway, NEESGrid portal, DOE Fusion portal, NCSA Alliance portal, and others.

The OGCE effort is being led at IU at the Community Grids Lab (CGL) by Marlon Pierce, the PI on this NSF grant. It is a 3-yr grant ending in August '06 but a no-cost extension will likely extend it by another year. OGCE v2 spans the CGL, the Extreme! and the Community Grids Labs, and TACC. Core portlet development, support for portlet development tools, building and testing environments are usually carried out at the CGL, while science application support service development areas are the focus for Extreme Lab. Other OGCE partners include NCSA, UMichigan, UChicago/ANL, and SDSU. The OGCE project essentially brings all major grid portal players under the same umbrella.

- Current OGCE software release is 2.0. It is an integrated release (meaning all software necessary to run the OGCE portlets is included). In addition, OGCE v2.0 includes portlets from several other grid portal projects.

- OGCE release 2 provides JSR 168 compliant portlets.

- Portlets can run under Gridsphere or uPortal portlet containers.

- OGCE services employ the COG API (described in 3.1 earlier).

- OGCE is built using open software such as Apache Ant/Maven, COG, etc.

- Non-grid, collaborative OGCE portlets from CGL include shared spaces (chat, discussion, calendar, newsgroups, polls), whiteboards, group authorization, shared application services, and Access Grid).

- OGCE JSR 168 grid portlets include:
    - Grid certificate/MyProxy based single sign-on authentication to grid resources using GSI,
    - Remote file management, file staging,
    - Remote job submissions via Globus GRAM/RSL,
    - Remote job submission to Condor pools,
    - Remote job management,
    - Resource discovery and monitoring via GPIR (Grid Portal Information Repository - a TACC developed info repository system that stores system monitoring/status info).

- Though the COG Java toolkit, OGCE supports multiple, simultaneous versions of the Globus toolkit (GT2, 3, and 4).

- New grid portlets require COG programming.

OGCE is funded by the NMI program and is distributed as part of the NMI GRIDS software releases. It spans several different core NMI technologies, including:

- The Sakai Project: Sakai members collaborate with OGCE partners to develop standards compliant portlets and WS for collaboration.
- GridPort: The Texas Advanced Computing Center and San Diego State University built OGCE-compatible portlets and services. These are available as stand-alone components or as an integrated build.
- Tupelo: Tupelo is a semantic grid-based data and metadata system.
- The COG Toolkit.

11.3.1.1. The Linked Environments for Atmospheric Discovery (LEAD) Gateway

The most ambitious science gateway being built using OGCE is a severe weather gateway for the LEAD project. It is being developed at IU by Dennis Gannon's team (including Marcus Christie and Suresh Marru) in collaboration with the LEAD PI Kelvin Droegemeier at Univ. of Oklahoma.

LEAD-specific portlets being developed include the following additional functionalities:

- Workflows,
- Data access via the OGSA-DAI framework,
- Monitoring using TACC's GPIR

11.3.2. Relationship between OGCE, CHEF, Sakai:

CHEF is a now-defunct portal development framework from UMichigan used by OGCE v1.0. Sakai v1 was essentially a port of the CHEF API with backward compatibility with CHEF. CHEF/Sakai v1 have evolved into what is now Sakai v2. Since Sakai v2 portlets are JSR 168 compliant, OGCE v2 release includes them in its integrated release. (Sakai portlets however run in a separate Tomcat web container. An OGCE portlet then

uses the HTML 4.0 IFrame tag to
display it. This is done for authN reasons.)

11.3.3. The GridPort Toolkit

Developed at TACC, GridPort started as a perl based portal toolkit in 1997 created to build the NPACI Hotpage portal. It has since been rewritten in Java and provides JSR 168 portlets.

Portals based on GridPort include the NPACI portal, TACC user portal, SURA Grid User Portal, etc.

GridPort is mentioned here since it was the first grid portal toolkit. Development has essentially stopped since the GridPort and OGCE efforts have now merged. However, a current GridPort release is still available at the GridPort website. It is an independent integrated build of TACC portlets along with the Gridsphere portlet container.

- Current version is v4.0.1. It is an integrated build.

- GridPort uses the Gridsphere portlet container.

- Provides JSR 168 compliant portlets:
    - GRAM job submission
    - Condor job submission
    - GIS and GRIS
    - GridFTP
    - GPIR
    - CFT: Comprehensive File Transfer. A TACC developed file transfer system
    - SRB support

- Uses only the Java COG API.

- New portlet development requires COG programming.

11.3.4. Gridsphere GridPortlets (GP)

Developed under the auspices of the European Gridlabs project at the Albert Einstein Institute (AEI) of the Max Plank Institute for Gravitational Physics (MPG) in Potsdam. Gridsphere's authors are Jason Novotny (SDSC) and Oliver Wehrens (MPG). GP was written chiefly by Michael Russell (now at Poznan Supercomputing and Network Center, Poland).

- GP is the fastest framework currently (in the author's experience) to build a grid portal owing to the rich API provided by both GS and GP.

- Current portals using Gridportlets include the LSU HPC portal, Cactus portal, SCOOP portal, etc.

- Current version is 1.3. It is packaged separately from GS but must be deployed along with GS. The integrated environment supports JSR 168 portlets, GS specific non-grid portlets, and GP specific grid portlets.

- GP uses the GS + its own proprietary API for portlet development. These APIs provide:

    - a high level interface to UI building tools (visual JavaBeans, Tag libraries) which obviate the need to write HTML to display results
    - a high level, uniform framework to access underlying services, resources, etc.

- a high level interface to the underlying Globus infrastructure implemented via the COG kit

GP's reliance on these APIs has the downside that GP grid portlets do not run anywhere except in the GS portlet container. However, the programming time and effort saved are well worth it. (The GP team is also in the process of developing a framework that removes this dependency.)

- GridPortlets require no COG programming experience.

- Ample documentation on the API and GS/GP available. This is supplemented by locally produced documentation on how to use the APIs to create simple grid portlets. This IU doc will be included in future GS/GP releases.

- Supplied grid portlets in GP 1.3 include:
    - Globus (non WS-) GRAM batch job submission
    - Job management
    - File management that supports GridFTP, GASS
    - Grid information services
    - Credential management (using MyProxy)

The supplied GridPortlets portlets are technically JSR 168 compliant. However, they are not portable due to the factors alluded to earlier.

- GridSphere basic portlets provide:
    - User login, logout
    - User and access control management

- Multi-language support (German, English, Czech, Polish, Hungarian, and Greek).

- Significant progress has been made on GT4 GridPortlets by the user community at large.

- Flexible XML-based portal presentation description that can be easily modified to create customized portal layouts.

- Built-in support for role based access control (RBAC) (to manage access for guest, users, admins, and superusers).

11.3.5. Relationship Between OGCE, GridPort, GridSphere, and GridPortlets

GS GridPortlets uses a GridSphere- and GridPortlet-specific UI and services API (that addresses jobs, file, h/w resources, security, tasks, etc.) + the Java COG kit to develop grid portlets.

OGCE is exactly the same in that it too provides grid portlets. The exception however is that its grid portlets do not depend on a custom API as GP's do. The API (COG kit) is bundled in the integrated build. This means that OGCE portlets can run in any JSR 168 compliant portlet container and are thus more portable.

OGCE integrated build for example provides its own portlets, the GridPort portlets, NCSA portlets, and Sakai tools and portlets.

GridPort too provides grid portlets that can run in any JSR 168 compliant portlet container. Though TACC continues to develop grid portlets, GridPort and OGCE efforts have essentially merged into a single OGCE project now.

By default, all three use Tomcat as their web container and GridSphere as the portlet container in their integrated builds.

The only WS enabled entity in the current stable builds of these three is TACC's GPIR portlet. GT4 GS GridPortlets (still in alpha) support WS by default since GT4 is largely WS-based.

12. Current Grid Workflow Management Systems

Workflows are quickly becoming one of the most crucial components in grid computing. They allow an end-user to submit not just a single job but an entire sequence of compute, storage, post-processing, etc. steps at once.
Current open source grid workflow systems are:

- Askalon : U. Innsbruck
- DAGMan : U. Wisconsin
- GrADS : Collaboration among various US academic institutions
- GridAnt : Argonne
- Gridbus : U. Melbourne, AUS
- GridFlow : U. Warwick, UK
- ICENI : London e-Science Center, UK
- Karajan : Argonne
- Kepler : Collaboration
- Pegasus : ISI, USC
- Taverna : Collaboration among European institutes and industry
- Triana : Cardiff U., UK
- Unicore : Collaboration among German research institutes and industries

13. Topics Not Covered Here

- GridShib - an integration of Shibboleth and Globus-based grids (Shibboleth is an authentication/authorization framework that allows access to remote resources using one's home institution authN/authZ systems)

- GAMA (Grid Account Management Architecture) - a system for creating and managing grid accounts for portals and application users

- Portlet Bridges - technologies that allow legacy apps to be presented via portlets

- Project Gemstone - a Mozilla based rich client to access Gridsphere based portals

14. The Future

14.1. OGCE:

- Grid-enabled WSRP.
- GridFaces based on JavaServer Faces. JSF is a Java technology which makes it easy to architect software that allows a clean separation between business logic and presentation. For portal development JSF provides even finer granularity than portlets in building portals via reusable components that can be put together to create portlets. JSF is however, considered hard to implement, so Antranig Basman at UCambridge has invented RSF, "Reasonable Server Faces", to overcome JSF's shortcomings (also, Dave Meredith at UK Daresbury e-Science Lab has recently spent time exploring JSF as a portal/portlet

technology - for both Meredith and Basman talks, see the "Portals and Portlets 2006" workshop reference in the bibliography.
- XML-based workflow tools such as Karajan and OGRE.
- JSR 286 compatibility. JSR 286 defines the next version of portlet standard and has been submitted by IBM for review.

14.2. GridPort:

While GridPort and OGCE have merged, both projects will continue to create portlets. The GridPort team at TACC will focus in the future on portlets that implement:

- AJAX (Asynchronous JavaScript and XML). AJAX allows a portion of a web page to be refreshed independently of others, unlike regular portal frameworks where the entire web page must be redrawn and all portlets updated. This provides a better UI to the end-user.
- JavaServer Faces (JSF) to provide more reusable interface widgets.
- WSRP to provide a "deploy once, run anywhere" approach.
- Add services like grid information caching, grid user history, resource description service (RDS).
- Support for RFT and WS-GRAM.

14.3. GridSphere GridPortlets:

- GP v2.0 to be released soon, will introduce the "Vine" project.
- More human resources (5) to be added.
- Port of GP1 portlets to GP2.
- Many UI enhancements.
- A login portlet to support logging into a particular VO or a VO subdomain.
- Shift away from Gridsphere-specific APIs to make GP truly portable (new codebase is called "Vine").
- Support for persistence, security.
- SRB, HPSS support.
- Full support for GT4 (WS-GRAM).
- JSR 286 compatibility.
- Enable developers to run and test portlet services independently of the servlet/portlet container.
- Virtual organization (VO) support.
- Ability to plug in arbitrary security mechanisms.

Michael Russell is currently leading two new projects (among others) at PSNC, namely Grid Resource Management System (GRMS) and Grid Authorization Services (GAS). Projects utilizing this technology include BeInGrid, HPC-Europa, IntelliGrid, OMII-Europe, BREIN, etc.

14.4. New Capabilities in Work

- Fault tolerance
- Performance management
- Quality of Service (QoS)
- Persistence and scalability
- Workflow composers/monitoring

14.5. Future Technologies to Watch:

- Mobile grid: grid infrastructure for mobile users for seamless, secure, transparent, and efficient computing
- ? (ran out of time!)

15. Grid Portal Issues

15.1. Pros

- Can make high performance computing (CPU cycles, storage, networking) easier
- Essential for instituting "wide" computing
- Software is becoming mature
- Has broad industry and academia support
- Exchangeable portlet pool growing
- Better development tools
- Ability to rely on standards

15.2. Cons (Challenges)

- Ultimately HTML based (limited UI capabilities)
- Still difficult to use for new users
- Performance (can be slow compared to native, tuned client-server systems)
- Cost in FTEs great to build and to maintain portals (e.g. to upgrade underlying software, etc.)
- Security issues unresolved/unknown
- Standards still evolving

[A "Grid Portal Requirements" document has been composed by the author as part of an effort to create a TG "Gateways Primer" currently under development. To get a copy, please email the author.]

2.1. Integrating Apps into a Portal

Issues for future portals to address:

- How to dynamically add services and to expose them to users
- How to dynamically configure apps as WS
- How to integrate workflows

16. Guidelines for Building a New Portal

The following lists basic guidelines to follow when you undertake a new portal project.

Clearly, in all following cases, determine and follow best practices.

16.1. Planning Stage

- Develop a clear idea of what functionality users are after.

- Do not assume that you as the developer/IT person can decide for the user what's best for them (this is the #1 mistake a lot of developers make). This approach has a high probability of failure. Involve people who know best - users, web and UI developers, usability people, etc.

- Determine clearly if a service is worthy of being "portalized". Aim only for cases where the service provides improved/added functionality (beyond what is available now) or implements entirely new class(es) of functionality.

- Create a precise list of requirements that the portal must meet (for example the organization's security and audit policies, etc.).

- Determine which of the requirements can be implemented using current technology. Ignore the rest (or design in the ability to implement them later if available resources/time allows it).

- Portals with a clear focus are more successful (than those that try to cram in too many disparate tools and services).

16.2. Design Stage

- Spend significant time/effort on design. Follow formal design principles (see references in the bibliography section below).

- Involve users in initial design. Rely on feedback by creating dummy pages.

- Choose a software framework based on the project needs and resources available:

    - Simple portal projects or those where the resources are scant or time to deployment short, non-standards-based, popular frameworks (such as Mambo/PHP) will produce the best/fastest results. However, be aware that you may have to pay a price for this approach later if/when the growth of your codebase makes it impossible to reap the benefits of standardized components and services.
    - Large scale projects and/or those with ample time to deployment and/or HR resources should aim for standards-based framework(s) whenever possible.
    - For standards-based portals, Gridsphere/GridPortlets provides the fastest development environment. Choose OGCE if you want your portlets to run in any JSR 168 container.

- Determine if aggregation of content/services to a single page makes sense. It is advisable to keep information density low.

- Focus strongly on the UI. The portal will be judged largely by ease of use, not necessarily by content, however good. The front-end portal interface is thus VERY important. Also, UI can become a very time consuming part of the project unless UI goals are documented clearly.
- Perform *formal* usability testing whenever possible.

16.3. Implementation Stage

- Hire Java developers who also have UI design experience, if possible.

- Develop in stages.

- Easiest to provide are static content and canned apps.

- Keep tabs on available portlets that you can use as is or with modifications.

- Design portlets to be as generic as possible. Highly customized portlets are not useful outside your project.

16.4. Operational Stage

- Remember that a portal is only as good as its content, so:
    - Institute new, popular apps regularly
    - Monitor new portal technologies and incorporate them if possible and if they provide added functionality and/or ease of use

- Join a community experienced in providing grid portal services and seek guidance and support.

- Regularly monitor the portal web server, web container, portlet container, integrity of portal apps, security (institute automatic monitoring of logs, for example), monitor abuse in real-time, etc.

- Institute operational procedures, such as policies as regards production services (24x7 support, pager, etc.), problem notification and escalation, user notification – both general and emergecy, etc.

- Institute helpdesk procedures, such as providing a knowledge base online, cookbooks and training for portal use and problem debugging (to the users and helpdesk personnel), problem ticket resolution, etc. as needed.

17. A Complete List of Services for Service Implementors

The following attempts to provide a comprehensive list of services that a grid portal might provide. Please note that some these services are: a) the focus currently of research & development only, b) being implemented currently, or c) already in production in existing portals.

- Session management
- VO management
- Login/AuthN
- Policy based AuthZ
- Shopping cart
- Queries and result
- Job composition
- Job submission, execution, monitoring, profiling, interaction
- Query-based resources discovery
- Query-based app services/selection
- Problem specification
- Service registry lookup
- Service deployment
- Lifecycle management
- Workflows
- Resource scheduling
- Event notification
- Video/audio and other collaborative tools

18. Projected Cost

A rough estimate of HR costs is 2 x FTE years to develop a complete portal (more if it is a very complex portal).

19. Conclusion

As grid computing matures beyond its tight, high-end confines, the desire to provide computational, storage, database, etc. services to a much wider user base (that spans not only the sciences, but arts and humanities, business, medical fields, etc. also) is ever growing. Recognizing this, major funding agencies such as US NSF and the UK Research Council are pumping large amounts of money and effort into building national grids (such as TeraGrid, e-Science program) that make HPC resources available not only to the traditional HPC communities (such as physical scientists, engineers, etc.) but also to this new (and large) user base. In other words, the race is on to provide "deep" as well as "wide" computing.

Due to the intrinsic difficulty of using the Grid as it stands today, only technologies that hide well the

back-end complexity from users have any chance of succeeding. Grid portals seem currently to represent an appropriate (and perhaps the only) solution. Clearly, more R&D, formal usability testing, technology/design improvements and so on will be needed before our "wide computing" user base can grow from the usual 2,000 to 20,000.

20. Credits

Since this document aggregates a very large number of (web) documents, it is virtually impossible to create an authentic bibliography. However, major sources used include the Extreme Lab at IU, the Globus Alliance, the OGCE project, the Gridsphere project, the GridPort project, TeraGrid, the UK eScience program, OASIS/W3C/GGF websites, IBM, and the wonderful world of O'reilly books. Much of the most recent information included here comes from the "Portals and Portlets 2006" workshop in Edinburgh organized in July 2006 by Dr. Rob Allan, Grid Technology Group leader at the e-Science Center in Daresbury, UK. The author is grateful to Dr. Allan for making it possible for him to attend the workshop.

21. Bib(Web?)liography

- AJAX: http://en.wikipedia.org/wiki/AJAX
- Apache Jetspeed2: http://portals.apache.org/jetspeed-2/
- Apache Lenya: http://apache.lenya.org/
- Apache Pluto: http://portals.apache.org/pluto/
- Apache Tomcat: http://tomcat.apache.org/
- Askalon: http://dps.uibk.ac.at/askalon/
- BEA WebLogic: http://www.bea.com/diablo/
- BEEP: http://beepcore.org/
- BEinGrid: http://www.beingrid.com/
- Belfast Gene Grid portal: http://193.61.123.75:8080/GeneGrid/Portal
- BIRN portal: http://portal.nbirn.net/
- BREIN: http://www.gridsforbusiness.eu/
- CA CleverPath: http://www3.ca.com/solutions/Product.aspx?ID=262
- Cactus portal: https://portal.cct.lsu.edu/
- CCA: http://www.cca-forum.org/
- Chronos portal: http://portal.chronos.org/
- CHEF: http://www.dr-chuck.com/talks.php?id=13
- CIM: http://www.dmtf.org/standards/cim/
- CIMA: http://www.instrumentmiddleware.org/
- CIMA X-Ray Crystalloghy portal: http://www.instrumentmiddleware.org/
- COG kit: http://www.cogkit.org/
- Community Grids Lab: http://www.communitygrids.iu.edu/
- Condor: http://www.cs.wisc.edu/condor/
- CORBA: http://www.omg.org/gettingstarted/corbafaq.htm
- D-Grid: http://www.d-grid.de/index.php?id=1&L=1
- DAGMan: http://www.cs.wisc.edu/condor/dagman/
- Design:
  * BOOK: "Designing Interfaces", by Jenifer Tidwell, Publisher: O'Reilly, ISBN: 0-596-00803-1
  * BOOK: "Designing Web Usability", by Jakob Nielsen, Publisher: New Riders. ISBN: 1-56205-810-X
  * BOOK: "The Non-Designer's Design Book, Second Edition: Design and Typographic Principles for the Visual Novice", by Robin Williams, Publisher: Peachpit Press, ISBN: 0-321-19385-7
  * BOOK: "Proven Portals: Best Practices for Planning, Designing, and Developing Enterprise Portals", by Dan Sullivan, Publisher: Addison Wesley Professional, ISBN: 0-321-12520-7
  * BOOK: "Information Architecture for the World Wide Web", 2nd Edition, by Peter Morville, Louis Rosenfeld, Publisher: O'Reilly, ISBN: 0-596-00035-9
- DMTF: http://www.dmtf.org/home

- GAMA: http://grid-devel.sdsc.edu/gridsphere/gridsphere?cid=gama
- GAS: http://www.gridlab.org/WorkPackages/wp-6/
- GCE-RG: https://forge.gridforum.org/projects/gce-rg/
- GEONgrid: http://www.geongrid.org/
- GGF: http://www.ggf.org
- GGF-WGs: http://www.ggf.org
- GPIR: http://www.tacc.utexas.edu/projects/gpir.php
- GRMS: http://www.gridlab.org/WorkPackages/wp-9/
- GrADS: http://www.hipersoft.rice.edu/grads/
- Gemstone project: http://gemstone.mozdev.org/
- GridAnt: http://www.cogkit.org/
- Gridbus: http://www.gridbus.org/
- Gridlabs: http://www.gridlab.org/
- Globus Toolkit 2 (GT2): http://www.globus.org/toolkit/downloads/2.4.3/
- Globus Toolkit 3 (GT3): http://www.globus.org/toolkit/downloads/3.0.2/
- Globus Toolkit 4 (GT4): http://www.globus.org/toolkit/downloads/4.0.2/
- GRaDS: http://www.iges.org/grads/
- GridFlow: http://www.dcs.warwick.ac.uk/research/hpsg/workflow/workflow.html
- GridFTP: http://www.globus.org/grid_software/data/gridftp.php
- GridPort: http://www.gridport.net
- GridShib: http://gridshib.globus.org/
- Gridsphere: http://www.gridsphere.org/
- HPC-Europa: http://www.hpc-europa.org/
- HTTP-R: http://www-128.ibm.com/developerworks/webservices/library/ws-phtt/
- IBM Websphere: http://www.ibm.com/websphere
- ICENI: http://www.lesc.ic.ac.uk/iceni/
- InfoGlue: http://www.infoglue.org/
- IntelliGrid: http://www.intelligrid.net/ (the URL doesn't seem to work)
- jPortlet: http://jportlet.sourceforge.net/
- J2EE: http://java.sun.com/javaee/
- JBOSS: http://labs.jboss.com/portal
- JCP: http://jcp.org/
- Joomla: http://www.joomla.org/
- JSDL: http://jsdl.sourceforge.net/
- Java Server Faces (JSF): http://java.sun.com/javaee/javaserverfaces/
- JSRs: http://jcp.org/en/jsr/all
- JSR 168: http://jcp.org/en/jsr/detail?id=168
- JSR 286: http://jcp.org/en/jsr/detail?id=286
- Karajan: http://www.cogkit.org/release/4_0_a1/manual/workflow/workflow.html
- Kepler: http://kepler-project.org/
- Kosmos: http://labs.jboss.com/portal/kosmos/
- LEAD: http://leadproject.org
- Liferay: http://www.liferay.com/web/guest/home
- Mambo: http://www.mamboserver.com/
- Midgard: http://www.midgard-project.org/
- MyGrid Bioinformatics portal: http://www.mygrid.org.uk/
- MyProxy: http://grid.ncsa.uiuc.edu/myproxy/
- .NET: http://www.microsoft.com/net/default.mspx
- NEESGrid portal: http://www.neesgrid.org/
- NMI: http://www.nsf-middleware.org/
- NPACI: http://www.npaci.edu/ (archived website)
- NPACI HotPage: https://hotpage.npaci.edu/
- Novell exteND: http://www.novell.com/products/extend/enterprise.html

- OASIS: http://www.oasis-open.org/
- OGCE: http://www.ogce.org/
- OGSA: http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf
- OGSA-DAI: http://www.ogsadai.org.uk/
- OGSI: http://www-128.ibm.com/developerworks/grid/library/gr-ogsi/
  http://xml.coverpages.org/OGSI-SpecificationV110.pdf
- OMII: http://www.omii.ac.uk/
- Oracle AS: http://www.oracle.com/appserver/index.html
- ORNL NS Gateway: http://www.gridforum.org/GGF14/GGF14_SGW_WS_Cobb_v01.ppt
- OKI: http://www.okiproject.org/
- OKI OSIDs: http://okicommunity.mit.edu/filemgmt/viewcat.php?cid=24
- OpenPortal: http://sourceforge.net/projects/openportal/
- Pegasus: http://pegasus.isi.edu/
- Plumtree: http://www.plumtree.com/
- Portals and Portlets 2006 workshop PPTs:
  http://www.nesc.ac.uk/action/esi/contribution.cfm?Title=686
- Portlet: http://en.wikipedia.org/wiki/Portlet
- Portlet Bridges: http://www.nesc.ac.uk/action/esi/download.cfm?index=3239
- Purdue NanoHub: http://www.nanohub.org/
- RFT: http://www-unix.globus.org/toolkit/docs/3.2/rft/index.html
- RENCI Bioportal: http://www.tgbioportal.org/
- RSF: http://www2.caret.cam.ac.uk/rsfwiki/
- SAML: http://en.wikipedia.org/wiki/SAML
- SAP E-portal: http://h71028.www7.hp.com/enterprise/cache/3957-0-0-0-121.html
- SCOOP: http://scoop.lsu.edu/gridsphere
- Servlet: http://en.wikipedia.org/wiki/Servlet
- ServoGrid portal: http://www.servogrid.org/
- Sharepoint: http://office.microsoft.com/en-us/FX010909721033.aspx
- SOA: http://en.wikipedia.org/wiki/Service-Oriented_Architecture
- SOAP: http://en.wikipedia.org/wiki/SOAP
- Storage Resource Broker (SRB): http://www.sdsc.edu/srb/index.php/Main_Page
- Stringbeans: http://www.nabh.com/projects/sbportal
- Sun Java ES portal server: http://www.sun.com/software/products/portal_srvr/
- SURA: http://www.sura.org/
- Taverna: http://taverna.sourceforge.net/
- TeraGrid portal: http://www.teragrid.org/
- TeraGrid User portal: https://portal.teragrid.org/
- Tibco Portalbuilder: http://www.tibco.com/software/portal/portalbuilder.jsp
- Triana: http://www.trianacode.org/
- Tupelo: http://dlt.ncsa.uiuc.edu/wiki/index.php/Main_Page
- UDDI: http://www.uddi.org/
- UDDI Business Registry:
  http://www.microsoft.com/windowsserver2003/technologies/idm/uddi/default.mspx
- Unicore: http://www.unicore.org/
- uPortal: http://www.uportal.org/
- UT Flood Modeling portal:
  http://www.tacc.utexas.edu/research/users/features/floodmodeling.php
- Vignette Portal Server: http://www.vignette.com/
- Vine project: http://www.nesc.ac.uk/action/esi/download.cfm?index=3237
- W3C: http://www.w3c.org/
- WBEM: http://www.dmtf.org/standards/wbem/
- WS: http://en.wikipedia.org/wiki/Web_services
  * BOOK: "Web Services Essentials", by Ethan Cerami, Publisher: O'Reilly, ISBN: 0-596-00224-6

- WS-Addressing:
  http://www-128.ibm.com/developerworks/library/specification/ws-add/
- WS-I: http://www.ws-i.org/
- WS-Notification:
  http://www-128.ibm.com/developerworks/library/specification/ws-notification/
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn
- WS-Reliability:
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm
- WS-Security: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- WSDL: http://en.wikipedia.org/wiki/WSDL
  http://www.w3.org/TR/wsdl
- WSDM: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
- WSRF: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
  http://en.wikipedia.org/wiki/Web_Services_Resource_Framework
- WSRP: http://en.wikipedia.org/wiki/WSRP
- WSRP 1.0: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
- WSRP 2.0: http://www.oasis-open.org/committees/download.php/18617/wsrp-2.0-spec-pr-01.html
- WSRP4J: http://portals.apache.org/wsrp4j/
- X.509: http://en.wikipedia.org/wiki/X.509
- Xaraya: http://www.xaraya.com/
- XCAT: http://extreme.cs.indiana.edu/xcat

22. Postscript

This document was to be a few page summary of grid portals for a tutorial I had planned for folks here at IU. By the time I realized I was beyond ten pages already (a typical problem when you type in text using vi) it was too late to stop. The result is this whopping tome (in ASCII at least).

As a portal newbie, I'd tried to find something (like this) to get a thirty thousand foot view of grid portals, but days of Googling produced only a huge number of disparate documents and links (which I fortunately bookmarked). Since I cannot remember stuff well, I aggregated what I found and the result was this document. However, it seemed that it'd never end until I simply gave up trying to define everything I'd found (just the WS-this and WS-that part gave me a headache!).

Please feel free to report any errors (that are sure to be in here because I am not an expert). Also, please pardon the occasional, IU-centric content.

Thank you.

Anurag Shankar <ashankar@indiana.edu>, Ph.D.
Dept. of Astronomy & Univ. IT Services, Indiana University
2711 E. 10th St., Bloomington, IN 47408
Voice: +1.812.855.4981 Fax: +1.812.855.8299

$Id: portals-101.doc,v 1.7 2006/11/02 19:54:19 ashankar Exp $