The Grid Portal Development Kit

Jason Novotny[1] (jdnovotny@lbl.gov)

Abstract

Computational science portals are emerging as useful and necessary interfaces for performing operations on the Grid. The Grid Portal Development Kit (GPDK) facilitates the development of Grid portals and provides several key reusable components for accessing various Grid services. A Grid Portal provides a customizable interface allowing scientists to perform a variety of Grid operations including remote program submission, file staging, and querying of information services from a single, secure gateway.

The Grid Portal Development Kit leverages off existing Globus/Grid middleware infrastructure as well as commodity web technology including Java Server Pages and servlets. We present the design and architecture of GPDK as well as a discussion on the portal building capabilities of GPDK allowing application developers to build customized portals more effectively by reusing common core services provided by GPDK.

1. Introduction

Computational Grids have emerged as a distributed computing infrastructure for providing pervasive, ubiquitous access to a diverse set of resources ranging from high-performance computers (HPC) to tertiary storage systems to large-scale visualization systems to expensive and unique instruments including telescopes and accelerators. One of the primary motivations for building Grids is to enable large-scale scientific research projects to better utilize distributed, heterogeneous resources to solve a particular problem or set of problems. However, Grid infrastructure only provides a common set of services and capabilities that are deployed across resources and it is the responsibility of the application scientist to devise methods and approaches for accessing Grid services. For this reason, higher-level tools often in the form of problem solving environments (PSE) are designed for specific application areas to more effectively take advantage of Grid infrastructure. Another difficulty in developing higher-level tools is the deployment and

[1] Lawrence Berkeley National Laboratory

accessibility of application specific PSE's. Scientists and researchers are often required to download and install specialized software libraries and packages. For this reason, while PSE's are capable of providing the most direct and specialized access to Grid resources, we consider the web browser itself to be a widely available and generic problem solving environment when used in conjunction with a Grid portal. We define a Grid portal to be a web based application server enhanced with the necessary software to communicate to Grid services and resources. A Grid portal provides application scientists with a customized view of software and hardware resources specific to their particular problem domain and provides a single point of access to Grid resources they have already been authorized to use.

The concept of developing a web enabled gateway to the Grid has been employed by several research projects including the HotPage user portal [], the Gateway project [], and UNICORE []. While the technologies and design of these projects differ, they share similar goals in trying to provide easy access to Grid resources and performing various Grid operations including remote job submission and providing detailed information on resources allowing the user or a super-scheduler to make better informed scheduling decisions.

The Grid Portal Development Kit addresses many of the same issues related to providing secure, web-based access to resources as the previous projects, but differs in three important ways. First, the core of GPDK resides in a set of generic, reusable, common components used for accessing the various Grid services that are supported by the Globus toolkit []. Second, a portal user is provided with a persistent, customizable profile that contains information that is stored securely on the portal and provides details on past jobs submitted, the set of computers they have access to, and any other information that is of interest to a particular user. Third, GPDK is designed to provide a complete development environment for building customized application specific portals that can take advantage of the core set of GPDK Grid service components. In this paper we discuss the design and architecture of the Grid Development Kit with an emphasis on implementation and the technologies used. We also discuss the advanced portal development capabilities of the Grid Portal Development Kit followed by a discussion of related work and future directions.

2. Overview of the Grid Portal Development Kit

The Grid Portal Development Kit is based on the standard n-tier architecture adopted by most web application servers as shown in Figure 1. Tiers represent physical and administrative boundaries between the end user and the web application server. The client tier is represented as tier 1 and consists of the end-user's workstation running a web browser. The only requirements placed upon the client tier is a secure (SSL-capable) web browser that supports DHTML/Javascript for improved interactivity, and cookies to allow session data to be transferred between the client and the web application server. We discuss cookies in more detail within the context of user profiles in section 2.1.

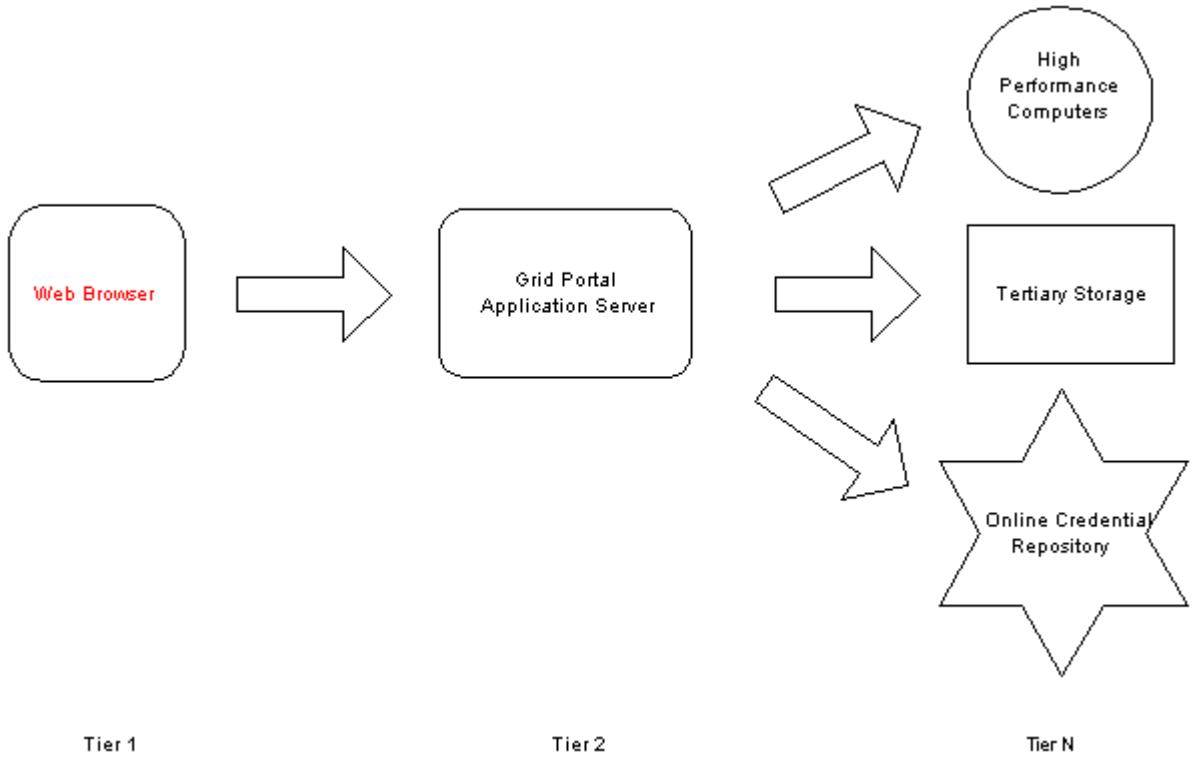

Figure 1: N-tier Architecture

The second tier is the web application server and is responsible for handling HTTP requests from the client browser. The application server is necessarily multi-threaded and must be able to support

multiple and simultaneous connections from one or more client browsers. A large part of the design of the Grid Portal Development Kit is based on providing multi-user access to Grid services and resources.

All other resources accessed by the portal including any databases used for storing user profiles or additional information forms the third tier, known as the back-end. Back-end resources are generally under separate administrative control from the web application server and subject to different policies and use conditions.

Figure 1. Standard 3-tier web architecture

2.1 Grid Portal Architecture

The core of the middle-tier application server is a secure web server capable of handling HTTP/S requests from client browsers and serving up static HTML pages to the client. The portal is comprised of a vertical layer of software components as shown in figure 2.

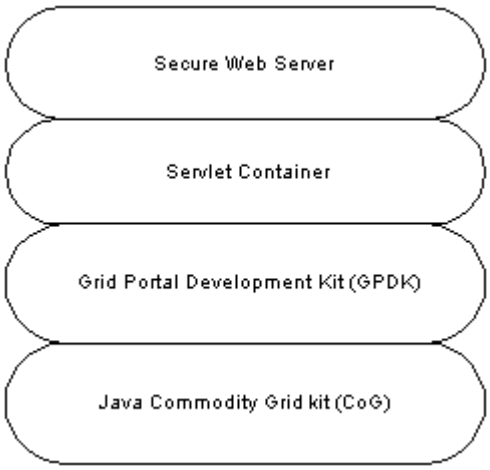

Figure 2. Portal Architecture (Web server, servlet container, GPDK, COG)

The modular and flexible design of the GPDK core services led to the adoption of a servlet container for handling more complex requests versus the traditional approach of invoking CGI scripts for performing portal operations. In brief, a servlet is a Java class that implements methods for handling HTTP protocol requests in the form of GET and POST. Based on the request, a servlet can be used as a controller to forward requests to either another servlet or a Java Server Page (JSP). A key design decision in developing GPDK was to adopt the standard Model-View-Controller

(MVC) design pattern common in many Java classes, notably the Java Swing GUI library. The MVC design allows for the natural separation of control and presentation from what is termed "business logic" or the actual operational code that must be executed as part of a portal operation, or action. Java Server Pages provides a scripting language using Java within an HTML page that allows for the instantiation of Java objects, also known as beans. The result is the dynamic display of data created by a Java Server Page that is compiled into HTML on the fly.

2.2 GPDK Architecture

The Grid Portal Development Kit has been developed using the Tomcat open source servlet container available from Sun. Although several other commercial servlet containers exist, Tomcat was chosen as it implements the latest JSP and Servlet specifications from Sun and is included as part of the Java Enterprise Edition (J2EE) production application server.

During installation, the Grid Portal Development Kit deploys the library of core service beans as well as a central servlet and a collection of fully functional demo template web pages to the Tomcat servlet container. The template web pages include HTML and Java Server Pages intended to demonstrate the GPDK core service beans.

The central servlet is at the heart of the Portal and is responsible for handling all incoming client requests to the portal. Figure 3 shows the sequence of events associated with performing a particular portal action. Upon startup, the servlet performs several key initialization steps including the instantiation of a BasicPortal object used to initialize and destroy resources that are used during the operation of the portal. The BasicPortal object is responsible for initializing a Logger that is used by core GPDK service beans as well as loading static information about known resources by querying the Grid information server specified in the GPDK configuration file. In addition, the servlet instantiates Page Objects (PO) as specified by the pages configuration file. Based on the value of the "action" parameter in the HTTP request header, an appropriate Page Object is executed as shown in Step 2. Page objects are responsible for performing the logic of a particular portal operation and use the GPDK service beans to execute the required operation. As shown in Step 3, the Servlet forwards control to a Java Server Page, known as a "view page" after the PO is executed. The view page has a minimal amount of Java code and is typically used to display the results of the Page objects calculation.

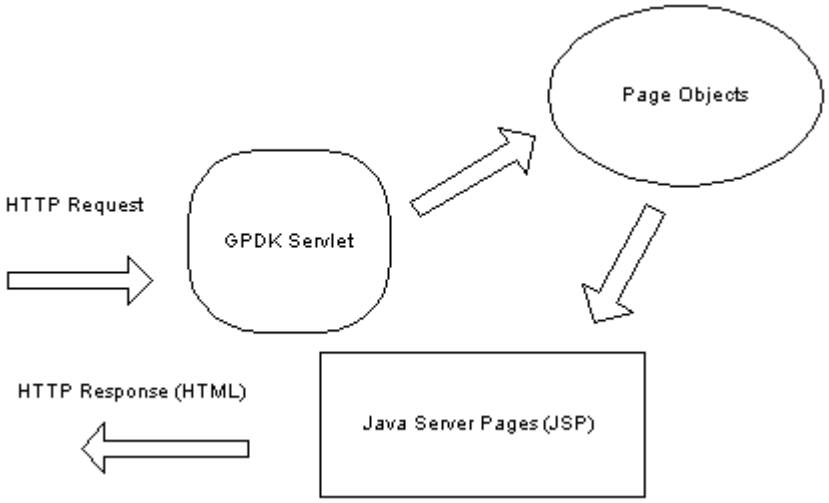

Figure 3: The Portal Event Cycle (Servlet -> Page object -> JSP)

2.3 GPDK Service Beans

Grid service beans are typically implemented using the Java Commodity Grid (CoG) toolkit, which provides a pure Java API to Globus services. One of the powerful features of Java beans in the context of web application servers is bean scope. Bean scope refers to the level of persistence offered by a bean within the servlet container. For instance, beans may have session, application, or request scope. Session scope implies that the bean persists for the duration of a user's session, typically determined by the servlet container. For instance user profiles are represented as session beans and persist until a user decides to log out of the portal or their session times out as determined by the servlet container. A bean with application scope persists for the complete duration of the servlet container and provides a persistent object used to store application specific static data that can be referenced by any Java Server Page on behalf of any user. The addition of collaborative capabilities such as a whiteboard or chat room, for instance, requires that messages be maintained with application scope and are visible to all users logged in. Beans with request scope persist only for the duration of a client HTTP request and are then destroyed after the JSP page is processed into an HTML response for the client. As discussed in the following sections, the GPDK service beans are organized according to Grid services in the areas of security, job submission, file transfer and information services.

2.3.1 Security

The security working group of Grid Forum has been actively promoting the Grid Security Infrastructure (GSI) [] as the current best practice for securely accessing Grid services. GSI is based upon public key infrastructure (PKI) and requires users to possess a private key and an X.509 certificate used to authenticate to Grid resources and services. A key feature of GSI is the ability to perform *delegation*, the creation of a temporary private key and certificate pair known as a *proxy* that is used to authenticate to Grid resources on a user's behalf. The GSI has been implemented over the Secure Sockets Layer (SSL) and is incorporated in the Globus and Java CoG toolkit.

One of the key difficulties in developing a portal to access Grid services is providing a mechanism for users to delegate their credentials to the portal since current web browsers and servers do not support the concept of delegation. Past solutions have involved the storage of users' long-lived keys and certificates on the portal. A user would then provide their long-term pass phrase to the portal, which creates a valid proxy that can be used on the user's behalf. The danger in this approach, however, is the risk of the web server being broken into and having possibly many users' long term private keys compromised. For this reason, the Myproxy service [] was developed to provide an online certificate repository where users can delegate temporary credentials that can be retrieved securely by the user from the portal. Briefly, a user delegates a credential to the Myproxy server with a chosen lifetime and username and pass phrase. The user would enter the same username and pass phrase from their browser over an HTTPS connection and the portal would retrieve a newly delegated credential valid for a chosen amount of time. Currently, GPDK doesn't enforce any maximum lifetime for the credential delegated to the portal, but when a user logs off, the proxy is destroyed reducing any potential security risk of their delegated credential being compromised on the portal. The portal retrieves credentials from the Myproxy Server using the GPDK security component, the MyproxyBean. The MyproxyBean component is actually a wrapper around the CoG toolkit client API to the Myproxy server.

Within the demo portal that comes with GPDK, authorization to the portal itself is based on whether the portal can successfully retrieve a delegated credential based on the users supplied name and pass phrase. However, it's easy to imagine that authorization could also be determined by a

name and password file on the portal or a back-end database, if the portal administrator wished to further restrict access to the portal.

2.3.2 Job Submission

Both interactive and batch queue job submissions are enabled using either the GSI enhance SSH client [] or using the Globus GRAM protocol to submit jobs to Globus gatekeepers deployed on Grid resources. The major GPDK components used to submit jobs are the JobBean, the JobSubmissionBean and the JobInfoBean. The JobBean provides a description of the job to be submitted. It includes methods for setting and returning values for the executable, additional arguments passed to the executable, number of processors for parallel jobs, batch queue if submitting a batch mode and more. The JobSubmissionBean is actually an abstract class that is subclassed by the GramSubmissionBean in the case of submitting a job to a Globus gatekeeper or a GSISSHSubmissionBean idf using the GSI enhanced SSH client. The GramSubmissionBean capabilities are provided once again by the Java CoG library. Once a job has been succesfully submitted, a JobInfoBean is created which contains a time stamp of when the job was submitted and other useful information about the job, including a GRAM URL that can be used to query on the status of the job.

2.3.3 File Transfer

Data access capabilities are provided by the GridFTP [] API implemented as part of the CoG toolkit and encapsulated into core GPDK service beans. Capabilities include file transfer, including third-party file transfer between GSI enabled FTP servers, as well as file browsing capabilities. The FileTransferBean provides a generic file transfer API that is extended by the GSIFTPTransferBean and the GSISCPTransferBean, an encapsulation of file transfer via the GSI enhanced scp coommand tool. The GSIFTPServiceBean provides a session scoped bean that manages multiple FTP connections to GSI enabled FTP servers. The GSIFTPServiceBean allows users to browse multiple GSI FTP servers simultaneously and a separate thread monitors server timeouts. The GSIFTPViewBean is an example view bean used by a JSP to display the results of browsing a remote GSI FTP server.

2.3.4 Information Services

The Grid Forum Information Services working group has proposed the Grid Information Services (GIS) architecture for deploying information services on the Grid and supported the Lightweight Directory Access Protocol (LDAP) as the communication protocol used to query information services. Information services on the Grid are useful for obtaining both static and dynamic information on software and hardware resources. The Globus toolkit provides a Metacomputing Directory Service (MDS), which is an implementation of a Grid Information Service using OpenLDAP, an open source LDAP server. Although, the Java CoG toolkit provides support for LDAP using the Java Naming and Directory Interface (JNDI), GPDK uses the open source Netscape/Mozilla Directory SDK [] as it proved easier to use in practice and also provides support for developing a connection pool for maintaining multiple connections to several Grid Information service providers, thus eliminating the need for clients to reconnect during each query. However, this model will need to be re-evaluated with the widespread deployment of the MDS-2 architecture which includes GSI enhancements making it necessary for clients to reauthenticate to the MDS for each query. GPDK provides an MDSQueryBean and MDSResultsBean for querying and formatting results obtained from the MDS. Currently GDK supports querying the MDS for hardware information such as CPU type, number of processors and other details as well as cpu load and queue information that can be used by the user to make more effective job scheduling decisions.

2.4 GPDK User Profiles

User profiles are an integral part of the portal architecture as they enable the customization of a particular set of resources (hardware and software) by portal users. In addition, user profiles create a "value-added" feature of using the portal to perform common Grid operations since a user history is maintained in a user profile allowing users to keep track of past jobs submitted and results obtained. A user profile is a persistent session bean that is either created the first time a user logs into the portal or is loaded (de-serialized) from a file or database if one exists already. User profiles are serialized when a user logs out of the portal or the profile has been modified.

3. GPDK as a portal development environment

It is envisioned that many computational science projects can benefit by providing web access to a particular communities set of codes, data and expertise. The Grid Portal Development Kit is designed to be a development environment that enables rapid development of application specific portals by leveraging off core GPDK service beans and the MVC architectural model. The core set of services remain generic enough to be useful to any community interested in staging data, executing codes, and providing a customizable, commonly accesible, collaborative environment.

The Grid Portal Development Kit relies on ANT for its build process. ANT provides an XML based build tool that is very similar to using Makefiles under Unix. Not only does ANT simplify the compilation and deployment operations necessary for portal development, but it provides a cross-platform build environment for the development of portals using GPDK and provides support for common operations such as invoking the java compiler, the java archiver tool used for creating libraries, the javadoc tool used for creating API documentation from commented source code, as well as file copying, and directory creation/deletion operations. ANT also provides support for regular expression variable substitution within files making it easy for GPDK to create application specific portals by performing substitution on the GPDK template files.

The GPDK template files are organized according to project specific source code and web pages in the form of HTML and Java Server Pages (JSP). The template source code contains the central servlet class used by the application specific portal as well as subclasses for project specific user profiles and a project specific BasicPortal class used to initialize and shutdown any resources required by the new project portal.

The GPDK template source code also contains all the Page objects necessary for performing the various portal operations that demonstrate core GPDK services and beans. The Page objects are composed of a LoginPage, UpdateProfilePage and LogoutPage that demonstrate the retrieval of credentials from the Myproxy server and the loading, editing and saving of user profiles. The JobSubmissionPage, FileTransferPage demonstrate the GPDK service beans in the areas of job submission and file transfer. The template Java Server Pages are displayed upon the successful completion of the Page object operations. If an error occurs during the execution of a Page object, a PageException object is thrown with a relevant error message that is displayed by a specialized error JSP.

Initially when GPDK is installed, only the core service beans are compiled and a library in the form of a Java Archive Repository (JAR) is created. By specifying a new project target when invoking the GPDK build script e.g. “build new CoolPortal”, a new portal subproject is created containing the preprocessed template portal pages. The new portal subproject contains documentation and the preprocessed template files. GPDK uses the substitution capabilities provided by ANT to properly preprocess the template files into working project specific source code and web pages. Only two variable substitutions are performed on the template code. The PROPER_NAME variable used within source code and JSP’s refers to the application specific portal name e.g. CoolPortal in the example above. The template UserProfileBean class is preprocessed to create a project specifc user profile bean e.g. CoolUserProfileBean from the above example that extends from the UserProfileBean class and is used by the newly created portal to extend a user profile to include project specific data about a user. Similarly, the BasicPortal template class that subclasses the GPDK BasicPortal object is used to create a project specific Logger and handles any additional initialization and shutdown routines required by the new portal project. The other variable present in template source code that is substituted is the PROPER_NAME variable. The PROPER_NAME variable is simply the lowercase name of the PROPER_NAME and helps to define a namespace for the portal project under which the new project classes are compiled into a library. For instance, the naming convention of GPDK core service beans is org.gpdk following the Java package naming convention, while the template Page objects and other project specific classes are org.coolportal in the case of the CoolPortal example. This logical separation of core GPDK service beans and project specific classes allows for the creation and deployment of multiple portal projects to the same application server. Not only are template source code files preprocessed accordingly, but the creation of project specific documentation and web pages is enabled through the use of preprocessing as well. The net result is an easy to use build environment that allows for the rapid deployment of an application specific portal that offers full functionality and requires only the additional creation of project specific Page objects and some tweaking of the web pages to create a customized display.

5. Related Work

The Grid Portal Development Kit has proven successful in the development of application specific portals being developed by several other groups. The following list briefly describes various ongoing portal projects that have been developed using the Grid Portal Development Kit framework:

The NASA Launchpad user portal seeks to provide web based access to users of the NASA Information Power Grid (IPG) []. Launchpad is based entirely on GPDK and takes advantage of the GPDK service beans to allow IPG users access to high-performance computer resources and IPG Grid Information Services.

The NASA Nebula portal provides a web-based interface to the Nebula simulation code. Scientists use the Nebula code to study the atomic composition of interstellar clouds. The Nebula portal allows researchers to create an appropriate list of input parameters to the Nebula simulation and submit their job to NASA IPG resources.

The Astrophysics Simulation Collaboratory Portal (ASC) [] is an application portal developed to provide astrophysics researchers web based access to the Cactus computational toolkit. Cactus is a framework for solving various wave equations with an emphasis on astrophysical simulations of colliding neutron stars and black holes. The ASC portal allows users to submit various Cactus simulations to a distributed set of HPC resources as well as remotely checkout and compile Cactus with specialized options on HPC resources.

The NCSA portal development effort is focused on the development of several application portals for computational chemistry and particle physics simulations. GridGaussian, a computational chemistry portal allows users to provide input files to the popular Gaussian chemistry package and easily submit simulations via the web. Similarly, the MIMD Lattice Computation (MILC) portal is aimed at providing particle physicists access to a popular community code used to understand

elementary particle interactions. Both portals rely on GPDK for job submission and file staging capabilities.

6. Conclusions and Future Work

We have described a portal architecture and a framework for the development of application specific portals using the Grid Portal Development Kit. Based on the Model View Controller design paradigm, the GPDK has proven to be an extensible and flexible software package in use by several other portal building efforts. The GPDK integrates nicely with other commodity technologies including the open-source servlet container Tomcat and the Apache web server. The Grid Portal Development Kit makes use of the Java Commodity Grid (CoG) toolkit for its pure Java implementation of client side Globus Grid services. In addition, the GPDK uses commodity libraries from Netscape to provide access to Grid information services using LDAP.

Future work on the Grid Portal Development Kit includes enhancing capabilities of existing Grid service beans. In the area of job submission, for instance, it would be nice to notify users via email when a job completes. Currently, a user must log into the portal and explicitly check the past job submitted to obtain the status. However, a JobMonitor object could be created as a thread to monitor submitted jobs and respond with an email when a job completes or fails. In addition, the user profile could be updated to reflect the new job status.

Another area of future development is in task composition. Many portals would like to specify Grid or portal operations as tasks and be able to combine tasks together to create a work flow system for an entire calculation involving staging data, running a simulation and migrating output data to a storage system. We're currently working with the GPDK user community to develop a task interface that will address these needs. Other future developments include the integration of a database to store user profiles rather than maintaining them on the portal. Additional application information may also be stored in the database. Using LDAP as the protocol to access the database would enable code reuse of the GPDK information service beans, but ODBC could be used as well.

7. Acknowledgements

We are grateful to many colleagues for discussions on portal development and working with early incarnations of GPDK to improve on its usefulness and robustness. In particular we wish to thank Jarek Gawor, Gregor Laszewski, Nell Rehn, George Myers, Mark Wallace, Farah Hasnat, Yinsyi Hung, John Lehman, Jeffrey Becker, Michael Russell, Shawn Hampton and Scott Koranda.

8. References

[1] Netscape Directory and LDAP Developer Central. Http://developer.netscape.com/tech/directory/index.html
[2] Java Servlet and Java Server Pages Specifications http://java.sun.com/products/servlets
[3] Tomcat open-source servlet container http://jakarta.apache.org/tomcat
[4] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Tuecke. GridFTP: Protocol Extensions to FTP for the Grid. Grid Forum Working Draft, March 2001. http://www.gridforum.org
[5] G. Allen, M. Russell, J. Novotny, G. Daues, J. Shalf. The Astrophysics Simulation Collaboratory: A Science Portal Enabling Community Software Development. To appear in P*roc. 10th IEEE Symp. On High Performance Distributed Computing*, 2001.
[6] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Grid Information Services for Distributed Resource Sharing. To appear in P*roc. 10th IEEE Symp. On High Performance Distributed Computing*, 2001.
[7] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture
[8] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High-Performance Distributed Computations. In P*roc. 6th IEEE Symp. On High Performance Distributed Computing*, p 365-375, 1997.
[9] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Pub. August 1998. ISBN 1-55860-475-8
[10] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*. 11(2):115-128, 1997
[11] Foster, I., Karonis, N.T., Kesselman, C., Koenig, G. and Tuecke, S. A Secure Communications Infrastructure for High-Performance Distributed Computing. in *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, 1997, 125--136.
[12] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object Oriented Software.* Reading, Massachusetts, 1995.
[13] T. Haupt, E. Akarsu, G. Fox, C. Youn, "The Gateway System: Uniform Web Based Access to Remote Resources" *Concurrency: Practice and Experience*, 12(8):629-642, July 2000.

[14] W. E. Johnston, D. Gannon and B. Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. In P*roc. 8th IEEE Symp. On High Performance Distributed Computing*, 1999.
[15] G. Laszewski, I. Foster, J. Gawor, "CoG Kits: A Bridge Between Commodity Distributed Computing and High-Performance Grids", *Proceedings of the ACM Java Grande Conference*, June 2000.
[16] J. Novotny, S. Tuecke, V. Welch. An Online Credential Repository for the Grid: MyProxy. To appear in P*roc. 10th IEEE Symp. On High Performance Distributed Computing*, 2001.
[17] M. Romberg, "The UNICORE Architecture", Proc. of the 8th IEEE Intl. Symp. on High Perf. Dist. Comp, August, '99.
[18] M. Thomas, S. Mock, J. Boisseau, "Development of Web Toolkits for Computational Science Portals: The NPACI HotPage" Proc. of the 9th IEEE Intl. Symp. on High Perf. Dist. Comp, August, '00.