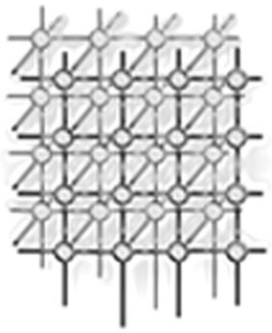
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2002; **14**:1129–1144 (DOI: 10.1002/cpe.669)

The Grid Portal Development Kit



Jason Novotny*,†

Lawrence Berkeley National Laboratory, Berkeley, CA 94704, U.S.A.

SUMMARY

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 GPDK leverages off existing Globus/Grid middleware infrastructure as well as commodity Web technology including Java Server Pages and servlets. The design and architecture of the GPDK is presented as well as a discussion on the portal building capabilities of the GPDK, allowing application developers to build customized portals more effectively by reusing common core services provided by the GPDK. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: Grid computing; computing environments; Grid portals; science portals; Globus

1. INTRODUCTION

Computational Grids [1] have emerged as a distributed computing infrastructure for providing pervasive, ubiquitous access to a diverse set of resources including high-performance computers (HPCs), tertiary storage systems, large-scale visualization systems, 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.

Unfortunately, it still remains a daunting task for an application scientist to easily 'plug into' the computational Grid. While command line tools exist for performing atomic Grid operations, a truly usable interface requires the development of a customized problem solving environment (PSE).

*Correspondence to: Jason Novotny, Lawrence Berkeley National Laboratory, Berkeley, CA 94704, U.S.A.
†E-mail: jdnovotny@lbl.gov

Received 29 June 2001
Revised 6 January 2002

Traditionally, specialized PSEs were developed in the form of higher-level client side tools that encapsulate a variety of distributed Grid operations such as transferring data, executing simulations and post-processing or visualization of data across heterogeneous resources. A primary barrier in the widespread acceptance of monolithic client side tools is the deployment and configuration of specialized software. Scientists and researchers are often required to download and install specialized software libraries and packages. Although client tools are capable of providing the most direct and specialized access to Grid enabled resources, we consider the Web browser itself to be a widely available and generic PSE when used in conjunction with a Grid portal. A Grid portal is defined as 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 from a Web browser.

Furthermore, Grid portals can be subdivided into application-specific and user-specific portal categories. An application specific portal provides a specialized subset of Grid operations within a specific application domain. Examples of application specific portals include the Astrophysics Simulation Collaboratory [2] and the Diesel Combustion Collaboratory [3]. User portals generally provide site specific services for a particular community or research center. The HotPage user portal [4], the Gateway project [5], and UNICORE [6] are all examples of user portals that allow researchers to seamlessly exploit Grid services via a browser-based view of a well-defined set of Grid resources.

The Grid Portal Development Kit (GPDK) [7] seeks to provide generic user and application portal capabilities and was designed with the following criteria.

- The core of the GPDK should reside in a set of generic, reusable, common components to access those Grid services that are supported by the Globus toolkit [8] including the Grid Security Infrastructure (GSI) [9]. As Globus [10] becomes a *de facto* standard for Grid middleware and gains support within the Global Grid Forum [11], the GPDK shall maintain Globus compatibility through the use of the Java Commodity Grid (CoG) kit [12]. An enumeration and description of the Grid services is provided in the next section.
- It should provide a customizable user profile that contains user specific information such as past jobs submitted, resource and application information, and any other information that is of interest to a particular user. GPDK user profiles are intended to be extensible, allowing for the easy creation of application portal specific profiles, as well as serializable, such that users' profiles are persistent even if the application server is shut down or crashes.
- It should provide a complete development environment for building customized application specific portals that can take advantage of the core set of GPDK Grid service components. The true usefulness of the GPDK is in the rapid development and deployment of specialized application or user portals intended to provide a base set of Grid operations for a particular scientific community. The GPDK shall provide both an extensible library and a template portal that can be easily extended to provide specialized capabilities.
- The GPDK should leverage commodity and open source software technologies to the highest degree possible. Technologies such as Java beans and servlets and widespread protocols such as HTTP and Lightweight Directory Access Protocol (LDAP) provide interoperability with many existing Internet applications and services. Software libraries used by the GPDK should be freely available and ideally provide open source implementations for both extensibility and for the widespread acceptance and adoption within the research community.

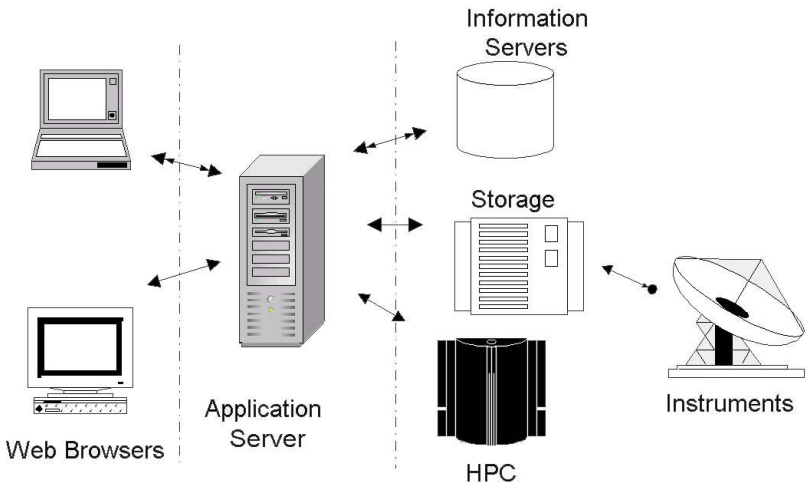


Figure 1. Standard three-tier Web architecture.

The following sections explain the design and architecture of the GPDK with an emphasis on implementation and the technologies used. The advanced portal development capabilities of the GPDK and future directions will also be discussed.

2. OVERVIEW OF THE GRID PORTAL DEVELOPMENT KIT

The GPDK is based on the standard three-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 (Secure Socket Layer (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.

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. The GPDK augments the application server with Grid enabling software and provides multi-user access to Grid resources. All other resources accessed by the portal including any databases used for storing user profiles, online credential repositories or additional resources form 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. The GPDK has been specially tailored to provide access

to Grid resources as the back-end resources. It is generally assumed that Grid resources understand a subset of defined Grid and Internet protocols.

3. GRID PORTAL ARCHITECTURE

The GPDK provides Grid enabling middleware for the middle tier and aids in providing a Grid enabled application server. The GPDK is part of a complex vertical software stack as shown in Figure 2. At the top of the stack is a secure high-performance Web server capable of handling multiple and simultaneous HTTPS requests. Beneath the Web server is an application server that provides generic object invocation capabilities and offers support for session management. The deployed GPDK template portal creates a Web application that is managed by the application server and provides the necessary components for accessing Grid services.

The GPDK uses the Model-View-Controller (MVC) design pattern [13] to separate control and presentation from the application logic required for invoking Grid services. The GPDK is composed of three core components that map to the MVC paradigm. The Portal Engine (PE) provides the control and central organization of the GPDK portal in the form of a Java servlet that forwards control to the Action Page Objects (APOs) and the View Pages (VPs). The APOs form the 'model' and provide encapsulated objects for performing various portal operations. The VPs are executed after the APOs and provide a user and application specific display (HTML) that is transmitted to the client's browser.

The Grid service beans form the foundation of the GPDK and are used directly by the PE, APOs, and VPs. The Grid service beans are reusable Java components that use lower-level Grid enabling middleware libraries to access Grid services. Each Grid service bean encapsulates some aspect of Grid technology including security, data transfer, access to information services, and resource management. Commodity technologies are used at the lowest level to access Grid resources. The Java CoG toolkit, as well as other commodity software Application Programming Interfaces (APIs) from Sun and Netscape, provide the necessary implementations of Grid services to communicate a subset of Grid protocols used by the GPDK service beans.

The modular and flexible design of the GPDK core services led to the adoption of a servlet container for handling more complex requests compared with the traditional approach of invoking individual 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, the GPDK servlet can be used as a controller to forward requests to either another servlet or a Java Server Page (JSP). JSPs provide 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 JSP that is compiled into HTML.

Figure 3 shows the sequence of events associated with performing a particular portal action. Upon start-up, the GPDK servlet performs several key initialization steps including the instantiation of a PE used to initialize and destroy resources that are used during the operation of the portal. The PE performs general portal functions including logging, job monitoring, and the initialization of the portal informational database used for maintaining hardware and software information. The PE is also responsible for authorizing users and managing users' credentials used to securely access Grid services. When a client sends an HTTP/HTTPS request to the application server, the GPDK servlet is responsible for invoking an appropriate action page (AP) based on the 'action value' received as part of the HTTP header information. The page lookup table is a plaintext configuration file that contains

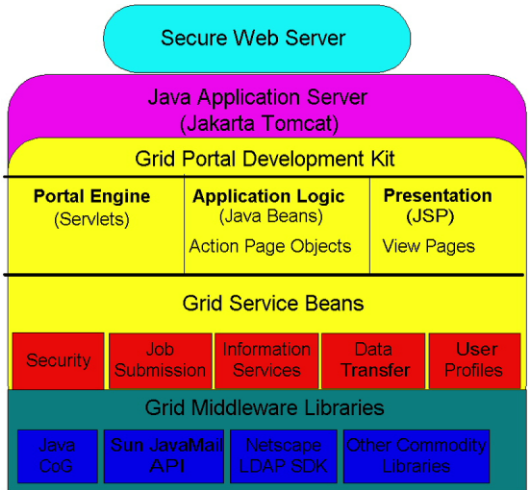


Figure 2. GPDK architecture and vertical stack of services and libraries.

mappings of 'action values' to the appropriate APOs and VPs. An AP is responsible for performing the logic of a particular portal operation and uses the GPDK service beans to execute the required operations. Finally, the GPDK servlet forwards control to a VP, a JSP, after the AP is executed. The VP formats the results of an AP into a layout that is compiled dynamically into HTML and displayed in a client's browser.

4. GPDK IMPLEMENTATION

While a Web server is unnecessary for the development of project specific portals using the GPDK, a secure Web server is needed for the production deployment of a GPDK-based portal. A production Web server should offer maximum flexibility including the configuration of the number of supported clients, network optimization parameters, as well as support for 56- or 128-bit key based SSL authentication and support for a Java application server. The GPDK has been successfully deployed using the Apache [14] Web server, a free, open source Web server that provides a high-level of scalability and SSL support using the modSSL [15] package.

As mentioned previously, the GPDK relies on commodity Java technologies including Java beans, servlets and JSPs for its general framework. The GPDK was developed under the open source Tomcat application server available from the Jakarta Apache project [16]. The Tomcat [17] application server was chosen as it is freely and widely available and implements the latest JSP and servlet specifications from Sun and is included as part of the Java Enterprise Edition (J2EE) production application server.

The Java CoG toolkit provides most of the functionality required to implement the various Grid services that have been encapsulated by the core GPDK service beans. The Java CoG kit was

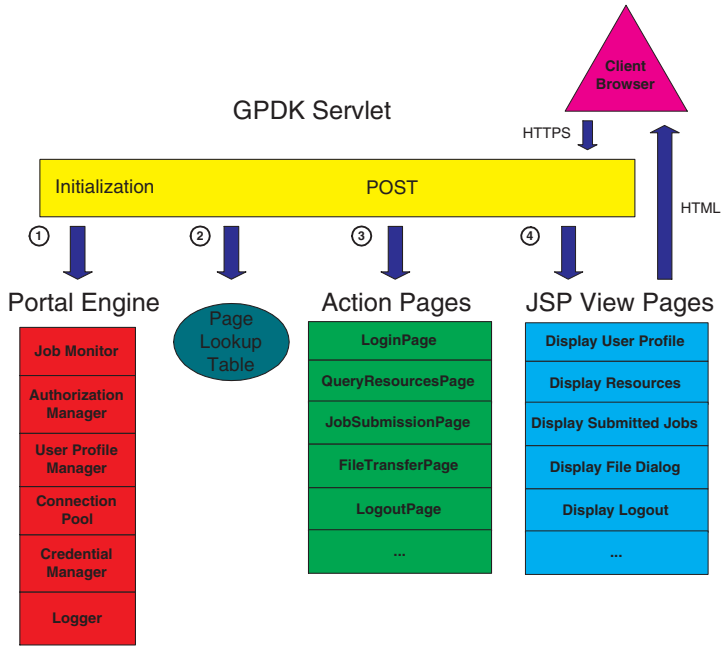


Figure 3. The GPDK event cycle.

developed to provide compliance with Globus Grid services and compatibility with the C reference implementation of Globus. The Java CoG toolkit provides the following Grid protocols and services that are used by the GPDK.

- The GSI provides a secure communication protocol that uses X.509 certificates and SSL to perform mutual authentication. The Java CoG toolkit implements GSI using the IAIK [18] security libraries. Although the IAIK libraries are proprietary and not open source, they remain free for research and academic use. Implementation of GSI using other Java security libraries such as Sun's Java Secure Sockets Extensions [19] is being investigated.
- The Globus Resource and Management (GRAM) [20] protocol is used to submit jobs to a Globus gatekeeper, a standard authentication and job spawning service provided by Globus enabled resources.
- The Grid FTP [21] protocol provides an optimized data transfer library and a specialized set of FTP commands for performing data channel authentication, third-party file transfers, and partial file transfers.
- The Myproxy [22] service provides an online credential repository for securely storing users' delegated credentials. The Java CoG provides a client API for communicating to a Myproxy certificate repository for retrieval of a user's security credentials.

One of the powerful features of Java beans in the context of Web application servers and the GPDK service beans 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. Session scoped beans rely on the use of cookies used by most Web browsers to retain state information on a particular client connection overcoming the inherent lack of state in the HTTP protocol. 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 JSP 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, so logged in clients can see others' messages. A bean with request scope persists only for the duration of a client HTTP request and is destroyed after the JSP page is processed into an HTML response for the client.

The GPDK has been developed and tested under Windows, Linux, and Solaris platforms using the various JDKs provided by Sun in conjunction with the Apache Web server available on both Windows and Unix platforms.

5. GPDK SERVICES

The usefulness of the GPDK as a portal development framework rests on the currently supported set of common Grid operations that are required for a typical scientific collaboration. A scientific collaboration may involve one or more of the following capabilities:

- submission, cancellation, and monitoring of specialized programs (serial and/or parallel) to a variety of computer resources, including those requiring batch (non-interactive) submission;
- the ability to store and retrieve data accumulated either from experiment or simulation to a variety of storage resources;
- the ability to use resource discovery mechanisms to enable the discovery of hardware and software resources that are available to a particular scientific collaboration;
- the ability to perform the above operations securely by allowing scientists to authenticate to remote resources as required by the remote site administrators;
- application specific profile information including user preferences, submitted jobs, files transferred, and other information that scientists may wish to archive, along with results obtained from computational simulations or laboratory experiments.

Within the GPDK framework, the above requirements have been encapsulated into one or more GPDK service beans. 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. The deployed GPDK demo portal highlights the capabilities of the supported Grid services through template Web and JSP pages.

5.1. Security

The security working group of Grid Forum has been actively promoting the GSI [23] 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 users behalf. The GSI has been implemented over the SSL and is incorporated in the Globus and Java CoG toolkits. 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 [22] 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 user name and pass phrase. The user would enter the same user name 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 does not 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. For users that have their delegated credential local to their workstation, the GPDK template portal allows them to upload the proxy to the portal directly, using standard file upload mechanisms over HTTPS. In addition, the JMyproxy package [24] provides a Java graphical user interface (GUI) that can create a proxy locally and delegate a credential to a Myproxy server. The initial login page that displays the Myproxy interface to the demo portal is shown in Figure 4(a). In the current implementation, all users that can either supply a delegated credential or retrieve one from the Myproxy server are authorized portal users. However, if the portal administrator wished to further restrict access, an encrypted password file on the portal or a secure back-end database could also be used to determine authorization information.

5.2. Job submission

The GPDK job submission beans provide two different secure mechanisms for executing programs on remote resources. A GSI enhanced version of the Secure Shell (SSH) [9] software enables interactive jobs to be submitted to Grid resources supporting the GSI enabled SSH daemon. For all other job submissions, including batch job submissions, the Globus GRAM protocol is used and jobs are submitted to Globus gatekeepers deployed on Grid resources. Briefly, the GRAM protocol enables resource submission to a variety of resource scheduling systems using the Resource Specification Language (RSL) [20], allowing various execution parameters to be specified, e.g. number of processors, arguments, wall clock or CPU time.

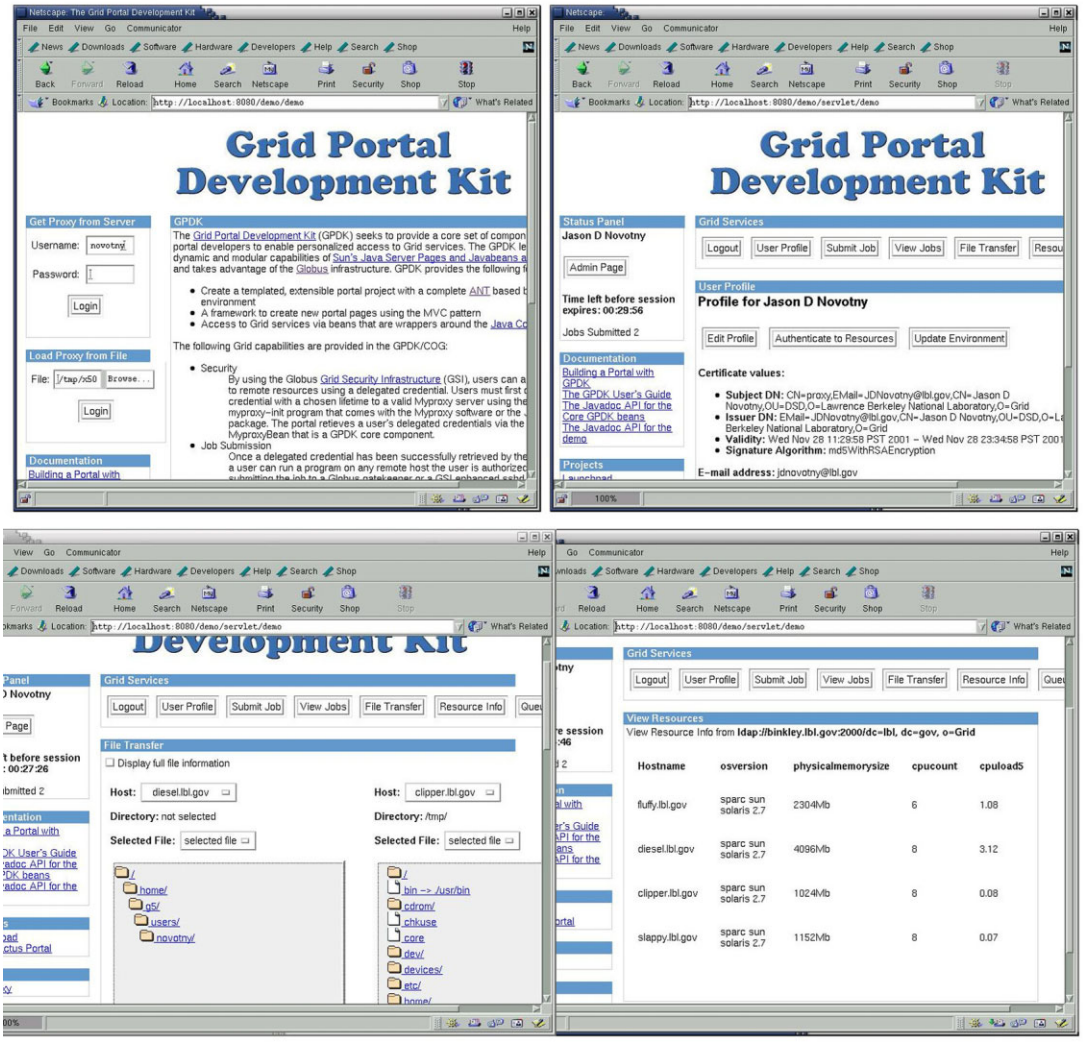


Figure 4. GPDK demo pages clockwise from top left: (a) login page; (b) user profile page; (c) resources page; (d) file transfer page.

The primary 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, and encapsulates RSL by including 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` using the GSI enhanced SSH client [9]. The `GramSubmissionBean` capabilities are provided once again by the Java CoG library.

Once a job has been successfully submitted, a `JobInfoBean` is created containing 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.

Job monitoring of submitted jobs is provided through the `JobMonitorBean`, a component initialized at start-up by the PE. The `JobMonitorBean` periodically queries the GRAM URLs on behalf of a user to keep track of job status based on the GRAM job status codes, e.g. active, running, or failed. Because the `JobMonitorBean` has application scope, it can save job status to a user's profile even if the user has logged out.

5.3. File transfer

The GPDK file transfer beans encapsulate the GridFTP [21] API implemented as part of the CoG toolkit and provide file transfer capabilities, 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 command 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 time outs. The `GSIFTPViewBean` is an example view bean used by a JSP to display the results of browsing a remote GSI FTP server. Figure 4 shows the demo file browsing and transferring page.

5.4. Information services

The Grid Forum Information Services working group has proposed the GIS architecture for deploying information services on the Grid and supported the 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 an implementation of a GIS, known as the Metacomputing Directory Service [25] using OpenLDAP, an open-source LDAP server. Although, the Java CoG toolkit provides support for LDAP using the Java Naming and Directory Interface, GPDK uses the open source Netscape/Mozilla Directory Software Development Kit [26] as it proved easier to use in practice and also provides support for developing a connection pool for maintaining multiple connections to several GIS providers, thus eliminating the need for clients to re-connect during each query. However, this model will need to be re-evaluated with the widespread deployment of the Metadata Directory Service (MDS)-2 architecture that includes GSI enhancements making it necessary for clients to re-authenticate to the MDS for each query. GPDK provides an

`MDSQueryBean` and `MDSResultsBean` for querying and formatting results obtained from the MDS. Currently, the GPDK 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.

5.5. 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. Application portals built on top of the GPDK can subclass the GPDK user profile bean to provide additional application specific information for a particular class of users.

6. GPDK AS A PORTAL DEVELOPMENT ENVIRONMENT

Computational science collaborations can benefit significantly by providing Web access to software, data, and expertise. The GPDK 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 incorporated by the GPDK. 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 accessible, collaborative environment.

The GPDK is packaged with a demo portal to showcase the functionality of the service beans and provide several common portal operations. During installation, the GPDK 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 Java application server. The template Web pages include HTML and JSPs intended to demonstrate the GPDK service beans. 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 PE used to initialize and shut down any resources required by the new project portal. The goal of the template portal is not to provide a fully polished interface ready for production deployment, but rather a demonstration of GPDK capabilities that can be easily extended and specialized for providing an application group with a common subset of Grid capabilities. Figure 5 shows the generalized GPDK architecture organized into core GPDK components, GPDK generated components, and application specific user developed components.

The GPDK template source code also contains all the APOs necessary for performing the various portal operations that demonstrate core GPDK services and beans. The APs 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` and `FileTransferPage` demonstrate the GPDK service beans in the areas of job submission and file transfer. The template JSPs are displayed upon the successful

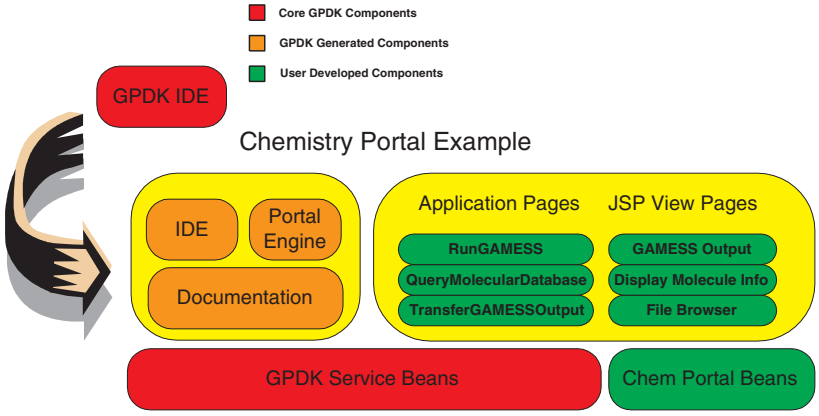


Figure 5. Creating a new portal.

completion of the page object operations. If an error occurs during the execution of an AP, an exception is thrown with a relevant error message that is displayed to the client by a specialized error JSP.

The GPDK relies on ANT [27] for cross-platform project compilation and deployment. ANT provides an XML based Java build tool that is analogous to the Makefile process 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.

The following steps are followed to create a new application specific portal called ChemPortal using the GPDK:

- Compile core GPDK service beans and portal objects. When GPDK is compiled for the first time using the ANT build script, e.g. `build all`, all of the necessary classes are compiled and placed into the GPDK Java Archive Repository file.
- Invoke the ANT build tool to create a new portal with the desired name, e.g. `build new ChemPortal`. By specifying a new project target, e.g. ChemPortal, when invoking the GPDK build script, a new portal subproject, ChemPortal, is created containing the pre-processed template portal pages. The template portal pages are composed of functional Java source code, template action page classes and VPs in JSP in addition to Javadoc generated API documentation for the generated source code. File pre-processing capabilities that allow the GPDK to generate a complete template portal project is made possible by the substitution capabilities provided by the ANT build tool. ANT pre-processes the template file names as well as the contents of the file to provide project specific Java code. For example, the template GPDK servlet is pre-processed to create a `ChemPortalServlet`. The PE template is pre-processed to create a ChemPortal object that uses the core GPDK Portal object for initialization and shut down routines. Similarly, the GPDK User Profile object is subclassed to create a ChemPortal User Profile object that can be edited to contain data pertinent to the users of the ChemPortal.

The specified project namespace also provides a packaging namespace for portal source code. Core GPDK objects and beans are packaged under the `org.gpdk` namespace as per the Sun source code packaging convention and the pre-processed template project code is packaged under a new project namespace, e.g. `org.chemportal`. 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 that can make use of the standard set of GPDK service beans.

- Modify and create new APs and JSP VPs for project specific portal operations. As new portal operations are needed, ChemPortal developers would create a new APO that implements the Page interface. The Page interface defines only one method, `execute`, that must be filled in by the developer. A corresponding VP written as a JSP would need to be developed to return a display of the operation results to the client. As an example, imagine a new capability that allows ChemPortal users to run a General Atomic and Molecular Electronic Structure System (GAMESS) [28] simulation with a particular set of parameters. (GAMESS is a scientific code used for performing *ab initio* chemistry simulations.) The ChemPortal developer would create a `RunGAMESSPage` class that would use the GPDK Job Submission beans to initiate a GAMESS simulation with the appropriate run parameters. Furthermore, the GPDK file transfer beans could be used to move any generated output files to a storage server for permanent archiving. Finally, the developer would create a `GAMESSOutput` JSP that would be used to provide information on a completed run to the end user.
- Deploy new portal project to application server. Simply invoking the ANT build script, e.g. `build.pl all`, will package all of the Web application files, HTML, Java classes, and JSPs in the form of a Web application repository file and deploy it to the Tomcat application server, where it can be accessed by a Web browser.

The steps presented above allow an application developer to create a fully functional portal complete with template code in a matter of minutes. The build environment provides easy to use compilation and deployment capabilities, eliminating the need for additional Makefiles or project specific build scripts. In fact, most of the work done in developing a new portal is in the action pages and JSPs that are project specific.

6.1. Application portals based on GPDK

The GPDK has proven successful in the creation of application specific portals under development by other research groups. The following briefly describes various ongoing portal projects that have been developed using the GPDK framework. The NASA Launchpad [29] user portal seeks to provide Web-based access to users of the NASA Information Power Grid (IPG) [30]. The 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 GISs. 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) [31] 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 remotely check out and compile Cactus with specialized options, as well as submit various Cactus simulations to a distributed set of 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 the GPDK for job submission and file staging capabilities.

7. RELATED WORK

While the GPDK demo portal is very similar to many other portal projects, the ability to develop new portals using the GPDK framework offers a new capability to the scientific community and is one of the first of its kind. As a software development kit, the GPDK is most similar to the GridPort [32] toolkit developed by the San Diego Supercomputer Center to facilitate the development of application specific portals. The GridPort toolkit is implemented in Perl and makes use of the existing HotPage [4] technology for providing access to Grid services. GridPort supports many of the same Grid services as GPDK, including the Myproxy service, GRAM, and GIS.

Application portal development using GridPort is enabled in two ways: the first approach requires that Globus software tools be installed because the GridPort scripts wrap the C Globus command line tools in the form of Perl CGI scripts. The second method of developing a portal using GridPort does not require Globus, but relies on the CGI scripts that have been configured to use a primary GridPort portal as a proxy for access to GridPort services such as authentication, job submission, file transfer, etc. In the second approach, an application scientist can quickly deploy a Web server configured with a set of GridPort CGI scripts to perform very generic portal operations. However, the ease of deployment comes at a cost of portal customizability. Because the HTTP response from the proxy GridPort server contains the HTML to be displayed, the graphical interface displayed to portal users is the same as the base GridPort server. In addition, both logic and design are encapsulated in the GridPort Perl CGI scripts, making code maintainability and customization more difficult over the approach taken by GPDK, which separates the graphical interface provided by template JSPs and the logic contained within the GPDK beans.

8. CONCLUSION AND FUTURE WORK

The architecture of the GPDK is highly modular and provides easy re-use of Grid components and a useful framework for the development and deployment of application specific portals. Based on the MVC design paradigm, the GPDK has proven to be an extensible and flexible software package in use by several other portal building efforts. The GPDK makes use of commodity technologies including the open-source servlet container Tomcat and the Apache Web server. The GPDK makes use of the Java CoG toolkit for its pure Java implementation of client side Globus Grid services, as well as other

widely available, commodity Java libraries. Future work on the GPDK involves development in the following areas.

- Improve and enhance the functionality of GPDK service beans to support emerging Grid services and develop new portal capabilities. The GPDK information service beans will be enhanced to support the improved and secure MDS-2 infrastructure as it becomes widely deployed. Other future developments include the integration of a secure database to store user profiles rather than maintaining them on the portal. Additional application information may also be stored in the database or in the information servers, for access by the GPDK information service beans. Task composition from a portal has become increasingly important as portal operations become more complex. Portal users 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. Emerging Grid monitoring and event technologies [33] will provide useful information on network and computer utilization and performance, allowing portal users and scheduling software alike to make better informed resource scheduling decisions. As these technologies become widely deployed, components for accessing these services will be developed in the GPDK.
- Investigate and prototype new distributed computing technologies including Web services. Web services have become increasingly important in the enterprise community and many new standards and implementations are emerging from Sun, Microsoft, and IBM. The Web Services Definition Language [34] permits services to be defined by a standard interface and registered and discovered using the Universal Description, Discovery and Integration [35] specification. The Simple Object Access Protocol [36] provides a standard for communicating structured information using XML. GPDK client Web service beans will be developed that are capable of exchanging SOAP messages with Grid enabled Web services as they become widespread.

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, Scott Koranda, and John White.

REFERENCES

1. Foster I, Kesselman C (eds.) *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann: San Francisco, 1998.
2. The Astrophysics Simulation Collaboratory. http://www.ascportal.org [22 November 2001].
3. Pancerella CM, Rahn LA, Yang CL. The diesel combustion collaboratory: Combustion researchers collaborating over the Internet. *Proceedings IEEE Conference on High Performance Computing and Networking*, November 1999.
4. Boisseau J, Mock S, Thomas M. Development of Web toolkits for computational science portals: The NPACI HotPage. *Proceedings 9th IEEE International Symposium on High Performance Distributed Computing*, 2000.
5. Akarsu E, Fox G, Haupt T, Youn C. The Gateway system: Uniform Web based access to remote resources. *IEEE Concurrency: Practice and Experience* 2000.

6. Romberg M. The UNICORE architecture. *Proceedings 8th IEEE International Symposium on High Performance Distributed Computing*, 1999.
7. The GPDK Project Page. http://www.itg-lbl.gov/Grid/projects/GPDK/index.html [22 November 2001].
8. Globus Web Site. http://www.globus.org [22 November 2001].
9. GSI Software Information. http://www.globus.org/security [22 November 2001].
10. Foster I, Kesselman C. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputing Applications*, 1997.
11. Grid Forum Web Site. http://www.gridforum.org [22 November 2001].
12. Laszewski G, Foster I, Gawor J. CoG Kits: A bridge between commodity distributed computing and high-performance grids. *Proceedings of the ACM Java Grande Conference*, 2000.
13. Gamma E, Helm R, Johnson R, Vlissides J. *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley: Reading, MA, 1995.
14. Apache Webserver Project. http://www.apache.org [22 November 2001].
15. ModSSL. http://www.modssl.org [22 November 2001].
16. Jakarta Apache Project. http://jakarta.apache.org [22 November 2001].
17. Tomcat open-source servlet container. http://jakarta.apache.org/tomcat [22 November 2001].
18. IAIK Security Libraries. http://jcewww.iaik.at/ [22 November 2001].
19. Sun Java Secure Sockets Extension. http://java.sun.com/products/jsse [22 November 2001].
20. Czajkowski K, Foster I, Karonis N, Kesselman C, Martin S, Smith W, Tuecke S. A resource management architecture for metacomputing systems. *Proceedings of the IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.
21. Allcock W, Bester J, Bresnahan J, Chervenak A, Liming L, Tuecke S. GridFTP: Protocol extensions to FTP for the Grid. *Grid Forum Working Draft*, March 2001. http://www.gridforum.org.
22. Novotny J, Tuecke S, Welch V. An online credential repository for the Grid: MyProxy. *Proceedings 10th IEEE Symposium on High Performance Distributed Computing*, 2001.
23. Foster I, Karonis N, Kesselman C, Koenig G, Tuecke S. A secure communications infrastructure for high-performance distributed computing. *Proceedings 6th IEEE Symposium on High Performance Distributed Computing*, 1997.
24. The JMyproxy client. ftp://ftp.george.lbl.gov/pub/globus/jmyproxy.tar.gz [22 November 2001].
25. Czajkowski K, Fitzgerald S, Foster I, Kesselman C. Grid information services for distributed resource sharing. *Proceedings 10th IEEE Symposium on High Performance Distributed Computing*, 2001.
26. Netscape Directory and LDAP Developer Central. http://developer.netscape.com/tech/directory/index.html [22 November 2001].
27. The Jakarta ANT Project. http://jakarta.apache.org/ant/index.html [22 November 2001].
28. The General Atomic and Molecular Electronic Structure System. http://www.msg.ameslab.gov/GAMESS/GAMESS.html [22 November 2001].
29. The NASA Launchpad User Portal. http://www.ipg.nasa.gov/launchpad [22 November 2001].
30. Johnston WE, Gannon D, Nitzberg B. Grids as production computing environments: The engineering aspects of NASA's information power grid. *Proceedings 8th IEEE Symposium on High Performance Distributed Computing*, 1999.
31. Allen G, Daues G, Foster I, Laszewski G, Novotny J, Russell M, Seidel E, Shalf J. The Astrophysics Simulation Collaboratory Portal: A science portal enabling community software development. *Proceedings 10th IEEE International Symposium on High Performance Distributed Computing*, 2001.
32. Boisseau J, Dahan M, Mock S, Mueller K, Sutton D, Thomas M. The GridPort Toolkit: A system for building Grid portals. *Proceedings 10th IEEE International Symposium on High Performance Distributed Computing*, 2001.
33. Grid Monitoring Architecture. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/ [22 November 2001].
34. Web Services Definition Language. http://www.w3.org/TR/wsdl [22 November 2001].
35. Universal Description, Discovery and Integration. http://www.uddi.org [22 November 2001].
36. Simple Object Access Protocol. http://www.w3.org/TR/SOAP/ [22 November 2001].
37. Java Servlet 2.3 and Java Server Pages 1.2 Specifications. http://java.sun.com/products/servlets [22 November 2001].

