Web Services Communication within the PROGRESS Grid-Portal Environment

Piotr Grzybowski, Michal Kosiedowski, Cezary Mazurek
Poznan Supercomputing and Networking Center, ul. Noskowskiego 10, 61-472 Poznan, Poland
{piotrgrb, kat, mazurek}@man.poznan.pl
tel.+48 61 8582030, +48 61 8582035, fax. +48 61 8525954

Abstract

The grid is the next generation computing infrastructure able to handle the growing requirements for computing power. Portals are anticipated as the user's access point to these resources. The whole grid-portal infrastructure constitutes a distributed environment in which efficient and flexible communication manners play a key role. The emerging web services technology has been chosen as the best solution for organizing communication in grid-portal systems. In this paper we would like to present the PROGRESS grid-portal environment in which we implemented web services communication between distributed modules of the system.

1. Introduction

The PROGRESS grid-portal environment has been designed and implemented as a result of the "Access Environment to Computational Services Performed by a Cluster of SUNs" project. This initiative was undertaken within the PIONIER National Program [1] and is funded by the State Commitee for Scientific Research and Sun Microsystems Poland. The project will last until May 2003, but we are keen to continue our research afterwards. The PROGRESS project aims to develop an access environment to computational resources, which would allow to create a comfortable work place for grid users. In Section 2 we present the overall architecture of the PROGRESS system.

The following sections describe our experiences with applying the Web Services technology within the grid-portal environment. Section 3 deals with external interfaces of PROGRESS system modules. These interfaces allow for flexible intercommunication between all PROGRESS items. Next, in Section 4, we present the internal communication in the data management system, which is a distributed storage system for scientific data used in experiments taking place within the PROGRESS grid. Eventually, in Section 5, we summarize our work and draw research and implementation paths for the future.

2. PROGRESS grid-portal environment

The PROGRESS grid-portal environment framework has been assumed to serve as a solution from the shelf for deployment of multiple grid access environments in the scope of the PIONIER program [2]. This could be achieved by designing a system, the architecture of which is presented in Figure 1.

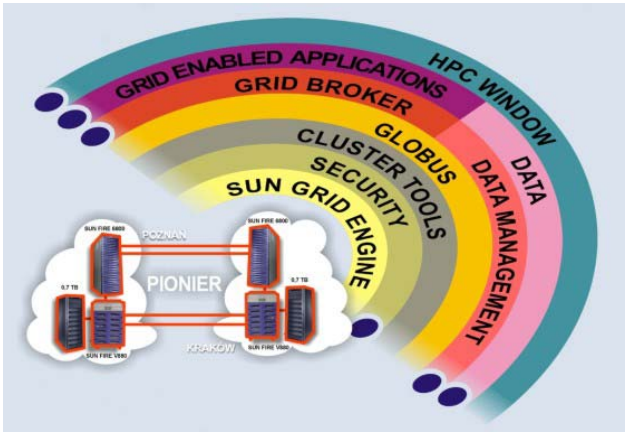

Figure 1. PROGRESS grid-portal environment architecture

The testbed prototype uses three bioX grid enabled applications for testing developed tools and the integrated system. These applications are available for running scientific experiments in the cluster of three Sun computing systems and two data servers distributed between Poznan and Krakow, which serve as the PROGRESS testbed grid. The PROGRESS bioX grid does not, however, limit the application factory to include this limited number of applications. We have developed tools that enable application developers to add their newly designed applications and publish them in the PROGRESS system.

The binaries of applications are stored in the data management system which is also used as the source for grid job input data and a place for storing the results of scientific experiments. There is also a special item within the data management system which serves as a proxy to

scientific data banks – an example of such bank used in the PROGRESS testbed is the SRS [3].

Applications are executed in the grid basing on the resource allocation performed by the grid broker. The grid resource broker analyzes the received computing job descriptions, downloads the input data and application binaries if necessary and schedules the jobs for execution. The job description is passed to the grid broker in the form of an Extended Resource Specification Language (XRSL) document [4].

In PROGRESS the job descriptions are prepared by the grid service provider. This is a layer in the grid-portal environment architecture introduced in [5]. It allows to flexibly deploy numerous client interfaces (most of them are web computing portals) utilizing the same grid resources [6]. The PROGRESS grid service provider is equipped with the following services: job submission, application management, service provider management and an example of an informational service, the short news service. The grid service provider serves as a mediator between user interfaces and the grid. The grid service provider and its client interfaces constitute the HPC Window [7].

The PROGRESS user interfaces include the HPC Portal [8] and the migrating desktop [9]. The PROGRESS computing portal is a bioX thematic web portal, which provides users with possibilities of accessing grid resources underlying the grid service provider, applications collected in the PROGRESS application factory and scientific data stored in the data management system. Additionally, the portal provides means of utilizing informational services of the grid service provider. The migrating desktop is an example of a Java standalone client interface. It allows to manage data available in the data management system and execute PROGRESS applications with the use of the grid service provider.

The security of the grid-portal environment described above is assured by a specially designed model. It is based on an intrusion detection system [10] and resource access decision model [11]. This infrastructure enables detecting abnormalities in hardware and software performance and secures the whole system from an unauthorized usage.

Despite layers mentioned above the PROGRESS grid-portal environment uses some software components integrated with the services which have been implemented by us. The Sun Grid Engine is used as a platform for management of the resources of each computing system. The HPC Cluster Tools provide libraries for parallel execution of grid job processes. The Globus Toolkit serves as the PROGRESS grid management system.

3. Web Services communication within PROGRESS

In the previous section we presented the architecture of the PROGRESS grid-portal environment. For communication between the described modules we assumed the Web Services technology. Additionally, we use FTP, GASS [12] and GridFTP [13] protocols for transferring data to and from the data management system. Communication manners in PROGRESS are presented in Figure 2.

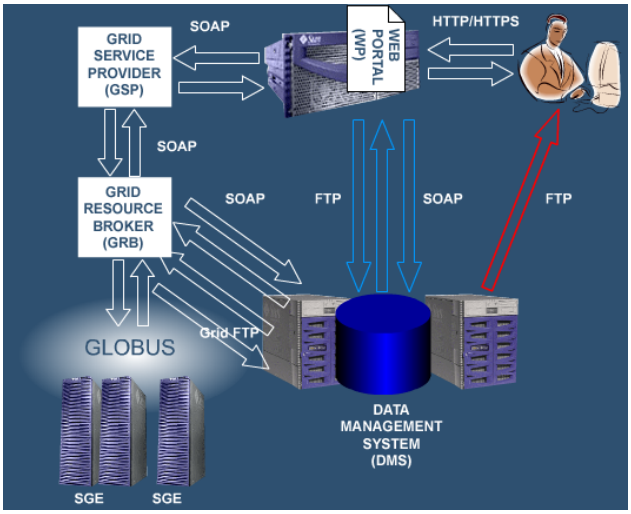

Figure 2. Communication in the PROGRESS grid-portal environment

The grid service provider interface is used by the client interfaces, for example the web portal, and the grid resource broker. The data management system Web Services methods are also invoked by the web portal and the grid broker. Finally, the grid broker provides functions for communication with the grid service provider. We describe Web Services operations delivered by these modules in this section. At the end we provide an example WS communication scheme between PROGRESS modules during a session, in which a computing job is created, submitted and executed in the grid.

3.1. Grid service provider interface

The grid service provider delivers services which might be used by grid user interfaces. Each of these services prepares data for presentation in the client interface, for example the web computing portal. The service then is responsible for logical analysis of client requests and data preparation, and the client (or portal content provider) for presentation of this data and

interaction with the user. The idea of this division of functions is presented in Figure 3. The grid service provider is implemented using J2EE technologies.

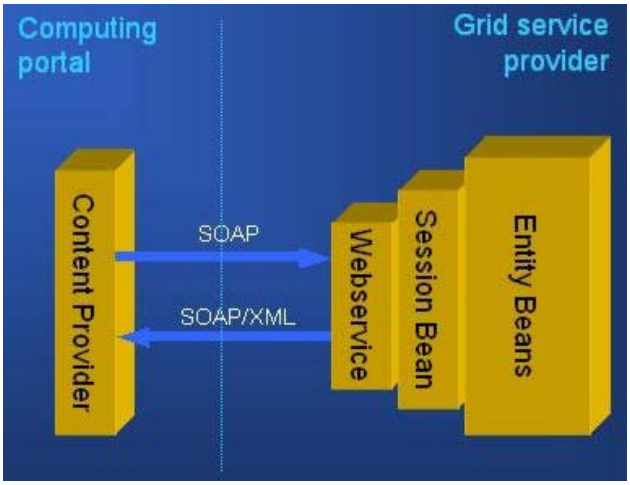

Figure 3. Web services communication between a computing portal and the grid service provider

There are four services available in the current version of the PROGRESS grid service provider. These are: the job submission service, the application management service, the provider management service and the short news service. They provide client interfaces with multiple web service operations that allow performing the actions described further on.

The job submission service allows to create, modify, submit and delete the computing jobs. The jobs may be constructed of a single task or a combination of sequences and/or parallels. The job submission service also provides operations for managing single tasks of a job, setting application parameters, adding references to input and output files, and editing task resource requirements. The service also delivers an operation for monitoring the execution of the job. The job submission service is the only service which provides a method to be used by the grid resource broker. This operation allows the grid broker to notify the service provider about changes in the job execution status.

The application management service enables adding applications to the application factory, as well as modifying the definition of and removing the application. The service also delivers methods for managing virtual applications, that is templates encapsulating sequences and/or parallels of application executions.

The provider management service allows to add and delete services as well as to add and delete instances of services if the service allows to create multiple instances. Also modification of the service description is available, which includes, among others, such parameters as URLs to the service and to the service WSDL description.

The short news service, which is an example of an informational service, provides operations for adding, modifying, deleting and reading news. This service is an example of a multiple instance service so it is also equipped with operations allowing to manage its own instances.

Grid service provider services were implemented using Sun Forte For Java 4 and the web services development framework delivered with this package. They are currently deployed on the J2EE Reference Implementation server. We plan, however, to improve the performance of these services using Sun One Studio 4 and the framework delivered therewith, and deploy the grid service provider to the Sun One Application Server 7 [14].

3.2. Data management system interface

The data management system delivers functions for directory, file and metadata management. These operations are available to user interfaces and the grid resource broker through the front end module of the data management system – the data broker (see Section 4 for more information on the data management system architecture).

As far as directory management is concerned, the data broker provides operations for adding, removing and renaming directories, and for retrieving root and current path as well as changing the path to another: a relative or absolute one. There is also an operation for listing the content of the current directory.

For file management we designed methods to enable adding, removing and renaming files. Additionally, since files in the data management system may be stored in numerous data containers, the data broker provides operations for adding and removing, as well as retrieving, the physical localization of data. Files may also be put into archives which are used as regular data containers within the system – we call them user data containers. Such containers may be added and deleted with the use of data broker methods. Finally, it is also possible to add and remove symbolic links to file and directories.

Eventually, the metadata operations available through the data broker include metadata scheme management (adding, removing and modifying). Methods for retrieving a list of available schemes and the scheme attributes descriptions are also provided there. Other functions allow to assign schemes to files and edit values of attributes. The data broker also delivers operations for searching the metadata repository for files according to the provided requirements.

The data broker and other data management modules (described in Section 4) utilize the Apache SOAP [15] implementation. They were deployed onto the Jetty

Container [16], which is a Java HTTP server and servlet container.

3.3. Grid resource broker interface

The grid resource broker delivers six web services operations for use by the grid service provider. They allow to submit jobs for execution in the grid, retrieve the list of jobs executed on behalf of a particular user, get the status of a job, and cancel, suspend and resume jobs. The grid broker Web Services were implemented using the Apache's Axis framework [17] and deployed onto the Tomcat application server [18].

3.4. Web Services communication between PROGRESS modules

The above-mentioned Web Services methods allow for flexible communication between modules. We illustrate possible communication schemes with the example of a computing job creation, submission and execution session. The session involves the PROGRESS HPC Portal (a grid user interface), the job submission and application management services of the grid service provider, the data management system and the grid resource broker.

At first the portal user clicks the "Add job" button on the PROGRESS portal web page. The provided job name and description are saved by the portal in the job submission service's database with the use of the *saveJob()* method. Then the user adds the main task of the job. The portal retrieves the list of available applications invoking the application management's operation *getApplications()*. The user chooses the application from the list and enters the description details for the task and the task is saved by the portal with the use of the *saveTaskOfJob()* operation. The user needs to choose the input file for the job. This is done with the use of the data broker's *listUserDirectory()* function, which allows to retrieve the contents of each directory found in the data management system file tree. Then the user creates a new file for storing the results of his/her job. To do this he/she instructs the portal to invoke the *addUserFile()* method in the data management system. After both of these actions the portal saves the references to the chosen or created files with the use of the job submission service's *saveStdOfTask()* operation.

When the job is fully configured it is submitted for execution in the grid. First, the portal instructs the job submission service to prepare the XRSL job description and submit it to the grid resource broker by invoking the service's *submitJob()* operation. The grid service provider prepares the description and passes it to the grid resource broker using its *submitJob()*). The grid broker analyzes the job description and retrieves the application binary and the input data. It uses the data broker's *getUserFileLocation()* method for learning the URLs of the files and downloads them. Then the job is submitted to Globus for execution and the grid resource broker informs the job submission service about the change in the job status (*changeJobStatus()*) – the job status is *active* at this time. After the job is executed the grid broker changes the job status to *done* and retrieves the location for the output file to upload the results. Then the status of the job is changed to *finished*. The portal user may always retrieve the list of his jobs and check their status by instructing the portal to invoke the job submission service's *getUserJobs()* and *getJobStatus()* functions.

The communication scheme described above is presented graphically in Figure 4. We want to clearly state that this is only an example of possible communication schemes in PROGRESS. However, the bounds of this paper does not allow us to present all possible actions and operations.

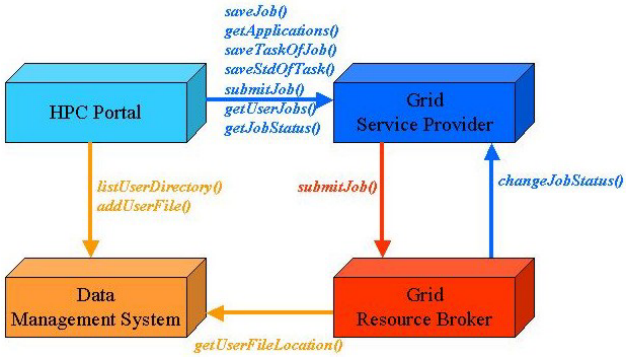

Figure 4. An example of Web Services operations invocation

4. Data management system internal communication

The data management system is a distributed environment for storing scientific data [19]. In PROGRESS it serves as the source of grid job input data and destination for the experiments results. It also provides the functionality of a proxy to scientific data banks. Figure 5 presents the data management system architecture. The metadata repository is used for managing information about data stored in the system. The data broker serves as an interface to external clients, such as the computing web portal and the grid resource broker. The main purpose of the mirror and proxy module is to grant access to various external objects, like data from the SRS system, in a uniform way. Finally, the data storage modules store physical data in one of numerous

data containers: computer file systems, relational database systems or tape storage systems.

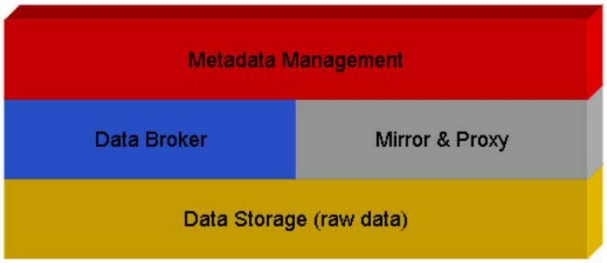

Figure 5. Data management system architecture

The distributed nature of the data management system has led to designing an internal communication scheme. We based it on the Web Services technology (see Figure 6). All but the metadata repository modules can appear in multiple instances.

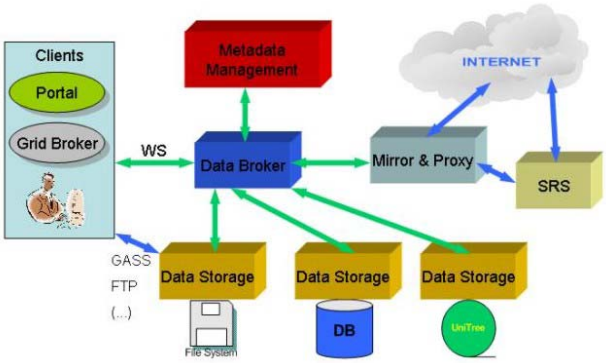

Figure 6. Data management system communication scheme

The metadata repository module is responsible for storing and manipulating metadata. The format of the metadata can be defined by the user or chosen from the list of predefined formats like, for example, the Dublin Core standard [20]. The decision to run this module as a single instance service was made basing on the complexity of the metadata handling problem. Running multiple instances would require assuring metadata consistency and reliability. The metadata management module delivers operations allowing to change meta-information about files stored in the data management system, create and remove files, add, delete and manage the meta-information connected with files. The metadata repository requires authorization; therefore it delivers functions which assure the authorization of users. This module stores all information about the data in the system and is the only access point to this information.

The data storage module enables access to physical data stored within the data management system. The data can be organized as files on computer file systems, binary large objects (BLOBs) in databases or files on tapes or optical disks in storage libraries. The data may be shared using FTP, GASS or GridFTP protocols. Applying multiple instances of the data storage module assures uninterrupted data accessibility, even in case of network or system failures. The data storage module delivers operations allowing to create and delete files, reserve container space, enable and disable access to files through available data transfer protocols, lock and unlock files, and retrieve information about the state of files and the container free space.

The mirror and proxy module is currently under implementation. We plan to add it in the upcoming months. The module will relay communication between clients and scientific databanks. These banks may be local mirrors of databanks (like the PSNC's mirror of the SRS bank) or databanks available from the Internet. Web Services operations of the mirror and proxy module will enable to access external data as an internal node in the DMS. This module will enable to describe external data with meta-information stored in the meta repository module. The functions of the mirror module will enable to manage the replicas of external databases.

5. Conclusions

Numerous grid-portal environments have been deployed around the world. Some of the most important include San Diego Supercomputer Center's HotPage [21], Legion Grid Portal from the University of Virginia [22] and NASA's Information Power Grid [23]. Deployed computing portals use grid portal frameworks like SDSC's GridPort [24], LBNL's Grid Portal Development Kit [25] or NICE's Enginframe [26]. Although these packages enable to create comfortable places for work with the grid, they lack flexibility given by introducing the Web Services technology. By utilizing this technology the PROGRESS grid-portal environment facilitates adding new components to the already deployed infrastructure and building new computing portals. These features are of key importance to the presumptions made before undertaking the project.

It is also important to mention that there have been similar approaches introduced recently. We believe that the solutions like the one presented in [27] and implementation of the OGSA architecture [28, 29] will add more flexibility to grid-portal environments. We think of integrating the latter into the PROGRESS framework. Introducing grid service provider services in the form of OGSA grid services should facilitate building computing portals at even a bigger rate. Although we believe our current implementation fulfills the

requirements of deployment flexibility and work place comfortableness, we feel it is very important to integrate the emerging standards into grid-portal environments.

The PROGRESS grid-portal environment with all functionality described in this paper is currently online as a testbed at http://progress.psnc.pl/portal. Its first presentation took place at the Supercomputing 2002 exhibition.

6. References

[1] J. Rychlewski, J. Weglarz, S. Starzak, M. Stroinski, and M. Nakonieczny, "PIONIER: Polish Optical Internet", Proceedings of ISThmus 2000 Research and Development for Information Society conference, Poznan, Poland, 2000, pp. 19-28

[2] M. Kosiedowski, C. Mazurek, and M. Stroinski, "PROGRESS - Access Environment to Computational Services Performed by Cluster of Sun Systems", Proceedings of The 2nd Cracow Grid Workshop, Cracow, Poland, December 2002, accessed from http://progress.psnc.pl/

[3] SRS System, accessed from http://srs.man.poznan.pl/

[4] J. Pukacki, and M. Wolniewicz, "Extended Resource Specification Language", accessed from http://progress.psnc.pl/

[5] M. Bogdanski, M. Kosiedowski, C. Mazurek, and M. Wolniewicz, "Grid Service and Access Management within User Service Environment", presented to the Global Grid Forum, Grid Computing Envrionments Research Group, September 2002, accessed from http://progress.psnc.pl/

[6] M. Bogdanski, M. Kosiedowski, C. Mazurek, and M. Wolniewicz, "GRID SERVICE PROVIDER: How to improve flexibility of grid user interfaces?", accepted for publication at The 3rd International Conference on Computational Science, June 2nd-4th 2003, St. Petersburg, Russia, accessed from http://progress.psnc.pl/

[7] M. Bogdanski, M. Kosiedowski, C. Mazurek, and M. Wolniewicz, "Facilitating access to grid resources with the use of the HPC Window", submitted for presentation at The 9th International Conference on Parallel and Distributed Computing Euro-Par 2003, August 26th-29th 2003, Klagenfurt, Austria, accessed from http://progress.psnc.pl/

[8] "PROGRESS HPC Portal", accessed from http://progress.psnc.pl/portal/

[9] M. Kupczyk, R. Lichwala, N. Meyer, B. Palak, M. Plociennik, and P. Wolniewicz, "Roaming Access and Migrating Desktop", Proceedings of The 2nd Cracow Grid Workshop, Cracow, Poland, December 2002

[10] M. Chmielewski, A. Gowdiak, S. Fonrobert, N. Meyer, and T. Ostwald, "VALIS/Valkyrie", Proceedings of The 2nd Cracow Grid Workshop, Cracow, Poland, December 2002

[11] "Resource Access Decision, Version 1.0", accessed from http://www.omg.org/technology/documents/formal/resource_access_decision.htm

[12] "Global Access and Secondary Storage (GASS)", accessed from http://www-fp.globus.org/gass/

[13] "The GridFTP Protocol and Software", accessed from http://www-fp.globus.org/datagrid/gridftp.html

[14] Sun Open Net Environment (Sun ONE), accessed from http://wwws.sun.com/software/sunone/

[15] Apache SOAP, accessed from http://ws.apache.org/soap/

[16] Jetty Java HTTP Servlet Server, accessed from http://jetty.mortbay.org/jetty/

[17] Apache Axis, accessed from http://ws.apache.org/axis/

[18] The Jakarta Project, accessed from http://jakarta.apache.org/

[19] P. Grzybowski, C. Mazurek, P. Spychala, and M. Wolski, "Data Management System for grid and portal services", submitted to Grid Computing: Infrastructure and Applications special issue of The International Journal of High Performance Computing Applications (IJHPCA), Cardiff University, UK., accessed from http://progress.psnc.pl/

[20] Dublin Core standard, accessed from http://www.dublincore.org/

[21] NPACI HotPage Grid Computing Portal, accessed from https://hotpage.npaci.edu/

[22] A. Natrajan, A. Nguyen-Tuong, M. A. Humphrey, and S. Grimshaw, "The Legion Grid Portal", accessed from http://legion.virginia.edu/papers.html

[23] Information Power Grid, accessed from http://www.ipg.nasa.gov/

[24] M. Thomas, S. Mock, J. Boisseau, M. Dahan, K. Mueller, and D. Sutton, "The GridPort Toolkit Architecture for Building Grid Portals", Proceedings of the Tenth IEEE International Symposium On High Performance Distributed Computing, 2001

[25] J. Novotny, "The Grid Portal Development Kit", accessed from http://www.cogkits.org

[26] Enginframe, accessed from http://www.enginframe.com/

[27] M. Pierce, G. Fox, Ch. Youn, S. Mock, K. Mueller, and O. Balsoy, "Interoperable Web Services for Computational Portals", Proceedings of Supercomputing 2002, Baltimore

[28] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Open Grid Service Infrastructure WG, Global Grid Forum, June 2002, accessed from http://www.globus.org/research/papers.html

[29] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman, "Grid Service Specification", Open Grid Service Infrastructure WG, Global Grid Forum, July 2002, accessed from http://www.globus.org/research/papers.html