Grid Service Provider: How to Improve Flexibility of Grid User Interfaces?

Maciej Bogdanski, Michal Kosiedowski, Cezary Mazurek, and Malgorzata Wolniewicz

Poznan Supercomputing and Networking Center, ul. Noskowskiego 10, 61-704 Poznan, Poland
{bogdan, kat, mazurek, gosiaw}@man.poznan.pl

Abstract. Poznan Supercomputing and Networking Center has been involved in continuous research concerning the development of grid environments. In the scope of the PROGRESS project we designed a Grid Service Provider (GSP) which acts as a new additional layer between the grid and grid user interfaces. The introduction of the GSP enables easy development of many different grid user interfaces such as web portals and standalone applications. In this paper we present this newly distinguished item of grid-portal environment architecture, which might improve the flexibility of grid user interfaces.

1 Introduction

Grids and grid access environments have recently been one of the most important problems in the area of High Performance Computing. There are many grid initiatives around the world which succeeded in deploying grid access environments. The most common manner of accessing the grid is using a web portal. This approach is most comfortable for the end user as web interfaces are easy to use through their daily usage in the Internet. Grid-portal environments (GPEs), which are environments where the grid is accessed with the use of a web portal, constitute a widely undertaken research problem. The known GPE and GPE toolkit implementations include Grid Portal Toolkit (GridPort) [1] and HotPage [2] designed by the San Diego Supercomputer Center (SDSC), Legion Grid Portal (LPG) [3] from the University of Virginia, and Information Power Grid [4] enabling access to a joint grid infrastructure under the auspices of NASA and using the Grid Portal Development Kit (GPDK) [5] from the Lawrence Berkeley National Laboratory (LBNL).

The "Access Environment to Computational Services Performed by Cluster of SUNs" project (PROGRESS) [6], which is currently under development in Poznan Supercomputing and Networking Center (PSNC) also aims to develop a GPE testbed. The initiative undertaken in the scope of the PIONIER National Program [7] started in 2001 and is funded by the State Committee for Scientific Research and Sun Microsystems Poland. The demo version of the PROGRESS HPC Portal testbed was presented at the Supercomputing 2002 (SC2002) conference in Baltimore. In this paper we discuss the grid service provider layer of the PROGRESS architecture. At first, we analyze the implemented GPE architectures. Then we introduce a solution to the problem of inflexibility of the up-to-date deployed systems. To illustrate the GSP

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2657, pp. 255–263, 2003.

functionality better, in Section 4 we discuss the grid job management problem within this newly distinguished layer of the GPE architecture. Finally, we describe the testbed implementation of the GSP and other modules constituting the PROGRESS GPE which was presented during SC2002 and is currently working on-line at http://progress.psnc.pl/portal.

2 Grid-Portal Environments

One of the most successful stories about GPEs is being written by the National Partnership for Advanced Computational Infrastructure (NPACI). Their HotPage Grid Computing Portal has been online for a few years now. Originally designed by the SDSC, it is an implementation of the GridPort infrastructure. GridPort comes as a collection of Perl modules that provide back-end grid functionality to web portals. Installation of the toolkit allows for building a portal on top of it. The grid access environment created this way enables file transfer, command execution and job submission.

Another GPE with a portal implemented in Perl technology is the Legion Grid Portal. Developed at the University of Virginia, it facilitates access to the grid. The LGP was deployed to employ the Legion worldwide grid. The logic of the LGP is a Perl CGI script, which is used to process most of the user requests. Its role is to issue Legion commands on behalf of the user. The script also uses some PHP modules, especially those accessing legacy databases. The LGP may be deployed for interaction with any underlying grid infrastructure, for example Globus.

NASA is developing the Information Power Grid, which aims to provide the basic framework for resource sharing and management across sites. The IPG version 1.0 includes Launchpad v. 1 computing portal, which provides access to grid services, provides access to IPG for unsophisticated user and maintains user profile information. The Launchpad enables submitting jobs to “batch” compute engines, executing commands on compute resources, transferring files between two systems, obtaining status on systems and jobs, and modifying the environment of the user. The IPG portal was created using the GPDK, which is under development at Lawrence Berkeley National Laboratory.

GPDK includes the library of core service beans, a central servlet and a collection of demo template web pages. The service beans are implemented in the J2EE technology and use the Java Commodity Grid (CoG) toolkit, which provides a pure Java API to Globus services. The template web pages include HTML and JSP and may be customized for the needs of a particular installation. The GPDK provides security, job submission, file transfer and information services.

All GPE solutions presented herewith have a common feature: the web portal user interface is integrated completely with the methods or libraries allowing for interaction with the grid infrastructure. The GUI and the data presented in it comes from one and the same application server. Such architecture is not flexible enough, therefore the PROGRESS research team applied a different approach. The reasons for developing a new GPE structure and the answer to the problem itself are presented in the next section.

3 Grid Service Provider

According to our knowledge, grid-portal environments that are currently deployed around the world are implemented as 3-tier systems with HPC resources at the bottom, grid management systems (GMS) in the middle and portals in the top tier (see Fig. 1.). This architecture works, but seems to be not flexible enough. When creating a new portal (using the GridPort for example) administrators are required to build both – the presentation and logical layers of the environment. Thus, there are as many installations of the logical layer as there are portals. This approach is also not very useful for business and market applications. The PROGRESS grid-portal environment which is under development in PSNC introduces a new solution to this problem. Functions of the logical layer of the portal are grouped into a separated module running independently of the other modules in the environment. This module is called a grid service provider (GSP) and adds another tier to the system, right above the GMS and below the portal itself. The GSP works as a web services factory. A similar architecture is under implementation at Indiana University and SDSC [8]. The portal in the new architecture serves as the user interface and implements just the presentation functions.

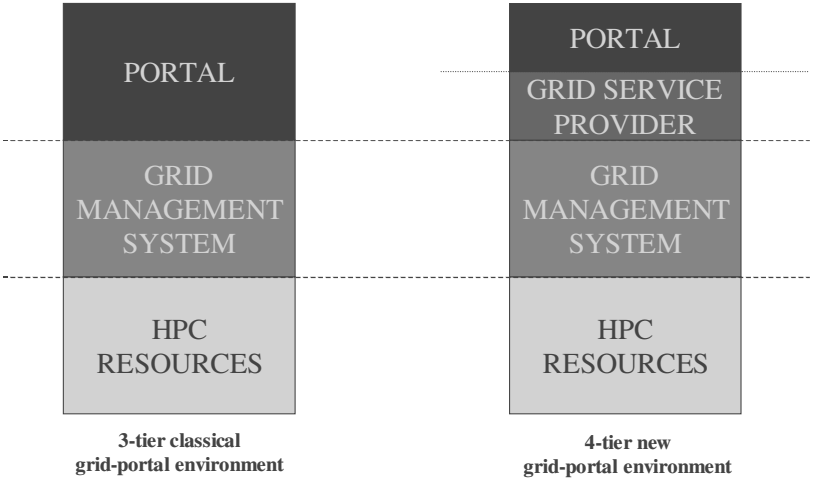

Fig. 1. Grid-portal environment architectures

Using a GSP within the grid access environment makes the use of the grid resources most comfortable to the end users. The GSP serves as the source of grid resources: computing power, grid enabled applications, collaboration tools and others, to web portal and other user interfaces. Installed on a separated server connected to the grid access network the web services based GSP allows for easy building of numerous portals and other user interfaces. Users of those interfaces can switch from one to another and use the same GSP services. This enables them to manage their grid jobs or run applications coming from one application factory with the use of two or more different interfaces. What is more, it is easy to imagine creating various thematic scientific web portals: a bioX portal (like the one deployed within the PROGRESS project), an electric engineering portal and a physics portal. Those

thematic web portals may also use the same grid services and resources available through the grid service provider. The next advantage is the possibility of providing all GSP clients with computing resources belonging to two or more different grids. All these resources can again be available through one and the same interface – the grid service provider and can again be easily utilized by the computing web portals clients.

A typical GSP should provide users with three main services: a job submission service (JS), an application management service (AM) and a provider management service (PM). The JS is responsible for managing the creation of user jobs, their submission to the grid and the monitoring of their execution (through reverse reporting performed by the GMS about events connected with the execution of jobs). The AM provides functions for storing information about applications available for running in the grid. One of its main features is the possibility of assisting application developers in adding new applications to the application factory. Finally, the PM allows the GSP administrator to keep up-to-date information on the services available within the provider. A detailed discussion of the functionality of those services is presented in Section 5.

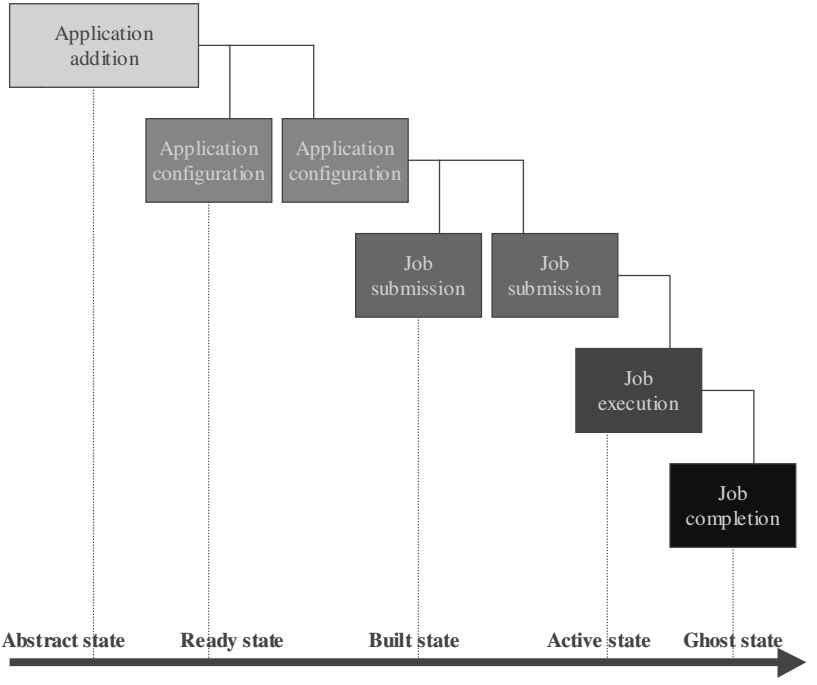

Fig. 2. Grid job state cycle within the grid service provider

4 Grid Job Management within the Grid Service Provider

One of the GSP functions is grid job management. This function is performed by both JS and AM services. The place, where the information about the current state of a grid job is available, is the GSP. It is a natural place to store job states along with job descriptors. Jobs within the GSP exist in five different states. For our approach we assumed the definition of four grid job states introduced in [9]. Additionally, we propose the fifth state – built.

Fig. 2. illustrates the grid job state cycle within the GSP. A job created on top of an unconfigured application is in the abstract state. When the user configures the job by setting values to application parameters and arguments, the state of the job changes to ready. A job may also be created already in the ready state if the user chooses one of the available multiple configurations of the application. A configured job with added references to input and output files is then analyzed by the JS service and the job definition is prepared; the job is in the built state. The GSP contacts the GMS and the execution of the job starts. The job is in the active state until its completion, when it turns to the ghost state.

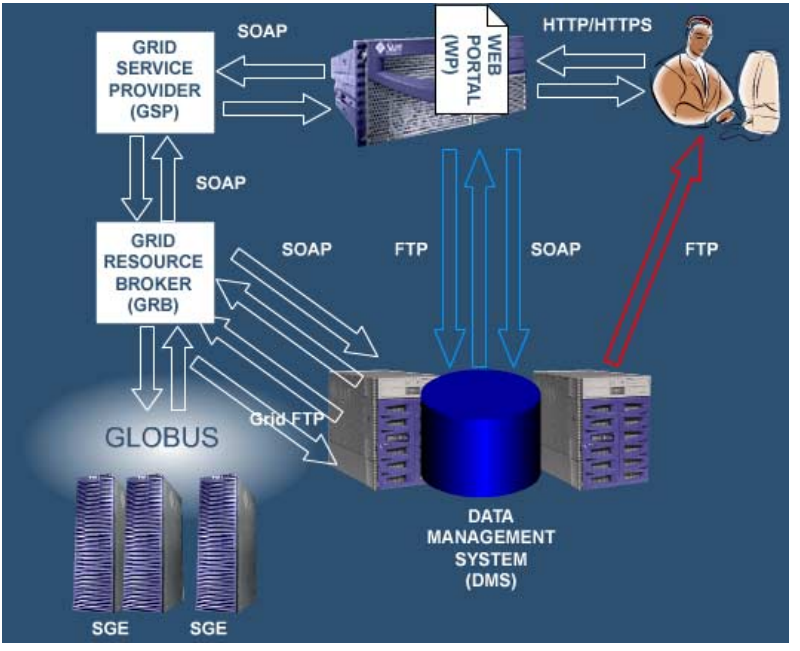

Fig. 3. PROGRESS architecture and communication

5 PROGRESS Testbed

The GSP, which is under development at PSNC within the PROGRESS project. is one of the crucial items of the PROGRESS grid-portal testbed. The demo of the

testbed was presented at SC2002 and the prototype of the PROGRESS GPE is on-line since then. We describe the functionality of the system which is currently accessible at http://progress.psnc.pl/portal/ in this section. Along with the GSP, four other modules constitute the PROGRESS system: the web portal (WP) [10], the migrating desktop (MD) [11], the data management system (DMS) [12] and the grid resource broker (GRB) [13]. The PROGRESS grid is powered by three Sun Fire 6800 servers, placed in Poznan and Cracow and connected by the Polish National Research Network. Fig. 3. illustrates the PROGRESS architecture and the communication manners between modules.

5.1 PROGRESS Grid Service Provider

The PROGRESS GSP is a web services factory with J2EE business logic applied upon the relational database system. It provides four services: the three main GSP services (JS, AM and PM) plus an example of an informational service, the news service, intended for use by web portals. The PRORESS GSP services are to be used by two client interfaces, that is the WP and the MD, and by the GRB. The DMS serves as the source of job input and the destination for job output files.

The JS service delivers functions for computing job building, submitting them to the grid for execution and viewing the results. Methods for building jobs allow for creating jobs, adding and editing job tasks, setting values of application arguments, providing references to job input and output files, and setting the required grid.

The JS prepares a job description using the XRSL language [14]. The description is then transferred to the GRB for the execution of the job. When a job is in the active state its current execution status can be monitored by contacting the GRB in request of the execution information. Additionally, the GRB reverse reports on grid events connected with the job. Finally, the results of the job are available through obtaining references to output files.

The SC2002 demo supported preparing jobs with a single execution of an application. This was a good base for adding functionality of building sequenced and paralleled jobs with multiple tasks. We implemented this feature recently. The JS Service currently enables executing simple, single-task grid jobs and jobs based on workflows. Each workflow can contain several tasks in any configuration. Tasks may be organized in sequences and parallels. It is possible for the JS to enable nesting tasks to an unlimited number of levels in the future. It is not only possible to build "workflowed" grid jobs using the PROGRESS JS service, but also to create a job using a so called virtual application. Virtual applications are in fact applications consisting of several (or more) components. Each component is an application available for execution in a single-task job and the virtual application descriptor contains information on the structure of workflow to be performed. A task descriptor, relating either to the task of a single-task job or a task of a sequenced or paralleled one, contains a reference to an application. Each task descriptor references exactly one application and the reference comes from the application factory managed by the AM service.

As it was already mentioned, the AM service manages the PROGRESS application factory. An application descriptor contains a reference to the application executable. It is either a reference to a file stored in the DMS or a path to a binary on grid computing server filesystems. Each application is also described by a set of available

(required or optional) arguments, required environment variables and required input and output files. Application arguments, environment variables, and additionally its grid resource requirements, may all be assigned values. These values may be different for different application configurations. When an application is unconfigured (the default values of arguments and other parameters are not set) it may be a base for creating a job in the abstract state. When the application configuration is entered, a user may use it for starting a new job in the ready state. Application configurations are stored and seen as independent applications. One executable may be referenced in as many applications (or application configurations) as required.

The provider management service enables keeping up-to-date information on services available in the GSP. The GSP services are web services therefore their descriptors include information on the URL at which the service is available, the service namespace reference (URN) and the service WSDL reference. Some PROGRESS GSP services may have multiple instances. If a service can have multiple instances, it is allowed to create, remove and manage its own instances. Each instance of a service may be understood as a separate virtual resource within the service with its own space in the service database. An example of an instance enabled service within the PROGRESS GSP is the news service.

5.2 Other PROGRESS Modules

The PROGRESS GSP services are accessible through the WP. It provides the functionality of:

- grid job management: creating, building, submitting, monitoring execution and analyzing results,
- application management,
- provider management,
- portal news reading and editing,
- DMS file system management: listing directory tree, uploading and downloading, as well as deleting and renaming of files.

The MD, which is a separate Java client application, is currently under implementation. It will provide a user interface for grid job management and DMS file system management.

The GRB enables the execution of PROGRESS grid jobs in a cluster of three Sun computers. The cluster is managed by Sun Grid Engine software with Globus deployed upon it. The GRB provides two interfaces: a CORBA interface and a web services one. The SC2002 PROGRESS demo used the former one, but this will be changed to the latter in the final deployment. Grid job definitions are passed to the GRB in a form of an XRSL document and the GRB informs the JS about events connected with the execution of a job (start, failure or success).

PROGRESS grid jobs use the DMS to store the input and output files. The DMS provides a web services based data broker, which handles all requests. The DMS is equipped with three data containers – the file system, the database system and the tape storage system containers. The DMS prepared for SC2002 demo used the first one, others will be added continuously. A data file is referenced within the DMS with a universal object identifier. The identifier allows for obtaining information on the

location of the file. Each file may be downloaded or uploaded using one of three possible protocols: FTP, GASS [15] or GridFTP [16].

6 PROGRESS Future

The PROGRESS project will be under development until the end of May 2003. However, the most important functionality has already been reached for SC2002. The current functionality of the PROGRESS GSP was presented in the previous section. We are going to extend the GSP functionality in the future. We believe that the most important additions should affect the web services interfaces of the GSP services; we plan to make them OGSA [17,18] compliant. Other extentions can be implementing new information services for scientific web portals, like link directory and discussion forum. The GSP will also get equipped with an authorization module; detailed security models will be added during the final deployment of the testbed environment.

We believe the PROGRESS implementation of the grid service provider facilitates building grid user interfaces and fulfills the requirements of deployment flexibility, which were set by the PROGRESS project paradigms. However, in our opinion, the OGSA architecture can add even more flexibility to how grids and user interfaces can be developed and operated. Thus, we plan to make the GSP services (and other PROGRESS modules, like the DMS data broker) fully OGSA compliant.

All actions performed by the GSP are conducted on behalf of a particular user. The user actions need to be authorized before their execution. We believe this may be enabled by the GSP authorization module, a security model completely independent of the grid. The GSP authorization module will be based on the Resource Access Decisions model [19]. This opens a way to grant rights to services and to specific resources to users and check whether a given user is allowed to perform a particular action or not.

We concentrated on the PROGRESS GSP in this paper. It is however worth mentioning that all PROGRESS modules are under constant development. They should provide functions for handling the GSP tasks or use functions provided by the GSP itself (the WP must provide necessary user interfaces enabling the management of virtual applications – this interface is under implementation). Finally, the DMS architecture assumes it will be a completely distributed system with multiple data containers and, last but not least, we are going to equip the PROGRESS GPE with means for the visualization of scientific experiments run in the grid with the use of PROGRESS applications.

References

1. Thomas, M., Mock, S., Boisseau, J., Dahan, M., Mueller, K., Sutton, D.: The GridPort Toolkit Architecture for Building Grid Portals. Proceedings of the Tenth IEEE International Symposium On High Performance Distributed Computing (2001)
2. NPACI HotPage Grid Computing Portal. Accessed at https://hotpage.npaci.edu/
3. Natrajan, A., Nguyen-Tuong, A., Humphrey, M. A., Grimshaw, S.: The Legion Grid Portal. Accessed from http://legion.virginia.edu/papers.html
4. Information Power Grid. Accessed from http://www.ipg.nasa.gov/

5. Novotny, J.: The Grid Portal Development Kit. Accessed from http://www.cogkits.org/
6. Kosiedowski, M., Mazurek, C., Stroinski, M.: PROGRESS – Access Environment to Computational Services Performed by Cluster of Sun Systems. Proceedings of the 2nd Cracow Grid Workshop, Cracow Poland (2002) 45–56
7. Rychlewski, J., Weglarz, J., Starzak, S., Stroinski, M., Nakonieczny, M.: PIONIER: Polish Optical Internet. Proceedings of ISThmus 2000 Research and Development for the Information Society conference Poznan Poland (2000) 19–28
8. Pierce, M., Fox, G., Youn, Ch., Mock, S., Mueller, K., Balsoy, O.: Interoperable Web Services for Computational Portals. Proceedings of Supercomputing 2002 Baltimore
9. Haupt, T.: Grid Job. Available from the Grid Computing Environments Research Group, Global Grid Forum
10. PROGRESS HPC Portal. Accessed from http://progress.psnc.pl/portal
11. Kupczyk, M., Lichwala, R., Meyer, N., Palak, B., Plociennik, M, Wolniewicz, P.: Roaming Access and Migrating Desktop. Proceedings of the 2nd Cracow Grid Workshop, Cracow Poland (2002) 148–154
12. Grzybowski, P., Mazurek, C., Spychala, P., Wolski, M.: Data Management System for grid and portal services. Accessed from http://progress.psnc.pl/
13. Kurowski, K., Nabrzyski, J., Pukacki, J.: User Preference Driven Multiobjective Resource Management in Grid Environments. Proceedings of CCGRID 2001 conference Brisbane Australia (2001)
14. XRSL Language. Accessed from http://progress.psnc.pl/xrsl/
15. Global Access and Secondary Storage (GASS). Accessed from http://www-fp.globus.org/gass/
16. The GridFTP Protocol and Software. Accessed from http://www-fp.globus.org/datagrid/gridftp.html
17. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG Global Grid Forum. Accessed from http://www.globus.org/research/papers.html (2002)
18. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C.: Grid Service Specification. Open Grid Service Infrastructure WG Global Grid Forum. Accessed from http://www.globus.org/research/papers.html (2002)
19. Resource Access Decision, Version 1. Accessed from http://www.omg.org/technology/documents/formal/resource_access_decision.htm