PROGRESS – Access Environment to Computational Services Performed by Cluster of Sun Systems

Michał Kosiedowski, Cezary Mazurek, Maciej Stroiński[1)]

[1)]Poznan Supercomputing and Networking Center
Noskowskiego 10, 61-704 Poznan, Poland

Abstract

The PROGRESS project aims at building an access environment to computational services performed by a cluster of SUN systems. PROGRESS integrates different parts of grid middleware. The job submission is handled by the Grid Broker. Data required for computations are managed by the Data Management System. Additionally, any user interface may use the Grid Service Provider developed within the project. The computing portal is an example of such an interface and was provided within the project. Another example is the migrating desktop. One of the most important parts of the realised grid-portal architecture is the grid security. The PROGRESS project enables complete grid-portal architecture for further deployments in different fields of grid enabled applications. Currently, a couple of bioinformatics computing applications are provided to run the tested environment. The whole software architecture consists of middleware which is already available (e.g. Globus, Sun Grid Engine) as well as tools and services developed within the project workpackages. Most of those components communicate with each other through interfaces based on Web Services and are distributed within the testbed installation. The testbed installation consists of three Sun Fire 6800 and two Sun Fire V880 systems installed in Poznan and Krakow. The current status of the project work will be described on the basis of the first testbed installation, which was presented during the Supercomputing 2002 Exposition.

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 the development and deployment of different grid components. A driving force for the development of such next generation computing infrastructure are scientific applications. The advancement of those computing environments is always related to the development of its three particular components: processing within the grid infrastructure, communication through advanced networks and access to computational services from the computing portals. Especially for services which handle access through the computing portals it is useful to identify common components that are needed and can be reused by

different grid-oriented applications. The availability of common and reusable components will accelerate and enhance the development of similar efforts in many application areas.

Poznan Supercomputing and Networking Center, Poland, together with Sun Microsystems, Poland, have been developing the PROGRESS project co-funded by the Polish PIONIER programme. Other PROGRESS participants are: the Academic Computing Centre Cyfronet of Krakow, and the Technical University of Lodz.

The **PIONIER** programme, issued by the State Committee for Scientific Research, Poland, for the years 2001 to 2005, aims at the development of the Polish Optical Network PIONIER [1]. The architecture of the PROGRESS project [2] is based on a really distributed cluster of Sun Fire 6800 servers connected with this network. A couple of fast (1Gb/s or 10 Gb/s), λ-based, dedicated channels for communication between cluster nodes will be enabled for the PROGRESS project testbed. Sun Fire servers have already been installed in Poznan and Krakow.

PROGRESS integrates several parts of grid middleware and therefore ensures that the complete grid-portal architecture will be developed for future deployments in different fields of grid enabled applications. The whole architecture consists of middleware which is already available (like Globus or Sun Grid Engine) as well as tools and services developed within the project workpackages and concerning grid services management, security, internal data management, visualisation and mobile access. Two kinds of the user interface provided by the portal and the migrating desktop application are built on top of the Service Provider layer. This layer provides the user authentication and authorisation, the management services for computing applications and other services, the computing job submission service and the module for communication with the Computing Broker.

A detailed description of particular components of the grid-portal environment being built in PROGRESS is presented in the following chapters.

2 Architecture and Functionality

The general system architecture of the PROGRESS grid-portal environment has been illustrated on Fig. 1. The main module of the Progress HPC Portal is the grid service provider (GSP). It is a new layer introduced to the grid-portal environment architecture by the PROGRESS research team. The GSP allows users to create, submit and execute their grid jobs using applications available in the PROGRESS application factory. Additionally, the PROGRESS GSP provides informational services intended for use by web portals and management services intended for GSP administrators.

The PROGRESS GSP services are accessible through two client interfaces: the web portal (WP) and the migrating desktop module (MD). The WP provides functionality of: grid job management, application and provider management, portal news reading and editing as well as DMS file system management. The MD, which is a separate Java client application, provides user interfaces for grid job management and DMS file system management.

The GSP transmits grid jobs definitions to the grid resource broker (GRB) for running in a cluster of three Sun computers. The cluster is managed by Sun Grid Engine software [3] with Globus [4] deployed upon it. The GRB is responsible for

computing resources brokering and grid job execution controlling. The GRB strictly cooperates with the DMS to download the job input data and upload the job results basing on the data files identifiers obtained from the job definition as well as reports to the GSP about events connected with the execution of jobs (start, failure or success).

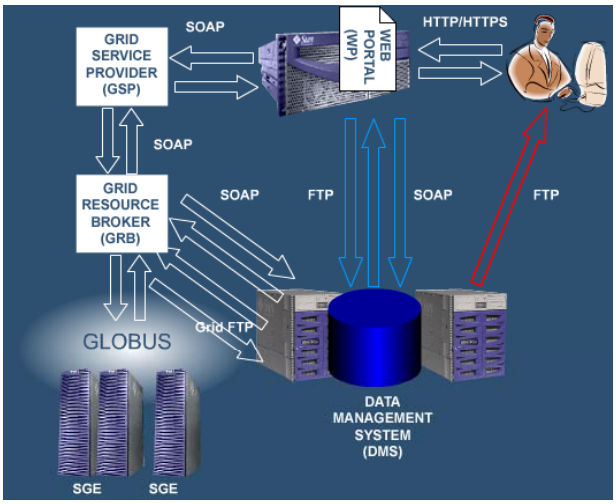

Fig. 1 General system architecture of PROGRESS

The DMS in PROGRESS is used to store the grid job input and output files. A data file is referenced within the DMS and the whole PROGRESS environment with a unique object identifier. The identifier allows for obtaining information on the location of the file in the distributed system. Each file may be downloaded or uploaded using different protocols; the WP uses the FTP, the MD and the GRB use GridFTP for data transmission.

3 Cluster configuration

Concerning the testbed configuration the project aims at building a really distributed computing cluster of Sun systems presented in Tab. 1.

Tab. 1 The configuration of the PROGRESS cluster for the testbed

	PSNC Poznań	Cyfronet Kraków
Computing servers	Sun Fire 6800 (24 proc., 24GB RAM) Sun Fire 6800 (24 proc., 24GB RAM)	Sun Fire 6800 (24 proc., 24GB RAM)
Data Management Servers	Sun Fire V880 (4 proc., 8GB RAM))	Sun Fire V880 (4 proc., 8GB RAM)
Front-end server	Sun Fire 280R (2 proc., 8GB RAM)	
Storages	Sun StorEdge T3 (9*36 GB, RAID5) Sun StorEdge T3 (9*36 GB, RAID5) Sun StorEdge T3+ (9*36 GB, RAID5)	Sun StorEdge T3 (9*36 GB, RAID5) Sun StorEdge T3 (9*36 GB, RAID5)

Both computational servers in Poznan are divided into logical domains. One has two domains and the other has three of them. In Krakow, however, the sole

computational server is divided into two domains. Currently, from the user's point of view, the whole installation is seen as four separated systems (the rest of the domains is used as code development and testing platform). The Sun Grid Engine (SGE) and Globus Toolkit 2.0 were installed on all domains. Both those packages make up a basis for development and running environment of the project.

As for now all four domains are running independently i.e. computational jobs can be submitted directly to each domain. It is a broker developed in this project, which is responsible for resource allocation. Globus package is an interface to each domain. It enables execution of computational jobs based on available hardware resources using its own mechanisms or schedules jobs execution to SGE. In the next stage some domains will become part of the cluster created based on SGE. Then, in that configuration and on a newly created cluster a single installation of Globus software will act as a job execution interface on a domains that is part of a cluster. Then all submitted jobs will be directed for execution and management to SGE. This approach will enable full usage of the SGE potential and will also provide the means to test the developed broker in different environments (group of independent domains and group of independent domains in conjunction with cluster(s)). Work on this subject is scheduled for the year 2003.

At present the development platform consists of MPICH-G2 software. Plans for the future include the usage of MPI, which is part of Sun HPC ClusterTools.

A fast optical network works as a foundation on which high performance computational systems can be interconnected. Its scalable architecture makes it possible to extend the created cluster in order to increase the computational power. Aiming to assure that functionality, all local interconnections between computational servers and data management systems are done basing on Gigabit Ethernet technology in both locations (i.e. Krakow and Poznan). Connection between locations is based on a dedicated link POL-622 (Polish NREN) made in the ATM technology. On the IP layer all system that are part of a computational cluster are using a distinct address class to achieve a logical separation form the Internet. The front-end server is connected to Poznan Metropolitan Area Network (POZMAN) by Fast Ethernet link. This configuration becomes obsolete with the advent of PIONIER optical network that will enable setting up a direct optical link between Poznan and Krakow.

4 Grid-portal environment

4.1 Grid service management system

The main tasks of this PROGRESS component is enabling the effective management of user job execution and the resource management in an advanced, distributed grid environment. The user will be endowed with the opportunity of defining all required attributes and possible resource constraints for his applications. Moreover, the precedence constraints between tasks might be specified providing the mechanisms for the definition of advanced computing experiments including monitoring of the particular tasks execution. To improve the performance of the whole system a resource monitoring system based on Jini/Jiro technologies has been included. The designed architecture is complied with the standards defined by the

Scheduling and Resource Management workgroup (SCHED-WG), established within the Grid Forum [5]. The architecture of the system is presented on Fig. 2

The system provides the functionality of the Grid Resource Broker by means of two fundamental functions:

- submitJob – a function for job submission basing on the XRSL definition of tasks,
- getJobId – returns the job identifier .

The main task of the broker is to read the job description in XRSL [6], call functions for parsing, checking its correctness and taking the decision concerning resource allocation for particular task execution. Remote task execution is handled by the GRAM protocol from the Globus system. Advanced procedures and algorithms for scheduling and resource management have been developed and integrated within the broker architecture; more information can be found in [7]. The broker is cooperating with two other modules: resource discovery and job manager.

According to the PROGRESS project, the timeline in the near future the MDS service from Globus currently used for resource discovery will be replaced by the monitoring system based on Jini/Jiro technologies.

The job management module is the next piece of the grid service management system. It provides the following functions:

- getJobStatus – a function which returns the status of the job;
- jobCancel – a function which removes the job;
- jobSuspend – a function which suspends the job execution;
- jobResume – a function which resumes the suspended job.

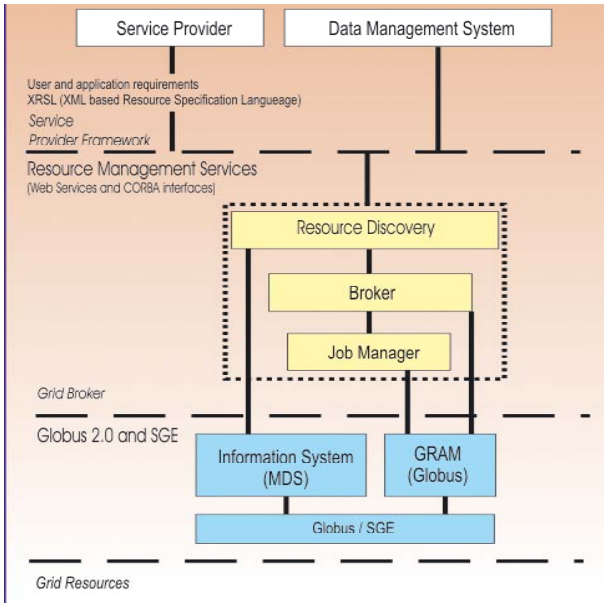

Fig. 2 Architecture of the grid service management system

The Job status describes the current state of the job in the resource distribution system. At the current project lifetime moment it determines if the job is being computed or has been finished.

4.2 Security management

One of the most important security components developed in PROGRESS grid-portal environment is the VALKYRIE Intruder Detection System. Its main feature is the realtime detection of intruders. VALKYRIE is a combined system which implements advanced mechanisms and algorithms of known IDSs as well as simple mechanisms for reaction on detected breakdowns. The System will ensure control and monitoring of designated systems and guarantee immediate revealing of any attack attempts. The system architecture is complied with the CIDF model [8]. The complete system description can be found in [9]

4.3 Data Management System

The Data Management System contains three logically distinguished modules: Data Broker, Metadata Repository and Data Transporter . The functionality of those modules will be described beneath.

Data Broker (DB) is designed to be the main access point to the DMS resources and services. The basic tasks that it delivers are as follows:

- asynchronous responding to the clients requests, without blocking access to those services for others clients,
- fulfilling the security policy on the repository elements level (access to data files, directories),
- passing on the clients requests to the metadata repository
- collecting and sending back the results to clients

DB is a module that mediates in the flow of all requests directed to the DMS. There is no possibility for external application to pass direct requests to the repository or even data transporters.

Within the confines of work over the DMS the technology of communication between DB and others DMS modules was designed and initiated. The idea of this mechanism can be illustrated basing on a single client request description:

- at first the DB authorizes the client that submitted the request. To this end DB uses remote authorisation module to verify that the given user is authorized to access specific resource.
- after a successful verification process DB prepares an appropriate query (basing on the clients request) and sends it to the central metadata repository.
- the received results are transferred to the client.

All the above-mentioned operations are executed in a distributed agent environment basing on the SOAP protocol for exchanging communicates (Web Service technology).

Metadata Repository (MR) is the main element of the DMS. Here it stores the following sorts of information:

- metadata about resources: data files, its physical localisation and possible way to access them,

- metadata about rights: all information related to the rights – users, their groups, access rights.
- metadata describing the standards of file description, e.g. *Dublin Core* (*DC*)

Access to the repository resources is services by the Metadata Management (MM) module. The main task that it fulfils is collecting data broker requests and creating answers according to the accessible knowledge (metadata) and also basing on the state of the data transporters and information from them.

MM via DB module makes available the following kind of services:

- catalogue-based services –for creating a metacatalogue, removing it or moving to another place in the structure within all its contents;
- file-based services – for adding, deleting, renaming file and additionally determining the physical localisation of the file and accessing the file;
- security services – to determine if the final user is entitled to a given operation on specific resource according to the metadata information.

There is one instance of the MM module in the DMS. It is motivated by the fact that it stores and manages the critical information about the metacatalogue structure, user data and security policy for the whole DMS.

Data Transporter module is responsible for delivering space on the storage resources that stays under its control and manages the operation of placing data files inside its resources and accessing them from those resources on demand. Operations executed by this module are connected with the making of reservation for the data planned to be placed on the transporter resources, blocking data files for access (and allowing access to the data file) and accessing the information about the state of the whole transporter module and separate data file staying under its control. Keeping in mind the fact that the transporter was designed to collaborate with other elements of the DMS it implements the internal functionality using interfaces, which allows to communicate with the other DMS modules. Implemented elements are made available (similarly to other modules of the DMS) as web services that can be called with the use of a SOAP protocol for exchanging communicates (the data transporter is controlled in that way). Direct data access is ensured by standard data transfer protocols used in the grid environments: the FTP protocol, GASS protocol (Grid Access to Secondary Storage) [10] as a standard and secure version, GridFTP protocol [11] (gsiftp – the enhance of the standard FTP protocol with the Globus Security Infrastructure).

A separate element of the DMS is an SRS system installation. The SRS System is a crucial data environment module, which stores the biological sequences essential for the computes executes? on the portal. The system uses a pilot installation in Poznan and it is configured with the use of indexed biological data banks. In addition a set of scripts was prepared for keeping the local copy of biological data up to date. The user interface of the SRS system is accessible at http://srs.man.poznan.pl/

The overall architecture of the Data Management System in PROGRESS has been presented on Fig. 3

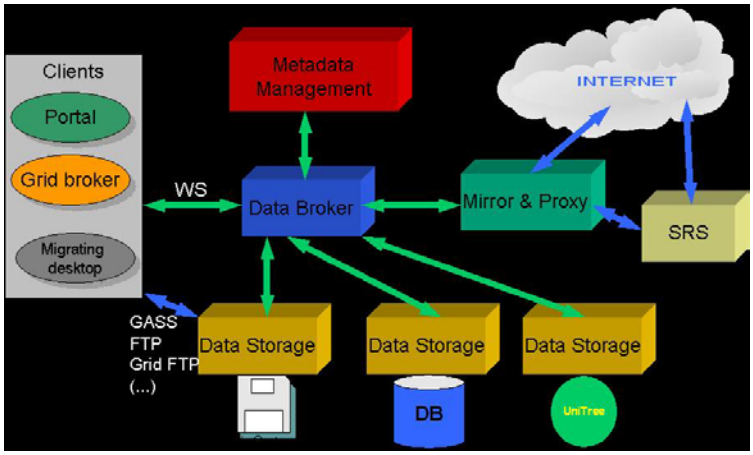

Fig. 3 Data Management System Architecture

4.4 Grid Service Provider and Application Management

In the scope of the Grid Service Provider and Application Management Workpackage a new architecture for grid-portal environments has been proposed. This architecture assumes separating the presentation functions of the portal from logical processing functions (that is separating the user interface from services enabling grid access) and introducing a new item – the grid service provider. The grid service provider is completely independent of user interfaces and the grid resources management system (GMS). It only provides functionality allowing to utilise grid resources from the level of multiple user interfaces and communicates with the GMS in order to execute grid jobs submitted by users. Users may switch between different service provider access interfaces and use the same services (especially the computing job submission service). Such a solution allows to manage grid jobs and run applications collected within one service provider from the level of one or more user interfaces. In PROGRESS such two independent user interfaces are the web portal and the migrating desktop.

It is worth mentioning that research teams dealing with similar problems, which are of interest in the scope of the PROGRESS project, have also noticed the need for separating a similar layer in their grid system access environments. During the Supercomputing 2002 conference in Baltimore an idea of a system constructed alike PROGRESS was presented. This solution is under implementation at SDSC and Indiana University [12].

The grid service provider is realised as a set of services with Web Services interfaces allowing to access functions prepared in the EJB technology (version 2.0 specification) [13]. The data obtained through the invoking of methods accumulated within the GSP are presented in the web portal environment. Presentation modules complementary to services are implemented as content providers for channels of a portal prepared in Sun ONE Portal Server [14] software environment (currently version 6.0 is in use).

Currently, the service provider allows to utilise one of four implemented services. These are:

- the computing job submission service, which allows to create, submit, monitor the execution process and access job results,
- the application management service, which allows to manage computing applications available from the application factory of the service provider,
- the provider management service, which allows to keep up-to-date information about services available within the service provider,
- the short messages services, which is an example of an informational service intended for web portals (other examples of this type services may be link directory or discussion forum services).

The testbed presented during the Supercomputing 2002 exhibition in Baltimore (Fig. 4) included the web portal (implemented in the Sun ONE Portal Server 6.0 environment), which allowed to utilize the services for: computing job submission, service provider management and short messages reading and writing as well as to manage data collected in the data management system.

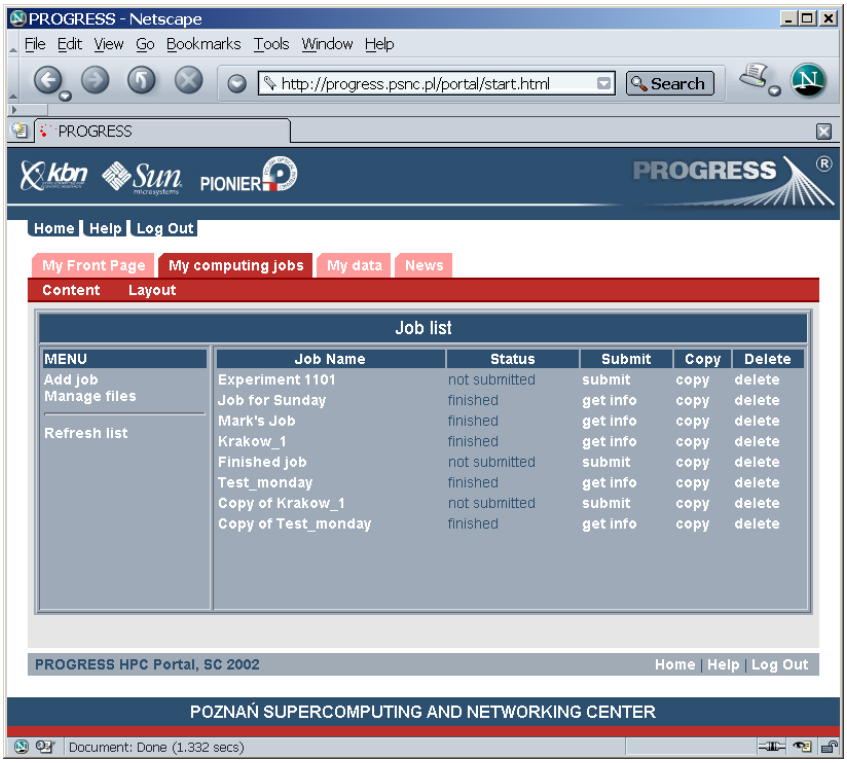

Fig. 4 PROGRESS portal testbed during Supercomputing 2002

5 Related work

There are several great projects aiming at the designing and development of grid-portal architecture.

National Partnership for Advanced Computational Infrastructure (NPACI) has developed HotPage Grid Computing Portal [15] which has been online for a few years now. Originally designed by the SDSC, it is an implementation of the GridPort infrastructure. GridPort [16] 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 project with a portal implemented in Perl technology is the Legion Grid Portal [17]. 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 [18], 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 [19], 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.

6 Conclusions

In many available grid-portal environments one common feature might be easy defined: the web portal user interface is often integrated completely with the interaction mechanisms for the grid infrastructure. The user interface and the data presented in it come from one and the same application server. Such architecture is not flexible enough, particularly in business solutions. Therefore, the PROGRESS research team applied a different approach presented in the paper. The whole software architecture consists of middleware which is already available (e.g. Globus, Sun Grid Engine, HPC Cluster Tools) as well as tools and services developed within project workpackages. Most of those components communicate with each other through interfaces based on Web Services and are distributed within the testbed installation.

Beyond the overall architecture approach all particular components of the system are very important itself if advanced grid environments are concerned. The job submission is handled by the Grid Broker. Data required for computations are managed by the Data Management System. Additionally, any user interface may use the Grid Service Provider developed within the project. The computing portal is an example of such an interface and was provided within the project. Another example is the migrating desktop. And finally, one of the most important parts of the realized grid-portal architecture is the Grid security.

It is also worth mentioning that before the end of the project (May 2003) all developed elements will be tested and practically used by bioinformatics computational applications. They can run in the grid being submitted directly from the computational portal as it was presented during the SC2002 testbed when two of them could be started from Baltimore and computations were performed either in Poznan or in Krakow.

There is also one additional profit of PROGRESS which was assumed before the project started in December 2000. The project will enable the created grid-portal environment for other advanced applications to become a product off-the-shelf. It is the main task for the deployment period which comes after R&D part is finished.

References

1 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), pp. 19-28

2 PROGRESS website. Accessed from http://progress.psnc.pl/

3 http://www.sun.com/software/gridware/

4 http://www.globus.org/

5 http://www.gridforum.org/

6 XRSL Grid Job Definitions. Accessed from http://progress.psnc.pl/xrsl/

7 Kurowski, K., Nabrzyski, J., Pukacki, J.: User Preference Driven Multiobjective Resource Management in Grid Environments. Proceedings of CCGRID 2001 conference (2001) Brisbane Australia

8 S. Staniford-Chen, Common intrusion detection framework, March 1998, http://seclab.cs.ucdavis.edu/cidf

9 Chmielewski M., Gowdiak A., Fonrobert S., Meyer N., Ostwald T.: VALIS/Valkyrie. To be published in Proceedings of Cracow Grid Workshop Cracow Poland (2002)

10 Global Access and Secondary Storage (GASS). Accessed from http://www-fp.globus.org/gass/

11 The GridFTP Protocol and Software. Accessed from http://www-fp.globus.org/datagrid/gridftp.html

12 Pierce, M., Fox, G., Youn, Ch., Mock, S., Mueller, K., Balsoy, O.: Interoperable Web Services for Computational Portals. Proceedings of Supercomputing 2002 Baltimore (2002)

13 Enterprise JavaBeans Technology. Accessed from http://java.sun.com/products/ejb/

14 Sun Open Net Environment (Sun ONE). Accessed from http://wwws.sun.com/software/sunone/

15 NPACI HotPage Grid Computing Portal. Accessed at https://hotpage.npaci.edu/

16 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)

17 Natrajan, A., Nguyen-Tuong, A., Humphrey, M. A., Grimshaw, S.: The Legion Grid Portal. Accessed from http://legion.virginia.edu/papers.html
18 Information Power Grid. Accessed from http://www.ipg.nasa.gov/
19 Novotny, J.: The Grid Portal Development Kit. Accessed from http://www.cogkits.org/