EZ-Grid: Integrated Resource Brokerage Services for Computational Grids

Barbara M. Chapman, Babu Sundaram, Kiran K. Thyagaraja
Department of Computer Science,
University of Houston, Houston. TX. 77204
{chapman, babu, kiran}@cs.uh.edu

Extended Abstract

1. Introduction

With the advent of high-speed networks that can couple wide area compute resources, storage devices, workstations and high-end visualization devices and people, solving large-scale problems has become a reality. Typically, such problems involve handling huge data and require very large amounts of compute power. Computational grids [13][14][16][17] have emerged to provide the infrastructures for dependable, consistent and pervasive access to high-end resources. They promote coordinated resource sharing to solve huge distributed problems in science and engineering. Existing middleware services such as the Globus Project [12] assist application developers like us to construct such computational grids to harness and collaboratively share isolated compute resources. It offers standard services such as security [15], resource management [10], information services [9] and data handling [1][2][7].

Current user interaction with grids is relatively low-level. Job submission is a cumbersome process that typically requires the use of scripting languages or similar mechanisms. Further, information on available resources, as needed in order to identify suitable resources for job submission, is only partially available in most grid information systems. In this project, we develop services that assist users to easily interact with computational grid environments. We feel that a solution can be achieved via a single interface that hides the internals of the middleware services, providing resource information and a brokerage service to assist in the resource selection and job submission process. A prototype of the EZ-Grid system has been designed and implemented to address the above issues.

The EZ-Grid system has been developed for a typical grid user. A grid user can be any end user who requires access to grid resources to submit a job or an administrator who desires more control over the shared resource. Our system aims to eliminate the complexity in grids by providing services like automatic resource discovery, help in preparing the user's jobs by profiling the application, efficient resource usage through brokering and transparent file transfer between the grid resources. For the administrators, EZ-Grid provides the policy framework to help them enforce fine-grained usage policy based authorization for the users of their resources.

The rest of the abstract is organized as follows. In the next section, the overview of the prototype is given, following which we describe the system in some detail, and then we focus on related work and conclusion.

2. Overview

The EZ-Grid project aims at promoting efficient job execution and controlled resource sharing across sites. This would involve the construction of resource brokerage systems coupled with policy frameworks and information subsystems. To focus on this aspect, we have identified the

following goals for the EZ-Grid system.

a) Developing generic brokerage systems for multi-site computing: Grid environments demand us to perform effective scheduling by assigning the job requests with appropriate resources in order to satisfy time and/or cost constraints. Such a matchmaking process requires knowledge current usage permissions on the shared resources, history information about previous job submissions, dynamic resource information status and so on. To design and implement scheduling mechanisms that analyze the relative quantitative and qualitative importance of these information to obtain resource choices is a big challenge.

b) Enforcing absolute control and priority for resource providers: Resource providers need absolute control and priority over their resources when committing them to a grid. They need mechanisms to allow them to specify usage permissions/restrictions that specify and control how other users gain access to the resources. Further, the resource choices for a job execution highly depend on the availability of the remote resource as determined by the policies of the remote site. Our prototype provides with a basic framework that allows the resource providers to specify and enforce usage policies and the users to examine them and make appropriate resource choices.

c) Graphical user interface: Graphical user interfaces are superior to command line based user interfaces. They are more intuitive and require very little training to accomplish most tasks and also can increase the productivity of the user. The EZ-Grid system prototype provides an interface developed in Java to leverage portability in a pool of heterogeneous resources in the grid environment.

d) Gathering and managing dynamic resource status information: Existing brokerage systems rely their decisions upon information services that provide only static resource information. EZ-Grid system utilizes dynamic resource status information provided by local schedulers or workload management systems. Information provided by schedulers about work queue status and current load on the resource is considered in making resource choices.

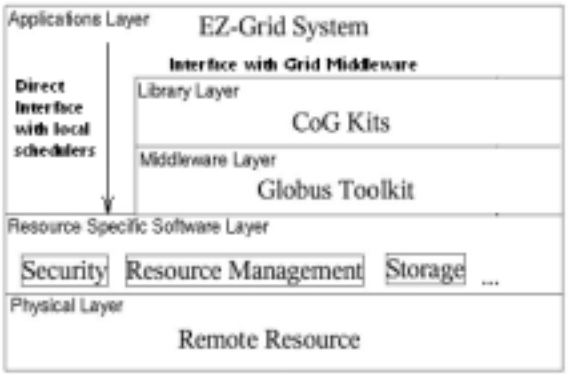

Figure1. Illustration of the EZ-Grid system stack.

Figure 1 indicates the interaction of EZ-Grid system with other middleware tools to exploit the underlying resources like parallel machines and clusters. Interaction of our system with Globus middleware tools is achieved through CoG kit [22] libraries. Direct interfaces with local schedulers like Sun GridEngine [43] is required to obtain scheduler specific configuration details and queue information.

3. Components of EZ-Grid System

The EZ-Grid system is a tool to enable more efficient and easier use of the grid environments. It is based on a well-defined user interface designed in Java. It is primarily composed of two subsystems based on their functionality: Client subsystem and information server systems. The components of the EZ-Grid system are illustrated in the figure below:

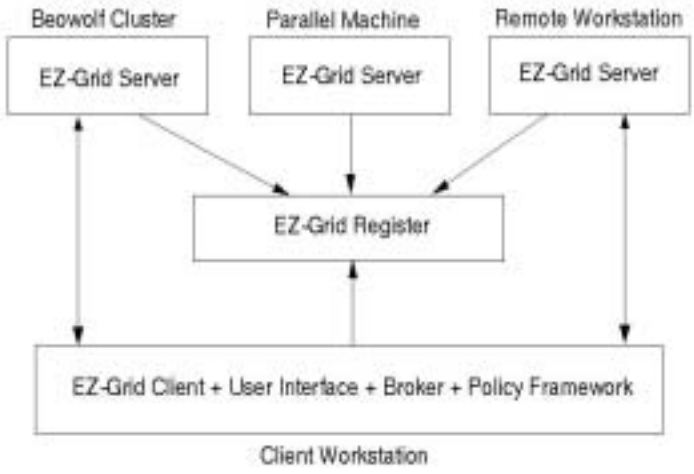

Figure 2. Components of EZ-Grid

There are three entities in the EZ-Grid system - the client component, EZ-Grid server and the EZ-Grid Register. The client subsystem comprises of the user interface, brokerage system and the policy framework. Components of the server subsystem, that form the information system, include the EZ-Grid server, EZ-Grid register and a GUI that assists the administrators in configuring the server components.

These components are discussed in some details in the following sections and discussed in more detail in the full paper.

3.1 Client subsystem

EZ-Grid client system acts a grid user's gateway to the grid resources. The client component that is instantiated on the user's workstation and enables the following important tasks:

1. Globus proxy creation through GSI, if a valid proxy does not exist. It allows the user to seamlessly establish his identity across all his grid resources.
2. Allowing the user to make job specifications such as the resource requirements of the job itself, budget/deadline constraints associated and so on.
3. Examining and reporting resource availability to the user or brokerage system in real time. This is accomplished by examining the user's credentials and also authenticating with the remote machine
4. Managing resource information, both static and dynamic attributes such as operating systems, CPU and queue information. This allows the user to view the status information about the grid resources at any given time.
5. Policy-based authorization and accounting to examine and evaluate usage policies of the

resource providers. Such a model is very critical while brokering for appropriate resources in a heterogeneous grid.

6. Resource brokering to match the abstract job specifications made by the user to those grid resources that would best satisfy the time and/or cost constraints.
7. Storing and retrieving relevant application profile information, history of job executions and related information. Application profiles can be composed on the client side and then exported to the remote resource to characterize the applications.
8. Transparent file transfer. Files residing on a third party machine will be transferred to the target machine before the job starts executing.

Besides the system being easy-to-use and providing single interface for job submission and monitoring in grids, the novel features of the client-side system lies in the policy framework and the sophisticated resource brokerage system.

3.1.1 Policy Engine Framework

Policy engine [31] is an automated framework that promotes policy-based authorization or access control and accounting for job execution in computational grids. This model allows the resource providers to dictate their resource usage through policies. For instance, a sample policy could read as "remote users are not allowed to access the resources during weekdays" or as "users belonging to group 'research' are given higher priority than other users". Such a framework gives fine-grained control to resource owners in highly dynamic resource sharing environment.

There are two major aspects to this model.

- One aspect would be to develop utilities and services for resource providers to specify usage policies for their shared resources over the grid. This would allow them to exercise absolute control and enforce authorization for all resource requests. The services would be flexible enough to enforce fine-grained user-level policies as well as global policies that apply to all users.

- The other aspect would be to develop tools to assist the user-side brokerage systems to fetch and examine remote site usage permissions to make better resource choices. Further, flexible cost models could be accommodated to enable accounting between the grid user and the remote site.

The model has been designed to be flexible enough to minimize the coordination efforts required between the two major modules mentioned above. For instance, the administrators should be able to change the policies dynamically at will and the user-side framework must ensure that the recent changes are checked for and that the appropriate policies are evaluated transparently prior to making resource choices. We have designed the policy expression in the form of attribute-value pairs in XML [35]. The feasibility of other policy languages such as extensible rights markup language (XrML) [45] is also being examined.

3.1.2 EZ-Grid Resource Broker

Efficient resource sharing to optimize throughput and increase resource availability is an inherent requirement of computational grids. However, achieving efficient job execution in a grid environment constrained by deadline and budget is a complicated task. The brokering system is designed to address this problem. The resource brokerage system uses the information subsystem, policy framework and the scheduler specific interfaces to make resource choices based on the job

specifications. The goal of this brokering process could be budget-based or deadline-based or both. The broker would then be expected to arrive at resource choices that would provide quickest execution time or least cost execution. However, the entire process of arriving at appropriate resource choices based on resource status information and history information is an extremely huge and complicated problem. Brokering process is split into a series of stages in which the potential set of resources is progressively narrowed down to ultimately identify the best resource for the given job specification.

We start with filtering based on static resource information, which needs no interaction with the remote resources. Then, the policy-based evaluation is done which again might need only little or no interaction at all depending on the changes of the remote policies, if any. After these initial stages are carried out, the resources are evaluated for network parameters like available bandwidth and latency that are critical for applications that require migration of huge data sets. Then, the history information about previous job submissions are fetched from the remote resource and analyzed. Finally, the scheduler interfaces at the remote resource are queried to obtain dynamic information about the work queues and the current loads in order to estimate job run times.

3.1.3 User Interface

Graphical user interfaces are more intuitive and require very little training to accomplish most tasks. Our current choice for a suitable programming language is Java and we use it as a front-end tool and a programming language that gives us an opportunity to develop quickly. Our choice of language was further strengthened with the availability of CoG Kits. Services currently used via CoG kits are job submission, security, data transfer and Information querying.

Figure 3. Job specification interface

The user interface, apart from being user friendly, enables the preparation of jobs, job monitoring, profiling applications, viewing resources and viewing information on resources. Job submission interface performs useful jobs like automated job preparation scripting, transparent file transfer, manual or broker assisted job submission, etc.

3.2 Server Subsystem

Existing systems such as Grid Information Services (GIS) [9] and Network weather service (NWS) [33] provide some of the information that is needed for standard tasks in computational grids, such as job submission. But their services are incomplete. Globus mostly provides static attributes of hardware resources such as CPU, memory, hostname information, basic software such as operating system information, and some dynamic information like CPU load snapshot. NWS provides additional dynamic information since it is able to obtain CPU loads and bandwidth and latency measurements. However, we argue that dynamic information about scheduler queues is critical to instrument effective brokerage systems. Static information on queues might include names of queues, their type, limits on jobs that may be submitted to them and the number of possible jobs. Dynamic information might include the number of jobs waiting in a queue and the status of running jobs. Further, history of previous job submissions, history of queue status information and application profile information need to stored and supplied to broker on demand for better scheduling. This is the motivation to have our own information system.

3.2.1 EZ-Grid Register

The EZ-Grid Register provides automatic resource discovery mechanism for a grid user and broker. It is designed to act as an index server, capable of providing availability information about the hosts that have registered with it. The EZ-Grid client subsystem, when instantiated by the grid user, queries the grid register to obtain the available set of resources. Grid resources running the EZ-Grid Servers are configured to send heartbeats to the EZ-Grid Register once every "N" time steps, where the value of "N" is configurable by the administrator of the remote resource. In our current grid setup at University of Houston, we have set the heartbeat time interval to be 6 hours.

3.2.2 EZ-Grid Server

The EZ-Grid Server is concerned with storage and retrieval of information for the brokerage system to use. It is based on the distributed server model and each information server resides on the remote resource working in tandem with GIS and, if available, NWS. This is the ideal model for information dissemination as the user can access resource information from any workstation.

Currently, the EZ-Grid Server stores and retrieves the following information for the broker residing on the client side:

- Static and dynamic resource information
- Usage policies and application profile information
- History information about job submissions and queue status

All the data are stored as XML files to leverage on its simplicity and portability without imposing additional dependence on EZ-Grid system. XML, with its widespread acceptance and standards, provides a promising choice to be the language for expressing, storing and retrieving information in a grid environment. The availability of Java-based APIs [53] for parsing and managing XML files provides an added incentive. Other mechanisms such as LDAP and databases would introduce additional dependencies to our system and hence were not considered.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <EZ-Grid>
  - <kiran>
      <Name>MG</Name>
      <Version>1</Version>
      <Author>NAS Bentchmarks</Author>
      <Absolute_Path>/export/home/kiran/RunMG</Absolute_Path>
      <No_of_Procs>4</No_of_Procs>
      <Memory_in_MB>512</Memory_in_MB>
      <Architectures>Sun Solaris</Architectures>
      <Arguments>(no of processors) (class A or S)</Arguments>
      <Job_Type>OpenMP</Job_Type>
      <Input_Files>none</Input_Files>
      <Output_Files>Standard Output</Output_Files>
    </MG>
    - <FT>
      <Name>FT</Name>
      <Version>1</Version>
      <Author>NAS Bentchmarks</Author>
      <Absolute_Path>/export/home/kiran/RunFT</Absolute_Path>
      <No_of_Procs>4</No_of_Procs>
      <Memory_in_MB>512</Memory_in_MB>
      <Architectures>Sun Solaris</Architectures>
      <Arguments>(no of processors)(class A or S)</Arguments>
      <Job_Type>OpenMP</Job_Type>
      <Input_Files>none</Input_Files>
      <Output_Files>Standard Output</Output_Files>
    </FT>
  </kiran>
  - <babu>
    - <SP>
      <Name>SP</Name>
      <Version>1</Version>
      <Author>NAS Benchmarks</Author>
```

```
            <Absolute_Path>/export/home/babu/RunSP</Absolute_Path>
            <No_of_Procs>4</No_of_Procs>
            <Memory_in_MB>512</Memory_in_MB>
            <Architectures>Sun Solaris</Architectures>
            <Arguments>(no of processors)(class A or S)</Arguments>
            <Job_Type>OpenMP</Job_Type>
            <Input_Files>none</Input_Files>
            <Output_Files>Standard Output</Output_Files>
        </SP>
      + <Comet>
            <Name>Comet</Name>
            <Version>1.001</Version>
            <Author>AK</Author>
            <Absolute_Path>/export/home/kiran/RunComet</Absolute_Path>
            <No_of_Procs>4</No_of_Procs>
            <Memory_in_MB>256</Memory_in_MB>
            <Architectures>Sun Solaris</Architectures>
            <Arguments>(tutorial no.)(number of threads)</Arguments>
            <Job_Type>OpenMP</Job_Type>
            <Input_Files>none</Input_Files>
            <Output_Files>Standard Output File</Output_Files>
        </Comet>
    </babu>
</EZ-Grid>
```

Figure 4. A sample application profile information file with data belonging to two users.

4. Related Work

Nimrod/G [6] and Application Level Schedulers (AppLeS) [3][4] are existing systems that support the scheduling of applications in the grid environment. Nimrod/G broker aims at solving applications that run on wide variety of parameters or input sets. Automatic resource discovery is not provided; a list of resources in the form of Globus gatekeeper contact strings has to be set up manually by the user before brokering. The brokerage system relies on Globus' GIS to gather information about remote resources. Although GIS provides sufficient static resource information and a little dynamic information like CPU load snapshot, it does not have local resource manager queue information, bandwidth details between nodes in a cluster, etc. Hence the resource selection process relies mainly on static information about the resource. Our broker analyzes static information mentioned above as well as dynamic information such as job queue information, usage policies, historical information about job executions and so on.

AppLeS is an application-centric scheduler for metacomputing applications. AppLeS agents provide a mechanism for scheduling applications over the resources based on static and dynamic resource information provided by NWS. Our work again differs from AppLeS in that we analyze application-specific information (such as application profiles, history of executions), dynamic resource-specific information (usage control policies, availability) and scheduler-specific information (such as work queue lengths and loads) to achieve better schedules. We also support budget/deadline based scheduling.

The features that distinguish our work from the existing resource brokerage systems are:

- Automated resource discovery mechanism for information about resource availability at any time
- Generic brokerage system that is not coupled to any specific application and that analyzes historical information for run time prediction
- Dynamic information gathering and management such as job queue details, scheduler-specific configuration information and so on.
- Usage policy frameworks for resource providers/administrators as well as users to enable fine-grained authorization.

5. Conclusion

The EZ-Grid system aims at enabling efficient use of grids by both end users and administrators. It uses a sophisticated brokering system coupled with usage policy framework and a distributed information subsystem to achieve user specified time/cost constraints. We have started to deploy our prototype in our campus grid environment and plan to get feedback from pilot users and improve our system. We expect that our work will lead to better understanding of practical resource brokerage approaches in a grid environment.

References

[1] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data Management and Transfer in High-Performance Computational Grid Environments," Parallel Computing 2001.

[2] J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," Sixth Workshop on I/O in Parallel and Distributed Systems, May 5, 1999.

[3] F. Berman, R. Wolski, S. Figueira, J. Schopf, G. Shao, "Application-Level Scheduling on Distributed Heterogeneous Networks," Proceedings of Supercomputing '96, 1996.

[4] F. Berman and R. Wolski, "The AppLeS Project: A Status Report ", Proceedings of the 8th NEC Research Symposium, Berlin, Germany, May 1997.

[5] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch, "A National-Scale Authentication Infrastructure," IEEE Computer, 2000.

[6] R. Buyya, D. Abramson, and J. Giddy, “Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid,” The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China. IEEE Computer Society Press, USA, 2000.

[7] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, "The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets," Journal of Network and Computer Applications, 2001.

[8] B. M. Chapman, B. Sundaram, K. Thyagaraja, S.W. Masood, P. Narayanasamy, "EZGrid system: A Resource broker for Grids," http://www.cs.uh.edu/~ezgrid

[9] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid Information Services for Distributed Resource Sharing," 2001.

[10] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[11] K. Czajkowski, I. Foster, C. Kesselman, "Co-allocation Services for Computational Grids," Proceedings of the 8th IEEE Symposium on High Performance Distributed Computing, 1999.

[12] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," International Journal of Supercomputer Applications, Summer 1997.

[13] I. Foster and C. Kesselman, "The GRID: Blueprint for a new Computing Infrastructure," Morgan Kauffman Publishers, 1999.

[14] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of Supercomputer Applications, 15(3), 2001.

[15] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids," ACM Conference on Computers and Security, 1998, 83-91.

[16] I. Foster, C. Kesselman, S. Tuecke, S, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.

[17] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An open grid services architecture for distributed systems integration," http://www.globus.org/ogsa

[18] J. Frey, I. Foster, M. Livny, T. Tannenbaum, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," University of Wisconsin Madison, 2001.

[19] G. Gheorghiu, T. Ryutov, B. C. Neuman, "Authorization for Metacomputing applications," Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998

[20] A. S. Grimshaw, Wm. A. Wulf and the Legion team, Legion: The next logical step toward Worldwide Virtual Computer, Communications of the ACM, January 1997.

[21] A. S. Grimshaw, A. Ferrari, F. C. Knabe, M. Humphrey, "Wide-Area Computing: Resource Sharing on a Large Scale," IEEE Computer, 32 (5). 29-37. 1999.

[22] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids," ACM 2000 Java Grande Conference, 2000.

[23] M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," Proceedings of the 8th International Conference of Distributed Computing Systems, pages 104-111, June, 1988.

[24] A. Natrajan, A. Nguyen-Tuong, M. Humphrey, A. Grimshaw, "The Legion Grid Portal," Grid Computing Environments 2001, Concurrency and Computation: Practice and Experience

[25] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing," Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July, 1998.

[26] R. Raman, M. Livny, and M. Solomon, "Resource Management through Multilateral Matchmaking", Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), Pittsburgh, Pennsylvania, August 2000, pp 290-291.

[27] T. Ryutov, B. C. Neuman, "Representation and Evaluation of Security policies for Distributed system Services," Proceedings of the DARPA Information Survivability Conference & Exposition, January 2000

[28] W. Smith, V. Taylor, I. Foster, Using run time predictions to estimate queue wait times to improve scheduler performance, Proceedings of the IPPS/SPDP '99 Workshop on Job Scheduling Strategies for Parallel Processing, 1999.

[29] W. Smith, V. Taylor, I. Foster, Predicting application run times using historical information, Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[30] J. Steiner, B. C. Neuman, J. Schiller, "Kerberos: An Authentication System for Open Network Systems," Proceedings of Usenix Conference, 1988, 191-202.

[31] B. Sundaram, B. M. Chapman, "Policy Engine: A Framework for Authorization, Accounting Policy Specification and Evaluation in Grids," 2nd International Conference on Grid Computing, Nov 2001.

[32] B. Sundaram, C. Nebergall, S. Tuecke, "Policy Specification and Restricted Delegation in Globus Proxies," Research Gem Presentation, SC2000, Oct 2000.

[33] R. Wolski, "Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service," Proceedings of the 6th IEEE Symp. on High Performance Distributed Computing, Portland, Oregon, 1997.

[34] “DUROC - The Dynamically updated request online coallocator," http://www.globus.orog/duroc

[35] "Extensible Markup Language," http://www.w3c.org/XML

[36] Global Grid Forum Steering Group, Argonne National Laboratory, Illinois, http://www.gridforum.org

[37] "GRB - The Grid Resource Broker," http://sara.unile.it/grb/grb.html

[38] HotPage - Grid Computing Portal," http://hotpage.npaci.edu

[39] "Java API for XML Processing," http://java.sun.com/xml/jaxp/index.html

[40] LoadLeveler, http://www.rs6000.ibm.com/software/sp_products/loadlev.html

[41] Portable Batch System, http://www.openpbs.org

[42] Resource Specification Language, RSL, http://www.globus.org/gram/rsl_spec1.html

[43] Sun Grid Engine, Sun Microsystems, http://www.sun.com/software/gridware

[44] Symphony - Work in progress paper, http://csgrad.cs.vt.edu/~mlorch/symphony.html

[45] XrML, Extensible Rights Markup Language, http://www.xrml.org