G-Monitor: A Web Portal for Monitoring and Steering Application Execution on Global Grids

Martin Placek and Rajkumar Buyya

Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
Email: mplac@students.cs.mu.oz.au and raj@cs.mu.oz.au

Abstract. *As Grids are emerging as the next-generation computing platform, the need for Web-based portals that hide low level details of accessing Grid services for deployment and execution management of applications is increasing. We have responded to this requirement by developing an easy to use, lightweight, scalable, web portal called G-Monitor that allows the user to monitor, control, and steer execution of application jobs on Global Grids. Unlike related systems, the users of G-monitor delegate the responsibility of selection of appropriate resources for execution of their application jobs to the broker.*

1 Introduction

Grids provide the infrastructure to harness a heterogeneous environment comprising of geographically distributed computer domains, to form a massive computing environment through which large scale problems can be solved. For this to be achieved, Grids need to support various tools and technologies that can support: security, uniform access, resource management, scheduling, application composition, computational economy and accounting [3][7]. Additionally, Grids need provide ubiquitous user-interface that hides complexities associated with the deployment and management of execution of application jobs on distributed resources.

A sample wide-area Grid computing environment is shown in Figure 1. In this environment, the Grid consumers interact with GRBs (Grid Resource Brokers) such as Nimrod-G [7] responsible for scheduling applications on distributed resources based on their availability, cost, capability, and user-specified QoS (Quality of Service) requirements. The users need a pervasive interface that allows them to monitor, control,

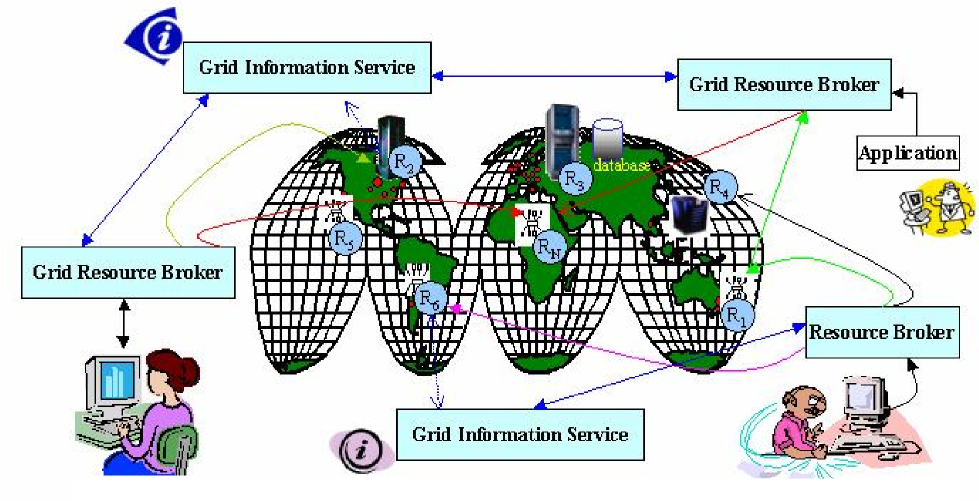


Figure 1: A Sample Grid computing Environment and key components.

and steer the execution of their applications with ease. To serve this need we propose a Web-based portal, called G-monitor, which is developed as part of the Gridbus project [8].

G-Monitor not only aims to provide an easy to use interface, but also a lightweight interface. Unlike heavyweight GUI (graphical user interface) applications, Web-based portals provide a pervasive and light-weight interface. In addition, exporting GUI-based application interfaces from a remote resource to the user desktop is infeasible when exposed to low-bandwidth and high-latency wide area network environments. We have already experienced such difficulties with a simple Nimrod-G monitoring client implemented as GUI application.

G-Monitor interacts with the Nimrod-G GRB (Grid Resource Broker), which is implemented using low-level services provided by middleware such as Globus [4] and Legion [5]. It provides a consistent interface that is easy to use, enabling the end-user to monitor, control, and steer execution of application jobs running within the Grid environment. G-Monitor is flexible enough to be run from anywhere without the need for custom client software. It enables the user to:

1. **Retrieve and set QoS (quality of service) parameters, such as**
   - Deadline, budget, optimisation preferences
2. **Monitor/Control Jobs Information, such as**
   - Start, Stop, grid node status, execution time
3. **Monitor Resource status, such as**
   - Server name, Host name, Service cost, status, remarks
4. **Monitor Experiment status, such as**
   - Deadline, Budget, Job status, resource status

This paper provides an insight into how G-Monitor fits into the Grid Infrastructure and how it interacts with surrounding components. We discuss G-Monitor's architecture, design, functionality, implementation, application and conclude by providing possible future enhancements.

2 Related Work

A number of projects have explored development of toolkits for development of Grid portals and construction of application specific portals. Some representative efforts include GridPort [9] and HotPage [10] from San Diego Supercomputing Centre, GPDK (Grid Portal Development Kit) [14] from Lawrence Berkeley National Laboratory, Legion portal [18] from the University of Virginia, GRB portal [13] from University of Lecce, SGE Technical Portal [11] from Sun Microsystems, and PBSWeb[12] from the University of Alberta. GridPort and GPDK allow construction of application-specific portals by providing generic interfaces/libraries for accessing Grid resources using Globus. Legion portal provides an interface to Legion Grid infrastructure. As these toolkits provide an interface to low-level Grid services, the users of portals constructed using them are responsible for manually identifying resources that are suitable for running their applications and deploying them. The same is the case with the GRB portal as it also provides Web-based interface to Grid resources running Globus and again users are responsible for selection of suitable resources for the execution of their applications. Unlike these systems, the unique aspect of G-monitor is that it has been designed to provide access to high-level Grid services (e.g., the Nimrod-G broker and Gridbus scheduler). As high-level Grid services (e.g., Nimrod-G) are in turn implemented using low-level Grid services (e.g., Globus), they hide issues related to the identification of resources that are suitable for running user applications and their aggregation.

SGE Portal and PBSWeb basically provide Web-portal for accessing cluster resources management systems, Sun Grid Engine (SGE) and PBS (Portal Batch System) respectively.

3 Architecture and Design

This section has been split up into the following two sections:

- Architecture: In this section we get an understanding as to how G-Monitor fits into the Grid Architecture.
- Design: We take a look at how G-Monitor itself works, detailing the main modules which make up the implementation of G-Monitor. An account of how G-Monitor's modules interact is also provided via the use of a work-flow example.

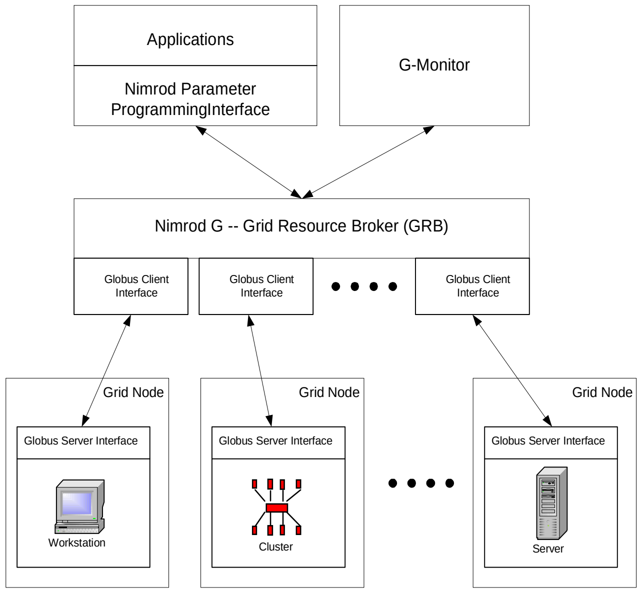


Figure 2: G-Monitor : Grid Architecture.

3.1 Architecture

The architecture of G-Monitor and its interaction with other components within a Grid Architecture[19] framework is shown in Figure 2. The user is able to start experiments with the use of "Applications" which interface into Nimrod G using the "Nimrod Parameter Programming Interface". Once the user has started their experiment they are able to monitor and control their experiment with the use of G-Monitor. G-Monitor is able to provide a high level interface by utilising tools such as Nimrod G and Globus to handle all the low-level requirements of handling Grid Resources. Nimrod G plays a broker role by farming out tasks to each grid node using the "Globus Client Interface". Globus is used by Nimrod G for submitting jobs, authentication, security and gathering information about resources. Grid Resources provide a hybrid environment which Globus and Nimrod G play a vital role in managing and ultimately providing a unified interface which G-Monitor utilises.

G-Monitor provides a web portal, which is used by the end-user and therefore is categorised as a component within the "Applications and Portals" category. G-Monitor resides on a web server, which sits between the GRB and the Web browser. The Grid consumer uses the Web browser to access G-Monitor's functionality. G-Monitor serves the users requests by communicating with the GRB. The GRB manages all the Grid nodes using low-level Grid middleware services, keeping a detailed database of information about the status of the Grid nodes whilst offering scheduling and farming out facilities.

3.2 Design

The five key components of G-monitor (as shown in Figure 3) are:

1. **Web Server Software**: Receives http requests made by the web browser, determines which of the "Function Modules" to execute and serves the HTML generated by the "HTML Generator" back to the web browser.
2. **HTML Generator**: Receives calls from the "Function Modules" to generate HTML containing specified data. The HTML generated is then passed to the Web Server software.

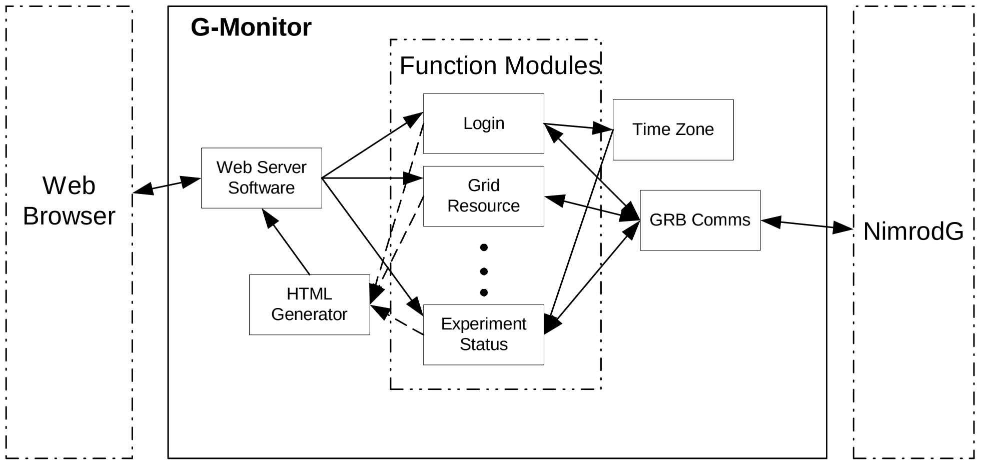


Figure 3: G-Monitor : Design.

3. **Function Modules**: A set of modules which provide a one-to-one mapping with the web pages offered by G-Monitor. Function Modules include:
   - Login: Handles user logins
   - Grid Resource: Responsible for listing status of all Grid resources.
   - QoS Parameters: Enables user to set QoS parameters and control the experiment execution.
   - Job Information: Provides user with information relating to each job.
   - Experiment Status: Gives a global account of how the experiment is progressing.
4. **Time Zone**: This module is called by the login script. Therefore, as the user is logging in, this module will register which timezone the user is coming from. The timezone module is then called by various "Function Modules" which require time conversion.
5. **GRB Communications**: Serves requests from the "Function Modules" by establishing a TCP socket connection with the GRB to retrieve and serve requested information.

To get a better understanding of the interaction between various components we shall walk through a work flow example. The user wishes to retrieve information on the current status of their experiment and uses the web browser to click on a link to take them to the page containing this information. The following events then occur:

1. The Web browser sends the user's request off to the Web Server.
2. The Web Server software receives this request and determines which of the G-Monitor "Function Module" should be executed.
3. The "Function Module" analyses the user requests and calls upon the "GRB Comms" module.
4. The "GRB Comms" module then initiates a TCP socket to the GRB server and uses the GRB protocol (e.g., Nimrod-G job management protocol [2]) to retrieve information about the current status of the user's experiment.
5. Upon receiving the information from the GRB, the "GRB Comms" module passes this information back to the calling "Function Module".
6. The executing "Function Modules" then makes a call to the "HTML generator" which wraps the data in HTML, and serves it back to the Web Server software.
7. The Web server software then serves the HTML back to the web browser.

4 Implementation

One of the main aims of developing G-Monitor was to overcome problems and limitations associated with interfaces with heavy bandwidth requirements. As grid computing is geographically distant in nature the user is likely to be exposed to low-bandwidth and high-latency wide area network environments. In situations such as these, interfaces which have heavy bandwidth requirements run into problems regarding application's responsiveness, leaving the user waiting for information, which by the time is relayed to the user is out of date. As our aim was to build an interface that is light weight a web based solution was proposed, whereby the browser was responsible for building the GUI based upon the HTML it received from the G-Monitor server. Providing a web-based portal had the added benefit of enabling the user to access G-Monitor using their web browser, avoiding the need for installation of any custom software to access G-Monitor. The Apache[1] web server was chosen to host G-Monitor. We utilised Apache's CGI module to call scripts that need to be executed on the G-Monitor server. The scripting language that was to be used in developing G-Monitor needed to be:

- Strong at Parsing: As we need to parse data coming from Nimrod G interface.
- Support TCP/IP Sockets: For communication with the Nimrod G interface.
- Relatively efficient: Gmonitor needs to be lightweight.
- High level in nature: Easy to maintain, rapid development.

The scripting language we decided to use for was Perl, as it suited our outlined requirements. G-Monitor is very much a server side based application with the only exception being a part of the "Time Zone" module. The "Time Zone" module makes a use of client side code to determine which timezone the user is present. The server side component has been implemented in Perl and is responsible for processing user requests, gathering data from the Nimrod G interface and generating a combination of HTML and JavaScript, which is then served back to the end-user. The JavaScript generated by the server side is the only client side executed code and is there to extract user's local time.

Refering back to the Design (Figure 3), all the modules depicted have been implemented using Perl. The "Login" module is the module responsible for gathering the user's local time and passing it onto the "Time Zone" module. Therefore the "Login" module serves both HTML and JavaScript. The "GRB Comms" module uses Nimrod-G job management protocols [2] for interaction with the Nimrod resource broker.

During implementation we came across some hurdles: in particular, time zones. While testing our system locally there were no time zone differences and everything functioned well. Upon running our system from Baltimore (refer to section 5 for more detail) we found the timezone difference between the web browser and the rest of the system was causing confusion in pages which contained time related data. The solution was to create a "TimeZone" module which would be responsible for registering which timezone the user was in and making the time adjustments automatically.

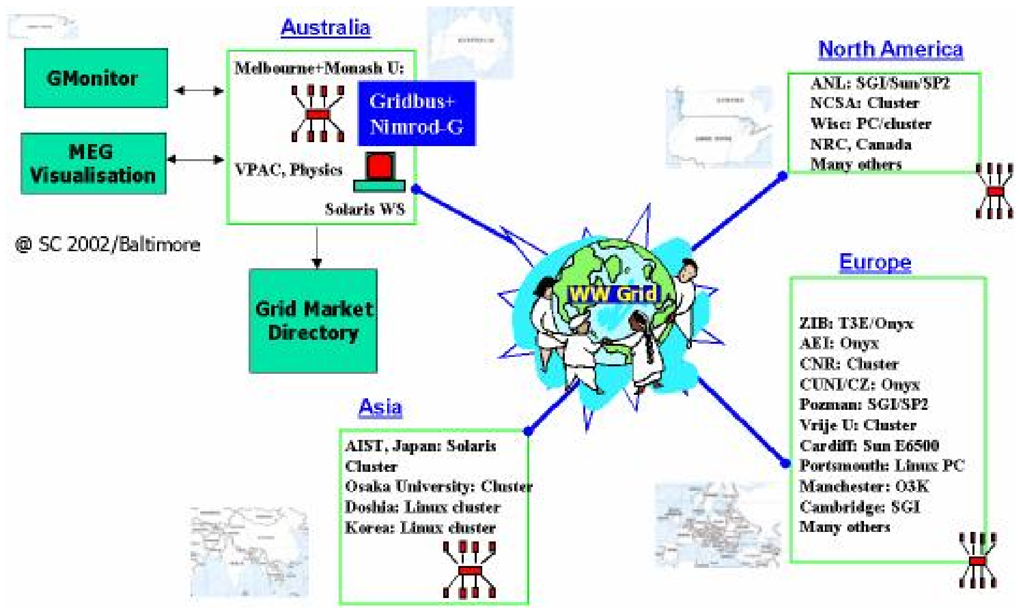


Figure 4: G-Monitor Usage during the HPC Challenge Demo @ SC 2002.

To provide a scalable interface, G-monitor allows users to define entity refresh rate and the number of entities that they are interested in monitoring at an instance. For example, if an application contains thousands of jobs, it is possible for users to define the number of jobs that they like monitor at a particular instance. The users can also monitor based on resources and resource groups.

5 Use Case Study

We have participated in the Global Grid Testbed Collaboration [15] that setup a world-wide distributed Grid testbed and demonstrated capabilities of the state-of-the-art Grid technologies and applications at the SC 2002 conference [16] held in Baltimore, USA. A demonstration setup depicting application deployment and access locations of various Grid entities is shown in Figure 4. The testbed had resources contributed by participating organizations from all over the world. We had access to most resources in the testbed as we were demonstrating one of the four Grid applications as part of this collaboration. As this collaboration fits to the notion of virtual organisation (VO), we used the Gridbus GMD (Grid Market Directory) infrastructure to create a VO registry and added participants as Grid service providers along with their contributed resources/services and their attributes. G-Monitor has been deployed on a machine running an Apache Web Server located in the University of Melbourne (Australia). *G-Monitor was accessed via a web browser in Baltimore (US), while the G-Monitor and Nimrod-G modules resided in Melbourne*. The Grid nodes which were then used in the experiment were scattered across the world.

We have demonstrated distributed analysis of brain activity data—captured using the MEG (Magneto-Encephalo-Graphy) instrument located in Osaka University, Japan—on Global Grids using the Nimrod-G broker [7] with the Gridbus scheduler [17]. The Gridbus scheduler, implemented as a plug-in scheduler for Nimrod-G, identified resources/services along with their attributes such as access-price by querying the GMD. The composition of brain activity analysis application and the results of its deployment on the Grid are reported in [17].

We have been able to use the G-Monitor portal to supply the user QoS parameters (deadline, budget, optimization preference) to the Nimrod-G broker and start the experiment. We were able to monitor progress of the experiment including progress of individual jobs and status of resources. We were also able to steer application execution by changing QoS requirements during the demonstration.

In the following sub-sections we will walk through some user scenarios to demonstrate G-Monitor's features and its usefulness in monitoring and steering application execution in Grid environments.

5.1 User sets QoS parameters

In the process of setting up an Experiment the user has the option to retrieve and set Qos Parameters using the G-Monitor (Figure 5). The user is able to set the Deadline, Budget and Optimisation preference, upon doing so the user is returned information regarding the feasibility of the settings provided. From the same page the user is also able to start, stop and shutdown their experiments.

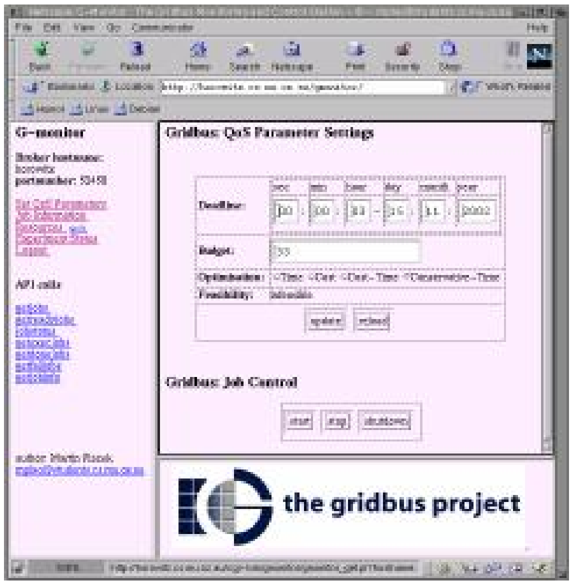
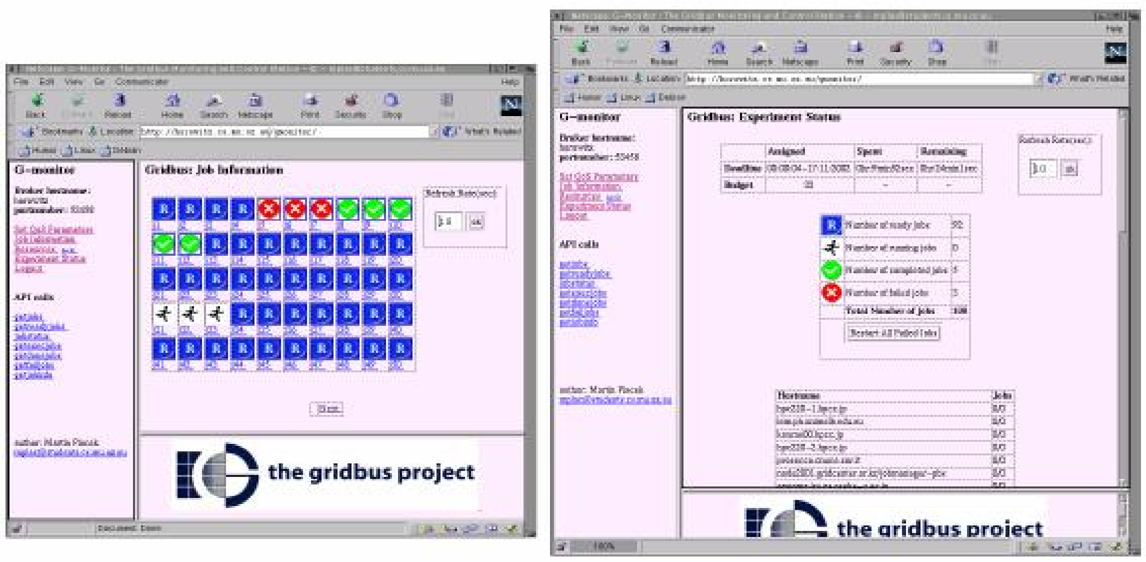


Figure 5: Setting QoS Parameters.

5.2 User experiment status

Once the user has started an experiment, they will be interested in the progress of their application execution. The users can do this by invoking the "Job Information" and "Experiment status" G-monitor Web links (see Figure 6).


Gridbus: Job Information

Gridbus: Experiment Status

	Assigned	Spent	Remaining
Deadline	09:08:04 - 17.11.2002	[illegible]	[illegible]
Budget	22	-	-

Number of ready jobs	92
Number of running jobs	0
Number of completed jobs	5
Number of failed jobs	3
Total Number of jobs	100

Hostname	Jobs
hpc028-1.hpcc.jp	0/0
[illegible]	0/0
[illegible].hpcc.jp	0/0
hpc028-2.hpcc.jp	0/0
[illegible]	0/0
[illegible]	0/0

Figure 6: Job Information and Experiment Status.

The "Job Information" page revolves around providing information about all individual jobs. We can see that each job has a status icon associated with it, giving the user an indication whether the job is "Ready", "Running", "Completed" or "Failed". These icons are links allowing the user to drill down and retrieve more information regarding that job and allow them to have the job restarted.

The "Experiment status" page provides a summary about the status of the entire experiment. The page is broken down into three main sections. The first section provides the user with information about how their experiment is progressing with respect to: Deadline, Budget and the time remaining for completion. The second section summarises the status of all the jobs in the experiment and displays the number of current jobs that are Running, Ready, Completed or Failed. This section also provides the user with the option to restart all the failed jobs. The final section lists all the nodes in the network showing the user how many jobs have been assigned to each grid node and the number of jobs that each node has completed.

5.3 Grid Infrastructure status

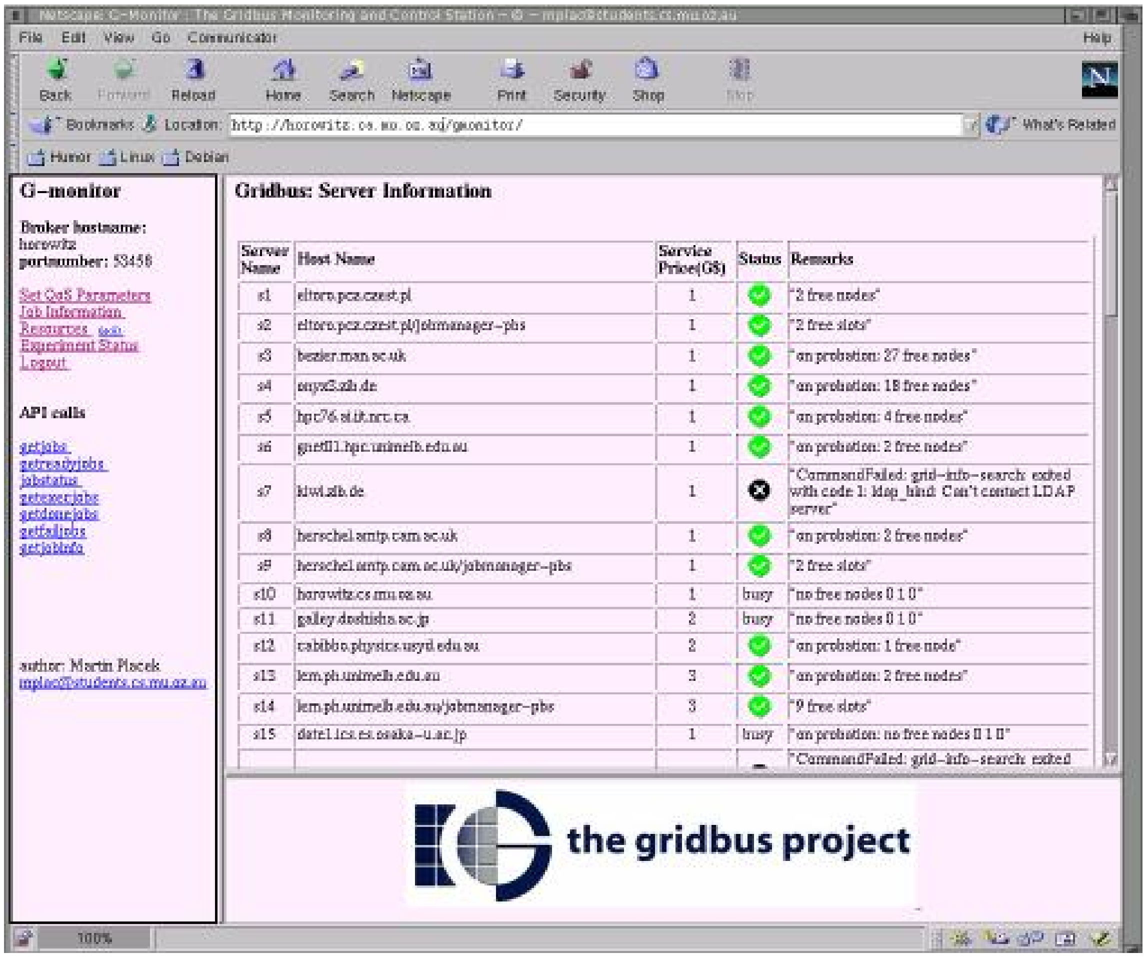
Before running the experiment the user may want to look at what grid nodes are available and how much they cost. If the user is on a tight budget and all the cheaper resources are down, they may decide to wait for them to come back up before starting to execute their experiment. The "Resources" page (see Figure 7) contains a table listing all the resources available to the user and provides information regarding the name of the server as defined within the Nimrod-G environment, the hostname, cost and its status.

6 Conclusion

In this paper we have identified the need for Web-based portals that hide low level details of accessing Grid services for deployment and execution management of applications. We developed a Web portal called G-Monitor that allows the user to monitor, control, and steer execution of application jobs on Global Grids. Unlike related systems, G-monitor provides web-based interface to high-level Grid service (the Nimrod-G broker). That means the users of G-monitor delegate the responsibility of selection of appropriate resources for execution of their application jobs to the broker. We have used G-monitor successfully in a number of Grid demonstrations including the SC 2002 HPC challenge as part of the Global Grid collaboration.

Some possible future developments for G-Monitor are as follows:

- Provide a graphical plot of how their experiment is progressing. eg: A plot that is updated every time delta showing its budget, number of jobs completed etc., giving the user a better idea as to the current status of the experiment.



G–monitor

Broker hostname: horowitz
portnumber: 53458

Set QoS Parameters
Job Information
Resources
Experiment Status
Logout

API calls

getjobs
getreadyjobs
jobstatus
getexecjobs
getdonejobs
getfailjobs
getjobinfo

author: Martin Placek
mplac@students.cs.mu.oz.au

Gridbus: Server Information

Server Name	Host Name	Service Price(G$)	Status	Remarks
s1	eltoro.pcz.czest.pl	1	✓	"2 free nodes"
s2	eltoro.pcz.czest.pl/jobmanager–pbs	1	✓	"2 free slots"
s3	bezier.man.ac.uk	1	✓	"on probation: 27 free nodes"
s4	onyx3.zib.de	1	✓	"on probation: 18 free nodes"
s5	hpc76.sit.nrc.ca	1	✓	"on probation: 4 free nodes"
s6	gnet01.hpc.unimelb.edu.au	1	✓	"on probation: 2 free nodes"
s7	kiwi.zib.de	1	✗	"CommandFailed: grid–info–search: exited with code 1: ldap_bind: Can't contact LDAP server"
s8	herschel.amtp.cam.ac.uk	1	✓	"on probation: 2 free nodes"
s9	herschel.amtp.cam.ac.uk/jobmanager–pbs	1	✓	"2 free slots"
s10	horowitz.cs.mu.oz.au	1	busy	"no free nodes 0 1 0"
s11	galley.doshisha.ac.jp	2	busy	"no free nodes 0 1 0"
s12	cabibbo.physics.usyd.edu.au	2	✓	"on probation: 1 free node"
s13	lem.ph.unimelb.edu.au	3	✓	"on probation: 2 free nodes"
s14	lem.ph.unimelb.edu.au/jobmanager–pbs	3	✓	"9 free slots"
s15	date1.ics.es.osaka–u.ac.jp	1	busy	"on probation: no free nodes 0 1 0"
				"CommandFailed: grid–info–search: exited

Figure 7: Resource Status.

- Provide a configuration page whereby the user is able to personalise G-Monitor for their environment. Example: Setup default Nimrod-G server, number of jobs per page, etc.

Availability

The G-Monitor software with source code can be downloaded from the Gridbus project website:

`http://www.gridbus.org/`

Acknowledgement

We would like to thank participants in the SC2002 Global Grid Testbed Collaboration for providing us access to the resources that have made large scale experiments possible. We thank Anthony Sulistio and Srikumar Venugopal for their comments on the paper.

References

[1] Apache Web Server, http://www.apache.org/

[2] D. Abramson et. al., *Using Application Programming Interface*, Chapter 9, EnFuzion Manual, 2002. Available at: http://www.csse.monash.edu.au/cluster/enFuzion/api.htm

[3] I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999.

[4] I. Foster and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit*, International Journal of Supercomputer Applications, Volume 11, Issue 2, 1997.

[5] A. Grimshaw and W. Wulf, *The Legion Vision of a Worldwide Virtual Computer*, Communications of the ACM, vol. 40(1), January 1997.

[6] NeuroGrid Project, http://www.gridbus.org/neurogrid/

[7] R. Buyya, D. Abramson, and J. Giddy, *An Economy Driven Resource Management Architecture for Global Computational Power Grids*, Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA., June 2000.

[8] The Gridbus Project, http://www.gridbus.org

[9] M. Thomas, S. Mock, J. Boisseau, M. Dahan, K. Mueller, D. Sutton, *The GridPort Toolkit Architecture for Building Grid Portals*, Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, Aug 2001.

[10] NPACI HotPage -- https://hotpage.npaci.edu/

[11] Sun, *Sun Grid Engine Portal*, http://www.sun.com/solutions/hpc/pdfs/TCP-final.pdf

[12] G. Ma and P. Lu, *PBSWeb: A Web-based Interface to the Portable Batch System*, 12th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS), Las Vegas, Nevada, U.S.A., November 6-9, 2000. http://www.cs.ualberta.ca/~pinchak/PBSWeb/

[13] G. Aloisio, M. Cafaro, P. Falabella, C. Kesselman, R. Williams, *Grid Computing on t he Web using the Globus Toolkit,* Proceedings of the 8th International Conference on High Performance Computing and Networking Europe (HPCN Europe 2000), Amsterdam, Netherlands, May 2000.

[14] J. Novotny, *The Grid Portal Development Kit,* Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[15] Global Grid Testbed Collaboration - http://scb.ics.muni.cz/static/SC2002/

[16] SC 2002, International Conference on High Performance Networking and Computing, November 16-22, 2002. http://www.sc-conference.org/sc2002/

[17] R. Buyya, S. Date, Y. Mizuno-Matsumoto, S. Venugopal, and D. Abramson, *Composition of Distributed Brain Activity Analysis and its On Demand Deployment on Global Grids*, Technical Report, Dept. of Computer Science and Software Engineering, The University of Melbourne, Australia, February 2003.

[18] A. Natrajan, A. Nguyen-Tuong, M. Humphrey, M. Herrick, B. Clarke, A. Grimshaw, *The Legion Grid Portal***,** Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[19] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, Ph.D Thesis, School of Computer Science and Software Engineering, Monash University, Australia, April 2002.