The Journal of Supercomputing, 36, 17–31, 2006
© 2006 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

A Market-Oriented Grid Directory Service for Publication and Discovery of Grid Service Providers and their Services

JIA YU jiayu@cs.mu.oz.au
SRIKUMAR VENUGOPAL srikumar@cs.mu.oz.au
RAJKUMAR BUYYA raj@cs.mu.oz.au
Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia

Abstract. The emergence of Grids as a platform for sharing and aggregation of distributed resources increases the need for mechanisms that allow an efficient management of resources. The Grid economy has been identified as one of the potential solutions as it helps in managing the supply and demand for resources and enables sustained sharing of resources by providing economic incentive for Grid resource providers. An economy based Grid computing environment needs to support an infrastructure that enables the creation of a marketplace for meeting of providers and consumers. This paper presents the *Grid Market Directory* (GMD) that serves as a registry for publication and discovery of Grid service providers and their services.

Keywords: grid computing, grid market, service publication, service discovery, web services

1. Introduction

Grids [9] are emerging as a global cyber-infrastructure [1] for solving large-scale problems in science, engineering and business. They enable the sharing, exchange, discovery, selection and aggregation of geographically distributed, heterogeneous resources—such as computers, data sources, visualization devices and scientific instruments. As Grids comprise of a wide variety of resources owned by different organizations with different goals, they present a number of challenges in managing shared distributed resources. To address these challenges, a distributed computational economy (aka Grid economy) [4] mechanism has been proposed. This mechanism would support efficient management of distributed resources by regulating the supply and demand through differentiated pricing strategies and promote sustained sharing of resources by providing incentive for Grid Service Providers (GSPs). It also encourages Grid Service Consumers (GSCs) to specify their Quality-of-Service (QoS) needs based on their actual requirements. Therefore, the use of Grid economy mechanism in managing shared Grid resources would lead to the creation of a Grid Market-Place (GMP).

It has been envisioned that Grids enable the creation of Virtual Organizations (VOs) [12] and Virtual Enterprises (VEs) [7] or computing marketplaces [6]. In a typical market-based model VO/VE, GSPs publish their offerings in a market directory (or a catalog) and GSCs employ a Grid Resource Broker (GRB) that identifies GSPs through the market

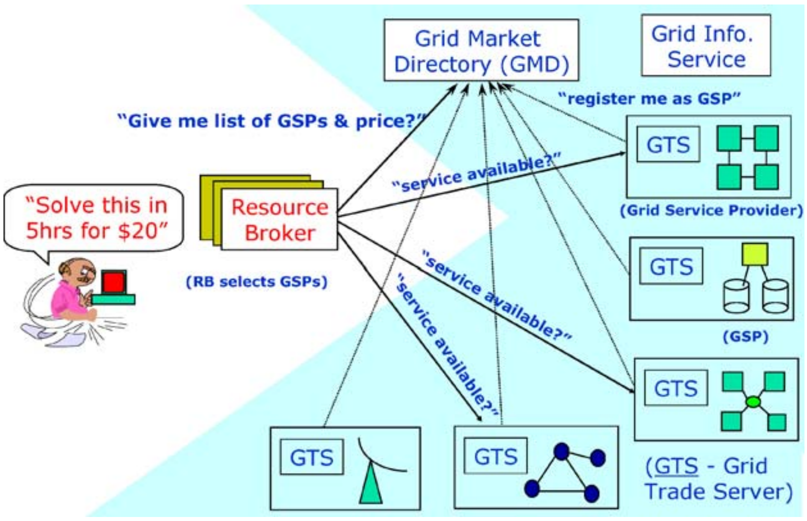


Figure 1. A typical community market-based Grid computing environment [2].

directory and utilize the services of suitable resources that meet their QoS requirements(see Figure 1).

To realize this vision, Grids need to support diverse infrastructures/services [12] including an infrastructure that allows (a) the creation of one or more GMP registries; (b) the contributors to register themselves as GSPs along with their resources/application services that they wish to provide; (c) GSPs to publish themselves in one or more GMPs along with service prices; and (d) Grid resource brokers to discover resources/services and their attributes (e.g., access price and usage constraints) that meet user QoS requirements. In this paper, we propose and develop an infrastructure called the *Grid Market Directory* (GMD) that supports these requirements. The GMD has been implemented using emerging web services technologies as part of the Gridbus Project [17], which is developing technologies for service-oriented utility computing that scale from centralized systems to Grids.

The rest of this paper is organized as follows: The related work within Grid and Web services communities is presented in Section 2. The detailed system architecture and features of the GMD are described in Section 3. Section 4 describes technologies that are used in the current implementation. A use-case study is presented in Section 5. Experimental results are presented in Section 6. We conclude in Section 7 with a discussion of current system status and future work.

2. Related work

Some related efforts for developing service publication registries include: Globus MDS [13], UDDI [30] and ICENI [14]. The working model and functionality offered by these directory services in comparison to the GMD are discussed below.

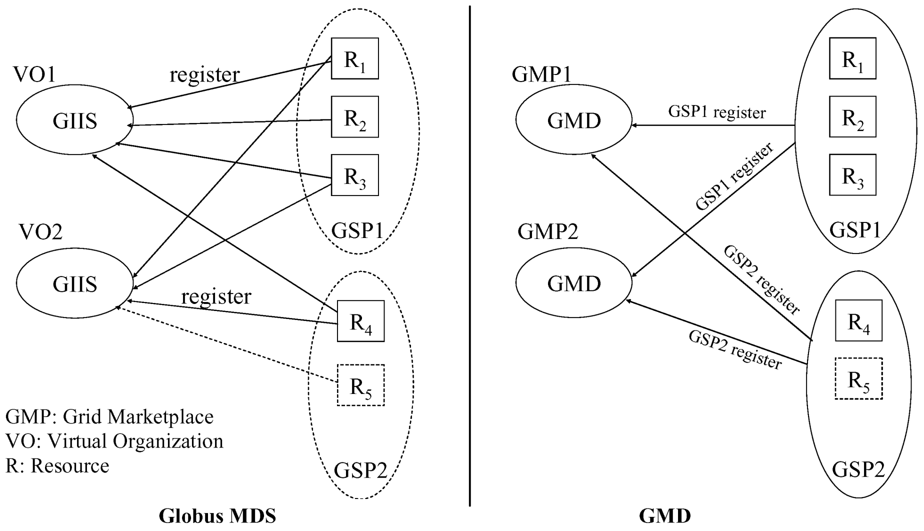


Figure 2. Resources/GSPs registration.

The Globus [16] Metacomputing Directory Service (MDS) [13] supports resources discovery mechanisms in a VO. MDS-2 [8] released as part of the Globus 2 (GT2) toolkit consists of a configurable information provider component called the Grid Resource Information Service (GRIS), and a configurable aggregate directory component called the Grid Index Information Service (GIIS). GRIS holds resource properties, such as characteristics of a compute resource and network, while GIIS accepts GRIS registry and provides aggregation services. An illustration of registering resources in Globus MDS GIIS and Gridbus GMD is shown in Figure 2. In Globus MDS, all resources that are part of the VO have to be registered explicitly with the corresponding GIIS of the VO; whereas in the Gridbus GMD based VO/VE, only the GSPs need to be registered with the corresponding GMP. This also means that, if a GSP acquires a new resource (e.g., R_5 in GSP_2 shown in Figure 2) and wants to offer its services on the Grid, in the case of Globus, the GSP needs to explicitly register this new resource services with GIIS. However, this is not required for the Gridbus GMD. After identifying the GSPs and their resources, the resource brokers can query the Globus GRIS to identify their low-level properties (e.g., resource architecture and configuration details). Therefore, GMD can be considered a complementary technology as it enables market-oriented Grid computing as it allows leveraging of the existing Grid technologies (e.g., MDS GRIS).

The Universal Description, Discovery and Integration (UDDI) specifications [30] define a standard for enabling businesses to publish or discover Web services. It does not provide the ability to query high-level service information such as identifying services within a particular price range. It has been targeted to serve as a registry for XML-based Web services where as GMD has been targeted to serve as a registry for publication of GSPs and their application-level services (e.g., CPU service/molecular docking service).

ICENI (Imperial College e-Science Networked Infrastructure) [14], developed by the London e-Science Centre, supports the concept of a computational community based on Jini technology [24]. The participants in the computational community publish their services through Jini lookup service, which serves as a registry in a Jini environment. Services can be identified through lookup services by matching data members of their entry object with requests. Compared to Web services based registry, Jini facilitates dynamic service discovery but restricts end-points in a pure Java environment.

3. GMD architecture and design

The architecture of the GMD is shown in Figure 3. The key components of the GMD are:

- GMD Portal Manager (GPM) that facilitates service publication, management and browsing. It allows service providers and consumers to use a web browser as a simple graphical client to access the GMD.
- GMD-Query Web Service (GQWS) that enables applications (e.g. resource broker) to query the GMD to find a suitable service that meets the job execution requirements (e.g. budget).

Both the components receive client requests through a HTTP server. Additionally, a database (GMD repository) is configured for recording the information of Grid services and service providers.

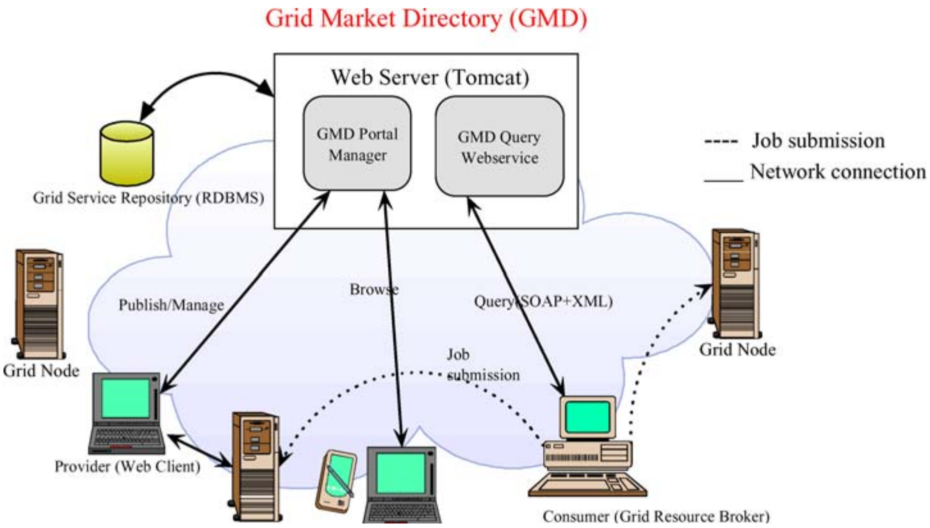


Figure 3. GMD architecture and design.

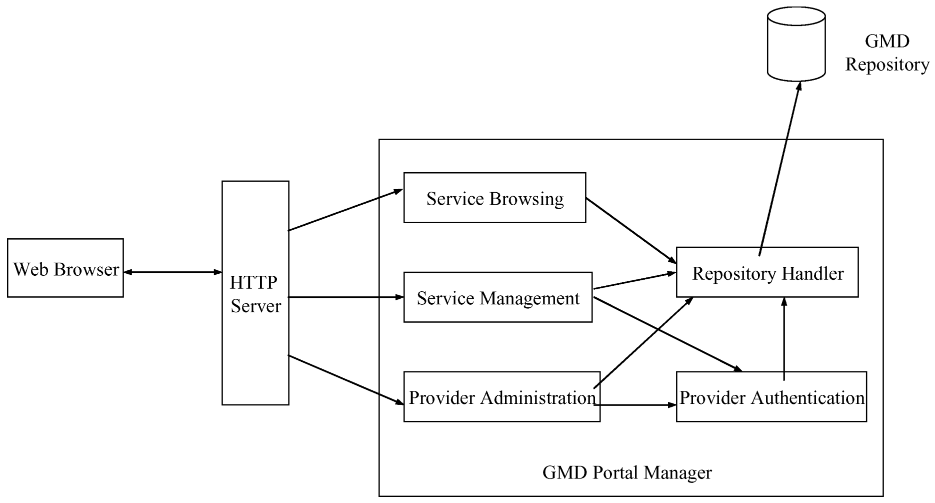


Figure 4. GMD portal manager architecture.

3.1. GMD portal manager

The architecture of the GMD Portal Manager (GPM) is shown in Figure 4. The GPM provides three different access interfaces: service browsing, provider administration and service management.

Service browsing. The GPM allows users to browse all registered services or only services offered by a specific provider. Additionally, services in the GMD are categorized by service type, such as Earthquake Engineering, Molecular Docking and CPU Service, so that users can browse them for a particular application area. For instance, the high-energy physics community can browse services related to its area along with their access costs.

Provider administration. The provider administration module is responsible for account management including registration and removal. The account information of the provider is acquired at the time of registration. This includes the provider's name, login name, password, contact address and some additional information.

Service management. The service management module enables the registered providers to maintain their services in the GMD. A service management page is dynamically generated for each registered provider, through which it can add, update and remove services. Basic service attributes include:

- service name
- service type
- hardware price (cost per CPU-sec)

- software price (cost per application operation)
- node host name, and
- location of application deployment (path).

In addition, security issues are also addressed in the GPM. A login authentication mechanism for identifying registered providers is employed in the service management and provider administration. In the service management interface, service modification operations are also authenticated before being committed to the repository.

3.2. GMD query web-services

The GMD provides web services [32] that applications can use to query the registry. The GMD Query Web-Service (GQWS) is built using SOAP (Simple Object Access Protocol) [27]. The main benefit of using SOAP is that it is based on standard XML [31] so that Web services can even be invoked from different applications irrespective of the language used in their implementation.

The six basic SOAP invocation methods supported by the GQWS are listed below:

1. QueryService()—returns a list of all services.
2. QueryServiceByType(serviceType)—returns a list of providers offering a specific service type.
3. QueryServiceByHost(hostName)—returns the service information associated with a host name.
4. QueryServiceByProvider(providerName)—returns a list of services supported by a provider.
5. QueryServiceContact(serviceType)—returns a list of contact addresses of services of specified type.
6. QueryPrice(serviceName)—returns service price for a specified service.

The GQWS consists of two modules: *Query Processor* and *Repository Handler*. The interaction between the GQWS and its client is illustrated in Figure 5. The GQWS communicates with its client by messages in XML format. The query message is encapsulated within a SOAP message, which is transferred by HTTP between the web server and the GMD client. The SOAP engine acquires the query message and forwards it to the GQWS. The *Query Processor* handles query message parsing and takes appropriate actions based on the content of the message, while the *Repository Handler* is responsible for retrieving data from the database. The response message is finally constructed by the *Query Processor* and sent back to the client.

The format of the query message for indicating a certain service type (e.g. CPU service) is:

```
<query_service>
     <service_type>CPU service</service_type>
</query_service>
```

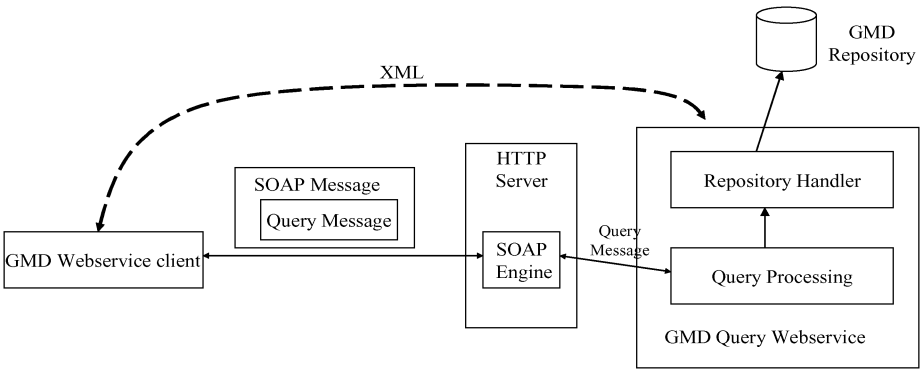


Figure 5. Interaction between the GQWS and its client.

This XML structured query message can be extended easily. The example below illustrates a query message with two constraints: service type and service provider.

```
<query_service>
      <service_type>...</service_type>
      <provider_name>...</provider_name>
</query_service>
```

If the query for services of a certain type is successful, the *Query Processor* responds with a message whose format is given below:

```
<?xml version = ‘‘1.0" encoding=‘‘UTF-8"?>
<service-details type=’’...’’ status=’’ok’’>
       <service>
              <name>...</name>
              <provider>...</provider>
              <price>
                     <hardware>...</hardware>
                     <software>...</software>
              </price>
              <address>...</address>
              <description>...</description>
       </service>
       <service>
              .
              .
       </service>
       <service>
              .
              .
```

```
        </service>
</service-details>
```

The attribute *type* indicates the service type of the services listed in the response message. The attribute *status* indicates the processing status of the query. If the query fails, the format of the response message appears as follows:

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<service-details type=''...'' status=''error''>
       <reason>...</reason>
</service-details>
```

The detailed specification of XML elements used in the GMD can be found in [17].

GQWS client API architecture. In addition to the standard SOAP client infrastructure, the GMD clients need XML message building and parsing capability depending on the query and response specification. The GMD provides Java client APIs that hide SOAP/XML specifications. Thus, the developers will be able to utilize the GQWS into their applications easily without knowing low-level details of SOAP and XML configurations. Figure 6 presents the architecture of the GMD client APIs and their interaction with the GQWS.

4. Implementation

The GMD has been implemented and tested on both Windows 2000 and Solaris platforms. The Internet/Web technologies used in GMD's implementation include: Jakarta Tomcat [28], Apache SOAP [19], JDOM [23] and MySQL [25]. Jakarta Tomcat is used to

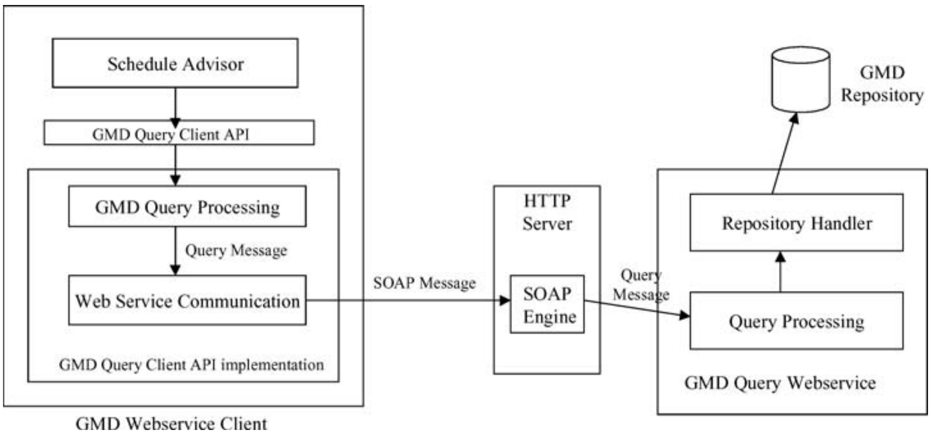


Figure 6. GMD query client API architecture.

provide the HTTP service and Apache SOAP is used as the SOAP infrastructure. JDOM is used to build and parse XML messages. MySQL is used as a relational database to maintain service information and provider profile.

The presentation view of the GPM is implemented by using Java Server Pages (JSP) [20]. The main benefit of using JSP is that it allows the instantiation of Java objects within HTML. Thus, the dynamic display of the data retrieved from the GMD database using JDBC [22] can be incorporated in HTML. Two pages generated by the GPM are shown in Figure 7.

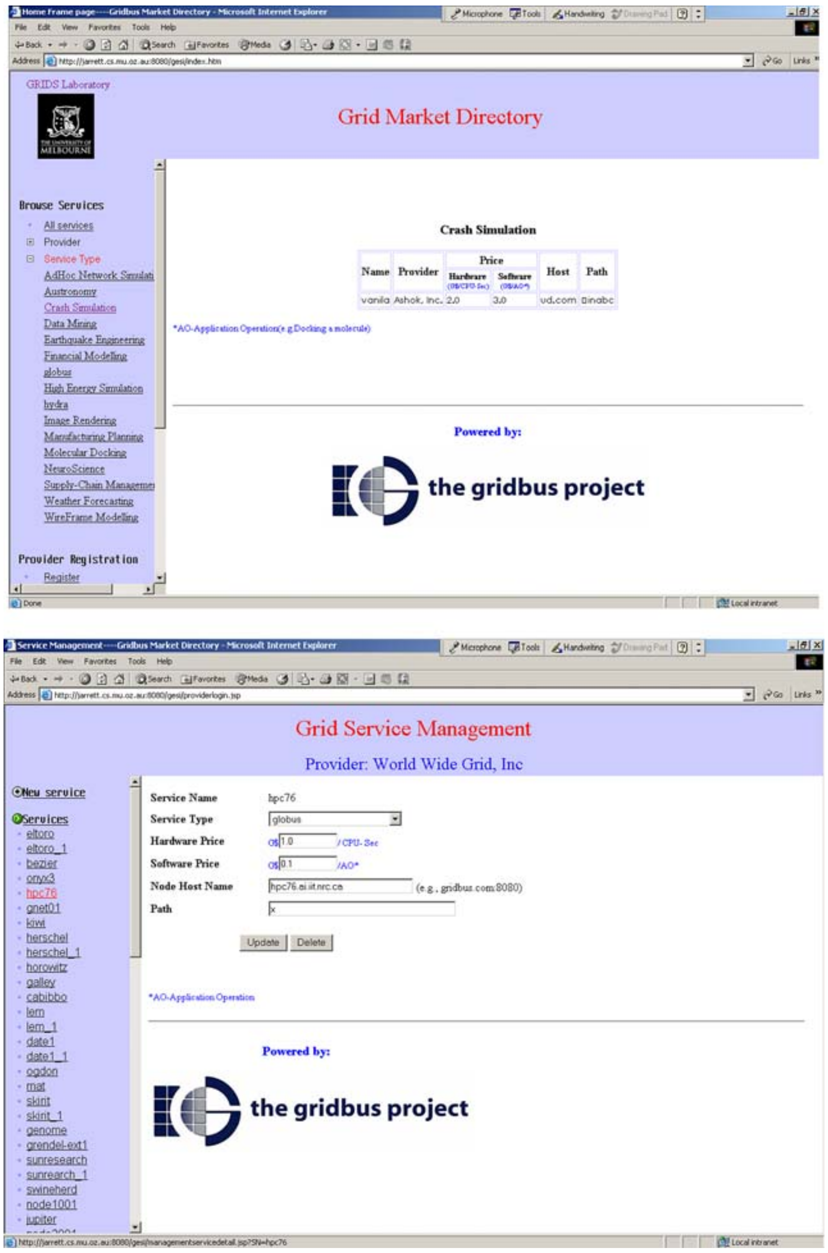
The GMD allows only authorised users to modify service information, as the portal access is protected through password-based authentication mechanism. The Java servlet HttpSession APIs [21] are used for the GMD security implementation. As shown in Figure 8, after the user logs in, a name attribute is set to the HttpSession object of corresponding request session. When the user logs out, the name attribute is deleted from the object. This ensures that only active and authenticated users are able to update their records.

5. Use case study

This section presents the utilization of GMD in the Global Grid Testbed Collaboration [15] that was set up to demonstrate capabilities of the state-of-the-art Grid technologies and applications at the SC 2002 HPC Challenge [26] held in Baltimore, USA. The testbed consisted of 69 machines located in 14 countries across 5 continents. We had access to the resources in the testbed as we were demonstrating one of the four Grid applications as part of this collaboration. As this collaboration fits the notion of VO, the GMD infrastructure, hosted on a (Sun Solaris) server located at the University of Melbourne, Australia (see Figure 9), was used to create a registry for this VO. Each participant was registered as a GSP and every contributed resource with its host address, application path and the access cost of its hardware and software was published. Since they all provide Globus job management service [16], we assigned service type *globus* to all resources. In this VO, the GMD services have been used for the following purposes:

- to register the testbed participants as Grid service providers,
- to publish resources/services contributed by each participant and their attributes, and
- to enable GMD applications/tools to discover resources/application services at runtime.

During the HPC challenge demonstration, the Nimrod-G broker [3] and Gridbus scheduler [5] have been utilized to deploy brain activity analysis application on the testbed resources. The gridbus scheduler was able to query the GMD at runtime to identify GSPs and their resources along with their access cost. The scheduler dynamically selected resources depending on their cost, capability and user QoS(deadline and budget) constraint. The results of the scheduling experiments that used the GMD services and the testbed resources are reported in [5].



Grid Market Directory

Crash Simulation

Name	Provider	Price		Host	Path
		Hardware (G$/CPU-Sec)	Software (G$/AO*)		
vanila	Ashok, Inc.	2.0	3.0	vd.com	Dinabc

*AO-Application Operation(e.g.Docking a molecule)

Grid Service Management

Provider: World Wide Grid, Inc

Service Name	hpc76
Service Type	globus
Hardware Price	G$ 1.0 / CPU- Sec
Software Price	G$ 0.1 /AO*
Node Host Name	hpc76.ei.iit.nrc.ca (e.g., gridbus.com:8080)
Path	x

*AO-Application Operation

Figure 7. GPM Demo pages: (a) left side—service browsing page by type *Crash Simulation* (b) right side—service modification page for a hypothetical GSP called the *World Wide Grid, Inc.*

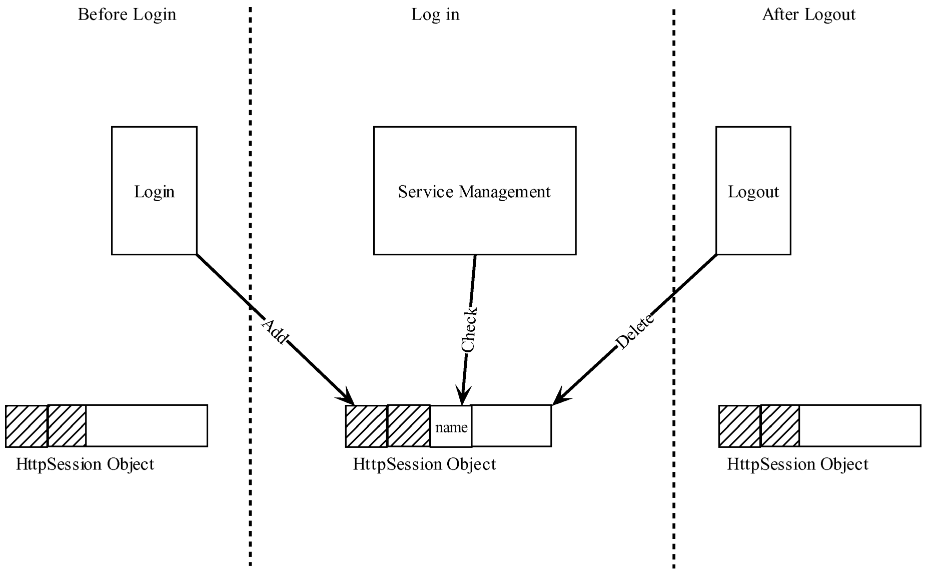


Figure 8. Security mechanism of GMD portal manager.

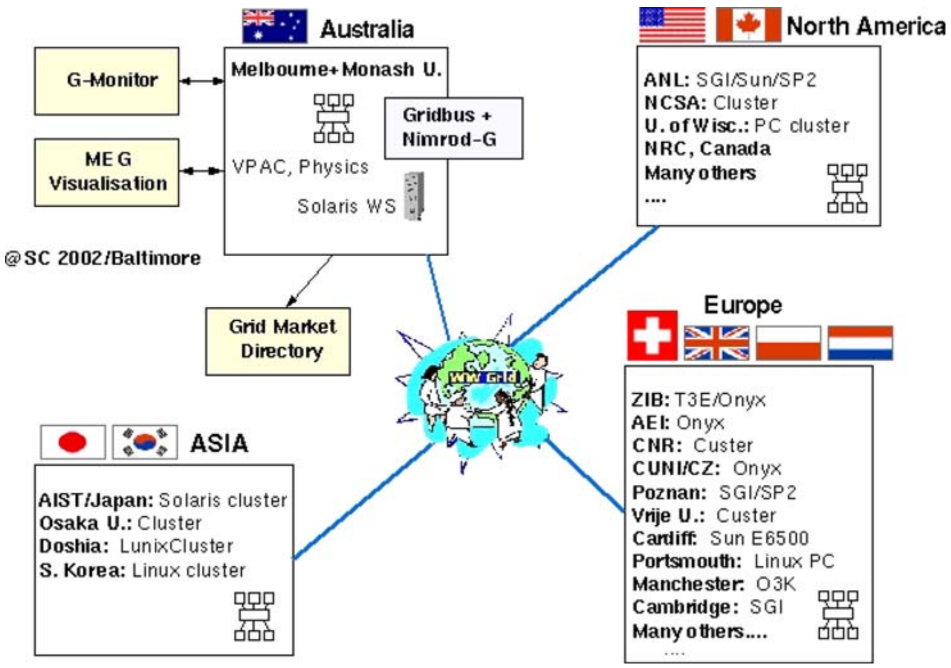


Figure 9. GMD usage in the global grid testbed collaboration demo @ the SC 2002 HPC challenge.

6. Experiments and discussion

In order to evaluate the performance of our research prototype, we measure the query response time. The major factors that affect the response time are network latency, machine speed and the implementation of http server, SOAP engine and database. We tested the system on a Pentium 4 (1.8 GHz) PC and ran client programs on a Crusoe 600 notebook. We used Apache tomcat 3.2.4 as http server, Apache SOAP 2.2 as SOAP engine and Mysql 3.23 as database. Three experiments have been done in our local network. Average route-trip time taken by pinging Pentium PC from the notebook is around 0.4 ms.

In the first experiment, we measured query response time for the service list of a provider using GMDs with different registered providers, but every provider provides same service. As shown in Table 1, the response time is independent of the number of providers registered.

In the second experiment, we measure the query response time for the service list of a provider using GMDs with the same number of registered providers, but the number of services provided by one provider is different. The result shows in Table 2 indicates the response time increases linearly by the number of provider's services.

In the third experiment, we measured the query response time for finding a price of a service. Given that different data location in the database may cause different processing time, we managed to query all services in each tested GMD and calculated the average response time. In the GMD of 5000 services the average response time is 0.078 second. As indicated in Table 3, we tested the system using 500, 1000, 2000 and 5000 services. The result shows that the response time does not change significantly with the number of providers and their services.

Table 1. Query time for the provider service list on GMDs with different number of providers

No. of providers	No. of services	Query response time (ms)
10	1	122
50	1	121
100	1	122
500	1	123

Table 2. Query time for the provider service list on GMDs with different number of services for each provider

No. of providers	No. of services	Query response time (ms)
100	1	122
100	5	147
100	10	179
100	20	219

Table 3. Average response time for querying service price on different size GMDs

Provider	Service	Overall services	Average response time (ms)
50	10	500	76
50	20	1000	77
100	20	2000	79
500	10	5000	78

As results show above, the response time does not increase linearly or exponentially with the number of providers registered, so the system can be considered to have the capability of being a large-scale registry for providers and their services. However, the results also indicates that it is not suitable for users to get a list of a large number services provided by a provider or a large number services for a service type, because transferring more information will cause longer delay.

7. Conclusion and future work

The Grid economy-based management of distributed resources helps in the regulation of the supply and demand for resources and offers an economic incentive for sustaining resource sharing and user-QoS requirements driven aggregation. Such market-oriented Grids need to support an infrastructure that enables the creation of a marketplace for meeting of providers and consumers. To meet this requirement, we have proposed and developed a VO marketplace registry called the Grid Market Directory (GMD), which serves as a registry for publication and discovery of Grid service providers and their services.

The GMD consists of two key components: the portal manager and the query Web-service. The GMD portal manager is responsible for provider registration, service publication and management, and service browsing. All these tasks are accomplished by using a standard web browser. The GMD query Web-service provides services so that clients such as resource brokers can query the GMD and obtain the information of resources to discover those that satisfy the user QoS requirements.

The GMD implementation makes use of commodity Java packages including Java Server Page, Apache SOAP, Java Servlet API, JDBC API and JDOM. An open-source servlet container, Tomcat, is used to provide HTTP services and MySQL is used as the GMD repository.

The current version of the GMD supports commodity market model and its services are being used by clients such as economy-based grid schedulers. Future work will enhance the GMD by supporting other economy models such as tender/contract and auction models. Furthermore, efforts are currently underway to enhance GMD to support OGSI [29]-compliant interfaces such as registry port-type and findServiceData. This enables the OGSI-compliant entities to publish their services in the GMD or discover them dynamically. The GMD will be complemented to service data query mechanism addressed in OGSA [10] and provide more advanced query features, for instance, locate services in a certain price range.

To date, security solution used in our prototype is only password protection for the portal access, as we only allow providers to register and manage service information via Web. In the future, we will allow service registration and management by applications. At that stage, GSI mechanisms [11] could be employed for provider authentication.

Availability

The GMD software with the source code can be downloaded from the Gridbus project website: `http://www.gridbus.org/gmd/`

Acknowledgments

We would like to thank Chee Shin Yeo, Steve Melnikoff, Anthony Sulistio and Rob Gray for their comments on the paper.

References

1. D. Atkins et al. Revolutionizing science and engineering through cyberinfrastructure. Technical Report, National Science Foundation, USA, February 2003.
2. R. Buyya. Economic-based distributed resource management and scheduling for grid computing. PhD Thesis, Monash University, Melbourne, Australia, April 12, 2002. Online at `http://www.buyya.com/thesis/`
3. R. Buyya, D. Abramson, J. Giddy. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. In *Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region* (HPC Asia 2000), Beijing, China. IEEE Computer Society Press, USA, 2000.
4. R. Buyya, D. Abramson, and J. Giddy. A case for economy grid architecture for service-oriented grid computing. In 10th *IEEE International Heterogeneous Computing Workshop* (*in conjunction with International Parallel and Distributed Processing Symposium*). San Francisco, California, USA, April 2001.
5. R. Buyya, S. Date, Y. Mizuno-Matsumoto, S. Venugopal, and D. Abramson. *Economic and on demand brain activity analysis on global grids*. Technical Report, University of Melbourne, Australia, Feb 2003.
6. R. Buyya and S. Vazhkudai. Compute power market: Towards a market-oriented grid. In *First IEEE/ACM International Symposium on Cluster Computing and the Grid* (CCGrid 2001), Brisbane, Australia, May 15–18, 2001.
7. L. Camarinha-Matos and H. Afsarmanesh (eds.) *Infrastructure for Virtual Enterprises: Networking Industrial Enterprises*. Kluwer Academic Press, 1999.
8. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Grid information services for distributed resource sharing. In 10th *IEEE International Symposium on High-Performance Distributed Computing* (HPDC 2001), San Francisco, CA, USA, 2001.
9. I. Foster and C. Kesselman (eds). *The Grid: Blueprint for a Future Computing Infrastructure.* Morgan Kaufmann Publishers, USA, 1999.
10. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. *The physiology of the grid: An open grid services architecture for distributed systems integration.* Technical report, Argonne National Laboratory, 2002.
11. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. *A Security Architecture for Computational Grids.* ACM Conference on Computers and Security, ACM Press, pp. 83–91, 1998.
12. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications* 15(3), 2001.

13. S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In 6th *IEEE Symposium on High Performance Distributed Computing* (HPDC'97), Portland, OR, USA, 1997.
14. N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. ICENI: An open grid service architecture implemented with Jini. *SuperComputing 2002* (SC2002), Baltimore, USA, Nov 2002.
15. Global Grid Testbed Collaboration, `http://scb.ics.muni.cz/static/SC2002`
16. Globus Toolkit, `www.globus.org`
17. Gridbus project website, `http://www.gridbus.org`
18. Marty Hall. *Core Servlets and JavaServer Pages*. Prentice Hall PTR, 2001
19. Java Apache SOAP, `http://xml.apache.org/soap/`
20. Java Server Page Technology, `http://java.sun.com/products/jsp`
21. Java Servlet Technology, `http://java.sun.com/products/servlet`
22. JDBC Technology, `http://java.sun.com/products/jdbc/`
23. JDOM API, `http://www.jdom.org/`
24. Jini Technology, `http://wwws.sun.com/software/jini/`
25. MySQL open-source Database, `http://www.mysql.com`
26. SC 2002, *International Conference on High Performance Networking and Computing*, November 16–22, 2002. `http://www.sc-conference.org/sc2002/`
27. Simple Object Access Protocol (SOAP) 1.1, `http://www.w3.org/TR/SOAP/`
28. Tomcat open-source servlet container, `http://jakarta.apache.org/tomcat`
29. S. Tuecke, K. Czajkowski, I. Foster, S. Graham, C. Kesselman, D. Snelling, and P. Vanderbilt. *Open grid services infrastructure (OGSI)*, Draft 23, February 17, 2003, `http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf`
30. UDDI Project, `http://www.uddi.org/`
31. W3C, Extensible Markup Language (XML), `http://www.w3c.org/XML`
32. W3C. Web services, `http://www.w3.org/2002/ws/`