the gridbus project

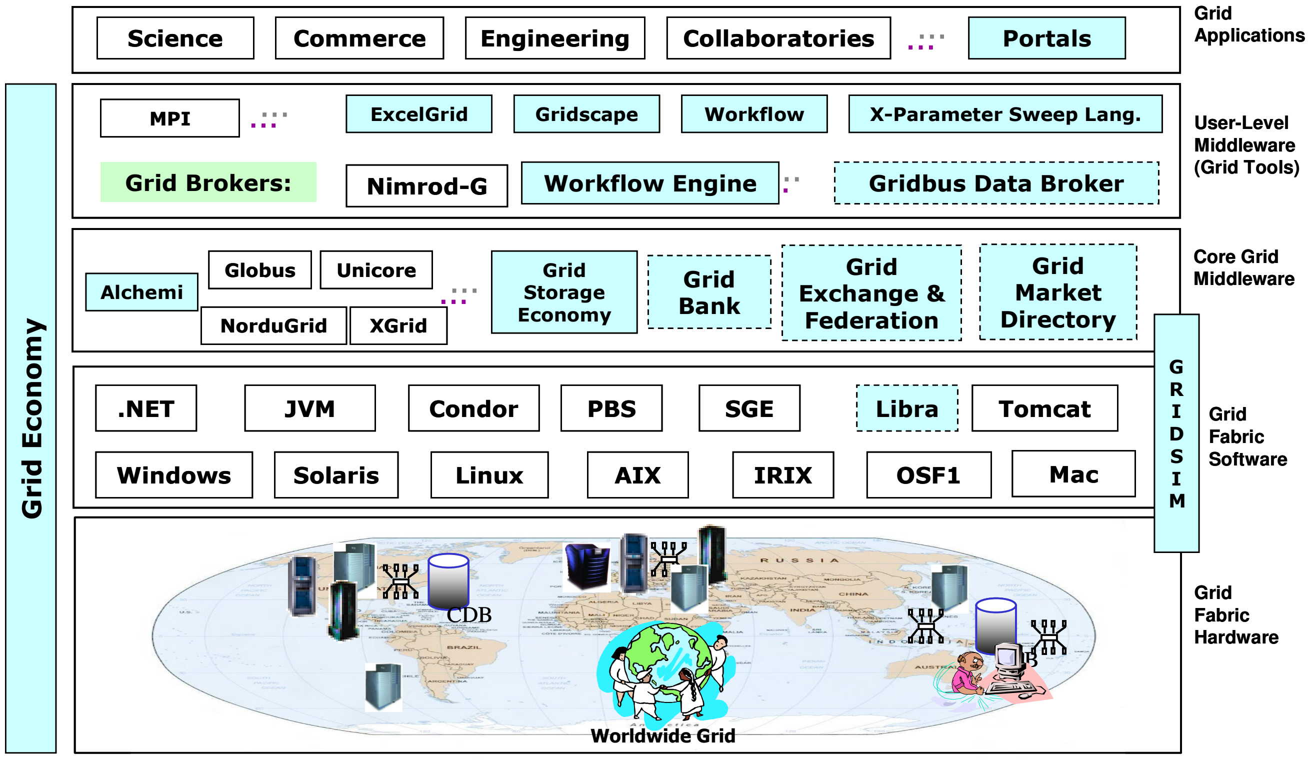
http://www.gridbus.org

THE UNIVERSITY OF MELBOURNE

NOV.2004

The Gridbus Middleware Manual

GRID COMPUTING AND DISTRIBUTED SYSTEMS LABORATORY
DEPT. OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
THE UNIVERSITY OF MELBOURNE, AUSTRALIA



Supported By:

STORAGETEK
Sun microsystems
SCS LIVING TECHNOLOGY
IBM
apac
VPAC VICTORIAN PARTNERSHIP FOR ADVANCED COMPUTING
Australian Government
Australian Research Council

Table of Contents

1 Introduction

The Gridbus Project is engaged in the design and development of grid middleware technologies to support eScience and eBusiness applications. These include visual Grid application development tools for rapid creation of distributed applications, competitive economy-based Grid scheduler, cooperative economy-based cluster scheduler, Web-services based Grid market directory (GMD), Grid accounting services, Gridscape for creation of dynamic and interactive testbed portals, G-monitor portal for web-based management of Grid applications execution, and the widely used GridSim toolkit for performance evaluation. Recently, the Gridbus Project has developed a Windows/.NET-based desktop clustering software and Grid job web services to support the integration of both Windows and Unix-class resources for Grid computing. A layered architecture for realisation of low-level and high-level Grid technologies is shown in the figure below. Some of the Gridbus technologies discussed below have been developed by making use of Web Services technologies and services provided by low-level Grid middleware, particularly Globus Toolkit and Alchemi.

2 What is included in the middleware?

The following table lists out all the components bundled with the Gridbus Middleware:
.

Component	Description	Current Status
Gridbus Broker	An economy-capable Data Grid service broker for scheduling distributed data oriented applications across Windows and Unix-variant Grid resources.	v.1.2 with XML input interface.
Alchemi	A .NET-based desktop Grid framework.	v.0.8 with interface for user-level scheduling.
GridSim	A toolkit for modeling and simulation of global Grids.	v.3.1 with network extensions and trace capabilities.
Grid Workflow Management System	Workflow Management System and Monitor for Grid systems	v.1.0
Grid Market Directory	A directory for publication of Grid Service provides and their services.	v.1.1 with web services based query interface.
Grid Bank	A grid accounting, authentication, and payment management infrastructure.	v.1.0 with web services interface.
Gridscape	A tool for the creation of interactive and dynamic Grid test bed web portals.	v.1.2 implemented as Web application within Tomcat.
G-Monitor	A web portal to manage execution of applications on Grids using remote brokers.	v.2.0

ExcelGrid	.Net based GUI client application for remote submission and monitoring of jobs to Grid middleware and brokering systems such as Gridbus broker and Alchemi.	v.0.8
Visual Parametric Modeller	A graphical environment for application parameterisation.	v.1.0 with visual parameterisation of data files.
Additional Tools:		
Libra	An economy based scheduler for clusters.	v.1.0 implemented as PBS plug-in.

3 Gridbus Broker

3.1 Introduction

The Gridbus Grid Broker is a software component which matches users' job requirements to the available resources and schedules the job to fulfill those requirements. The Broker therefore, takes care of resource discovery, job scheduling, execution, monitoring and gathering of output. It provides an abstraction to users particularly those who are not familiar with the complexity of Grid environments.

3.2 Prerequisites and Installation

Requirements on the Broker side:

- Java Virtual Machine 1.4.x
- Valid certificate (if using Globus machines)
- Additionally, all ports above 32000 must be open so that the jobs can connect back to the broker.

Requirements on the Grid nodes side:
Must run one of:

- Globus 2.4.x
- Alchemi Manager 0.8.0
- UNICORE Gateway 4.1.0 (Experimental support within Broker)

In case of Globus, the user must have valid credentials to submit a job to Globus, i.e., the user's certificate-id must be mapped to an account on the node.

Installation (these instructions assume a Linux platform, a Windows installation follows similar steps):

- Unzip the archive

```
$ tar -zxvf gridbusbroker.1.2.tgz
```

- In the `gridbus/bin` directory, change `classpath.rc` and `gridbus-broker` script to reflect your setup. (Change the `DB_HOME` variable to point to your broker directory)
- Set your `JAVA_HOME` variable to your Java installation. Additionally, it is recommended that you put the `.gridbus-broker`. command in your `PATH` by doing (for Bash shell)

```
$ export PATH=$PATH:<path-to-broker-directory>/bin
```

3.3 Using the Broker

There are 2 inputs to the broker:

- the job description which is encoded within a *plan file*
- the list of hosts to run the jobs on which is given within a *gatekeeper fi*le.

The Plan file:

The input to the broker follows the parameter-sweep execution model in which the same application is run for different values of input parameters often expressed as ranges. This model is similar to that followed by systems such as Nimrod/G,Enfuzion and AppLeS PST. Parametric executions are suitable for applications such as data analysis in which a program has to be run multiple times over different datasets. The Gridbus broker has extended this model to data grids, that is, one can specify remote files as input parameters. The Gridbus broker has adopted the Nimrod/G plan file format as the basis for its input description. Below is the plan file from the example in the Installation section

```
parameter st integer range from 1 to 10 step 1;
task main
node:execute echo "HELLO WORLD! $st" > $jobname.out
copy node:$jobname.out $jobname.out
endtask
```

As seen above, a plan file is a text file consisting of *parameters* which specify value ranges and *tasks* consisting of commands. There are as many jobs as there are unique sets formed by cross-product of parameter values, i.e., if there are 2 parameters each having 10 values then there will be 10x10=100 jobs. A job therefore is defined by a unique set of parameter values and the task specified within the plan file. The syntax of the Nimrod/G plan file can be found online at *http://www.csse.monash.edu.au/cluster/enFuzion*. However, the Gridbus broker does not support all the declarations that have been defined in the Nimrod/G plan file format. Notably, only the main task is supported at the present. Some other declarations such as random and compute ranges are also not supported. However, the broker has introduced some declaratives of its own such as:

- Gridfile parameter :
The Gridfile parameter allows the user to specify remote files as input parameters. The syntax of this parameter reads as:

```
parameter <name of parameter> Gridfile <file location>,<file
location>,..
```

Following is an example of how this parameter can be specified:

```
parameter INFILE Gridfile fn:/users/winton/fsimddks/fsimdata*.mdst
```

Here, the URI (Uniform Resource Identifier) of a Globus Replica Catalog location is passed to the broker as an input. Note the location holds multiple files. The broker internally resolves this to the actual number of files present. Also, while scheduling the jobs for execution, the broker takes care to see that the jobs are scheduled close to where the files are physically located. This means that the overhead of transferring large data files is significantly reduced. More details about the scheduling can be found in the technical report on the broker available at
http://gridbus.org/papers/gridbusbroker.pdf.
Currently, the broker supports only the Globus Replica Catalog. However, support for the SDSC SRB(Storage Resource Broker) is close to being finalised while support for other URI such as http://, ftp:// and file:// are in the offing.

- gcopy command :
The gcopy command allows the user to copy a single file to or from a defined remote host which is not the node on which the job is to be executed (for that facility, look at the copy command). This command is useful when all the jobs required a single data file which is however hosted on a remote

server or when the job outputs have to be piped to a defined location on a remote storage host. The syntax of this parameter is:

`gcopy <hostname>:<file locationnfilename> filename` or
`gcopy filename <hostname>:<file locationnfilename>`
An example of its usage is:
`gcopy belle.cs.mu.oz.au:input.dat input.dat`

Internally gcopy resolves to a GSIFTP URI so the file has to be available through GSIFTP and the appropriate permissions have to be in place. Support for SRB is being worked upon.

- **mcopy command :**
While the copy command allows only a single file to be copied over, the mcopy command copies multiple files at the same time. The syntax of the mcopy command is similar to the copy command except that it supports wildcards. For example:

```
mcopy input* node:input/
```

The destination (either remote or local) must be a directory.

The XML input:
The broker internally converts the text plan file to an XML file. This XML file is available in the same directory as the plan file when the broker exits execution. (Look out for `example.xml` when the example shown in the Installation section is run) The schema for this XML document can be found in `xml/planfile Schema.xsd` within the broker distribution. The schema is extensible and can be used to introduce new parameter types if necessary. Future versions of the broker may contain more functionality within the XML than can be made available through the plan file. Therefore, we encourage users to work with the XML file for developing applications on top of the broker. It is simple to extend the schema for your own purposes. The XML is parsed using a DOM parser and a reflection mechanism so the resolution of the XML document to jobs is independent of the schema itself. Introducing a new parameter involves writing a resolver and sticking to the naming scheme. Interested users can look through the source (package `org.gridbus.broker.plan`) for details on implementation.

Note: The broker can be provided with an XML input file by passing it through the .plan command line option. For example, you could try executing the sample plan file, referred in the installation section, by giving `--plan=example.xml` as the command line option. To create the XML out of a plan file, you can use the gridbus-plan2xml script in gridbus/bin as:

```
$ gridbus-plan2xml <plan file name> <xml file name>
```

Running the Broker for the first time:
All the instructions assume a Linux or a similar Unix environment. The sequence for Windows is similar. Also they assume that currently you are in the top-level directory of the distribution, i.e, in the gridbus/ directory.

- Change to the examples/simple directory within the distribution
- Modify the gatekeeper file to include hosts that you have access to
- In case of Globus resources, initialize proxy

```
    $ grid-proxy-init
```

- Type the following at the prompt

  ```
  $ gridbus-broker --plan=example.pln --gate=gatekeeper
  ```

  You should see lines of log output scrolling by. The broker finishes off with a "`Broker exiting..`" when its done.
- In the directory, you will see output files named j1.out,j2.out,etc. Each file should contain a .HELLOWORLD. message. If there are 10 output files, the broker has been able to successfully execute all the jobs.

The Gatekeeper file:
The gatekeeper file is passed to the broker by the `--gate= option` on the command line or by setting the `GATEKEEPER_FILE` property within DB.properties. This file contains the list of compute nodes to which the user has access for executing jobs. A sample gatekeeper file is shown below:

```
globus belle.cs.mu.oz.au 2
alchemi horowitz.cs.mu.oz.au http://horowitz/alchemi.
crossplatformmanager/crossplatformmanager.asmx
unicore testgrid.unicorepro.com:4001 3
```

The first column is the middleware name. Presently, 3 middleware systems are supported Globus 2.4.x ("globus"), Alchemi 0.8.0("alchemi") and UNICORE 4.1.0("unicore"). For Alchemi and Globus, the second column is the hostname of the resource. In case of UNICORE it is the hostname of the gateway and the port number. The third column can be used to specify an optional cost of using the resource for Globus and UNICORE and it is the URI of the web-service in case of Alchemi(Note: the remote resource must be an Alchemi manager and must be running the web service).
If the first column were to be omitted altogether, the broker will assume it is a Globus resource. The broker probes every resource to see whether it is alive and able to execute jobs. For Globus resources, it also probes the GRIS (Grid resource Information Service) for details about the resource. If the broker is not able to gather information successfully, it considers the remote resource as unavailable. It is also possible to avoid using the gatekeeper file in situations where the resources are available through an online index such as a Virtual Organisation index, GIIS (Grid Index Information Service) or a GMD (Grid Market Directory). In such cases, a (simple) program has to be written to add these resources to the broker. Future versions of the broker will include this feature.

Programming the Broker:
While the examples shown here assume a command line environment, the Broker's capabilities are not limited to such a configuration. Other applications such as web portals have been built on top of the Broker by using its APIs (Application Programming Interfaces). The Broker was also designed to be independent of the input model. Therefore, it is possible to port other programming and scheduling models to the Broker. Some of this work has been described in the tech reports available at *http://www.gridbus.org/broker* Since a job consists of tasks and tasks themselves consist of commands, its is possible to build a job from its constituents and add it to the broker independent of the input plan _les. Similarly, a compute node can be added in the same way. An example of the possible uses of APIs is given in `examples/prog/GridTask.java`. Interested developers are encouraged to look at this code and the APIs.

3.4 Feedback

This section of the manual has attempted to explain the design of the Gridbus Broker, how to install and configure it and how to create programs around it. While some of the sections have not been detailed, notably the programming section, it is hoped that this provides enough material for a first exploration. The Gridbus Broker team would be very happy to answer any queries that you may have regarding the Broker. Relevant contact information is given below.

Contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Srikumar Venugopal (srikumar@cs.mu.oz.au)
Krishna Nadiminti (kna@unimelb.edu.au)

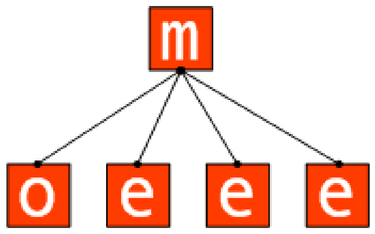
4 Alchemi

4.1 Introduction

Alchemi is a .NET-based grid computing framework that provides the runtime machinery and programming environment required to construct desktop grids and develop grid applications. It allows flexible application composition by supporting an object-oriented grid application programming model in addition to a grid job model. Cross-platform support is provided via a web services interface and a flexible execution model that supports dedicated and non-dedicated (voluntary) execution by grid nodes. It has been designed with the primary goal of being easy to use without sacrificing power and flexibility.
Alchemi includes:

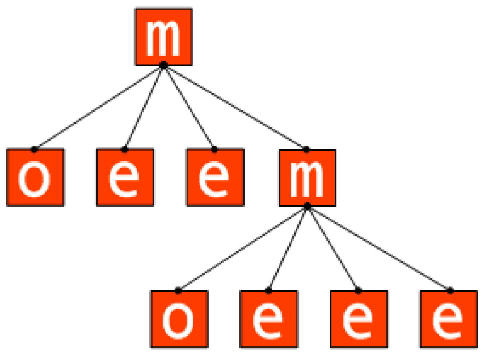
- The runtime machinery (Windows executables) to construct computational grids.
- A .NET API and tools to develop .NET grid applications and grid-enable legacy applications.

Uni-Level Grids
The simplest grid consists of one Manager. Multiple Executors are configured to connect to the Manager. One or more Owners can execute applications on the grid by connecting to the Manager.



A uni-level grid deployment

Multi-Level Grids
Multi-level grids are created by connecting Managers hierarchically. The key to accomplishing this is in Alchemi's inherent architecture, which allows a Manager to behave like an Executor towards a higher level Manager.



A multi-level grid deployment

The higher-level Owner has access to the entire computing power of the grid, while the lower-level Owner only has access to the computing power managed by the lower-level Manager. Grids can be scaled to an infinite number of levels in this fashion.

4.2 Prerequisites and Installation

The Alchemi distribution consists of:

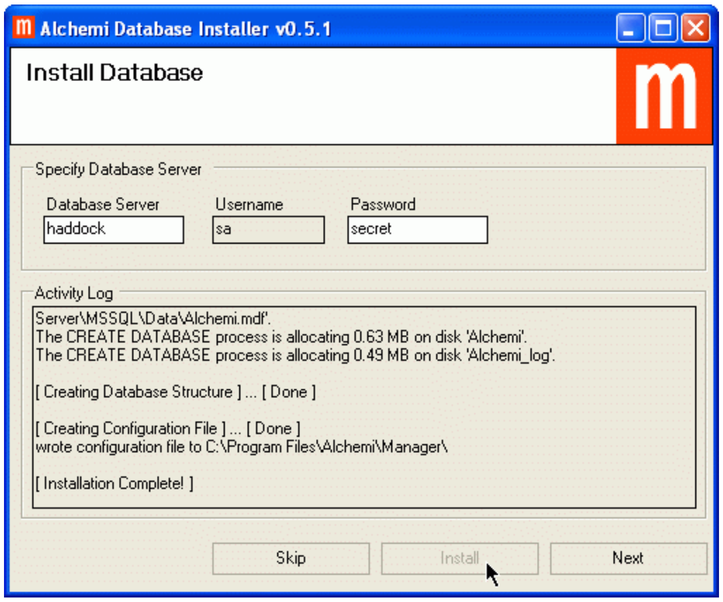
- Alchemi Manager
- Alchemi Executor
- Cross-Platform Manager
- SDK

Common Requirements

- Microsoft .NET Framework 1.1
- Manager

The Manager should be installed on a stable and reasonably capable machine. The Manager requires SQL Server 2000 or MSDE 2000. SQL Server / MSDE do not necessarily need to be installed on the same machine as the Manager. If using SQL Server, ensure that SQL Server authentication is enabled. Otherwise, follow the instructions at *http://www.asp.net/msde/default.aspx?tabindex=0&tabid=1* to install and prepare MSDE 2000 for Alchemi. Make a note of the system administrator (sa) password in either case.

Install the Manager via the Manager installer. Use the sa password noted previously to install the database during the installation:



Database Server: haddock
Username: sa
Password: secret

Activity Log
```
Server\MSSQL\Data\Alchemi.mdf'.
The CREATE DATABASE process is allocating 0.63 MB on disk 'Alchemi'.
The CREATE DATABASE process is allocating 0.49 MB on disk 'Alchemi_log'.

[ Creating Database Structure ] ... [ Done ]

[ Creating Configuration File ] ... [ Done ]
wrote configuration file to C:\Program Files\Alchemi\Manager\

[ Installation Complete! ]
```

Database installation during Manager installation

Cross Platform Manager

The Cross Platform Manager (XPManager) requires:

- Internet Information Services (IIS)
- ASP.NET

Install the XPManager web service via the Cross Platform Manager installer.

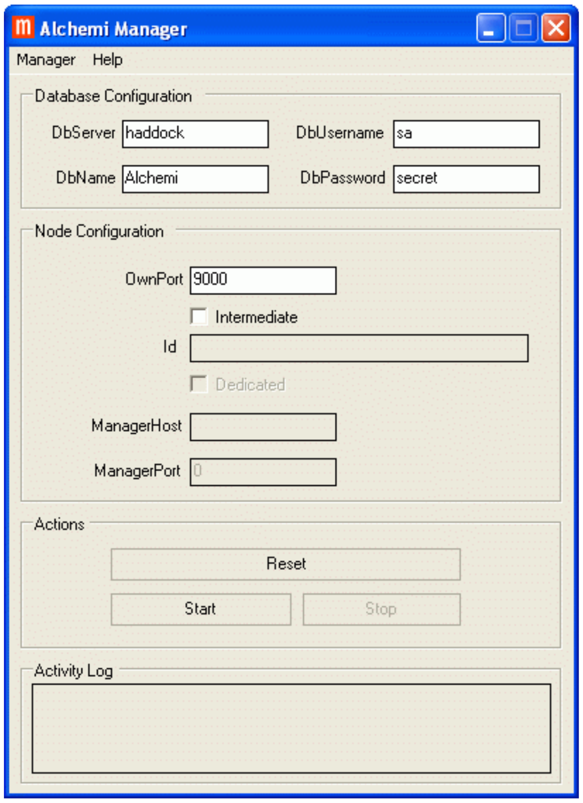
Configuration
If the XPManager is installed on a different machine that the Manager, or if the default port of the Manager is changed, the web service's configuration must be modified. The XPManager is configured via the ASP.NET `Web.config` file located in the installation directory (`wwwroot\Alchemi\CrossPlatformManager` by default):

```
<appSettings>
  <add key="ManagerUri" value="tcp://localhost:9000/Alchemi_Node"
/>
</appSettings>
```

Executor
Install the Executor via the Executor installer.

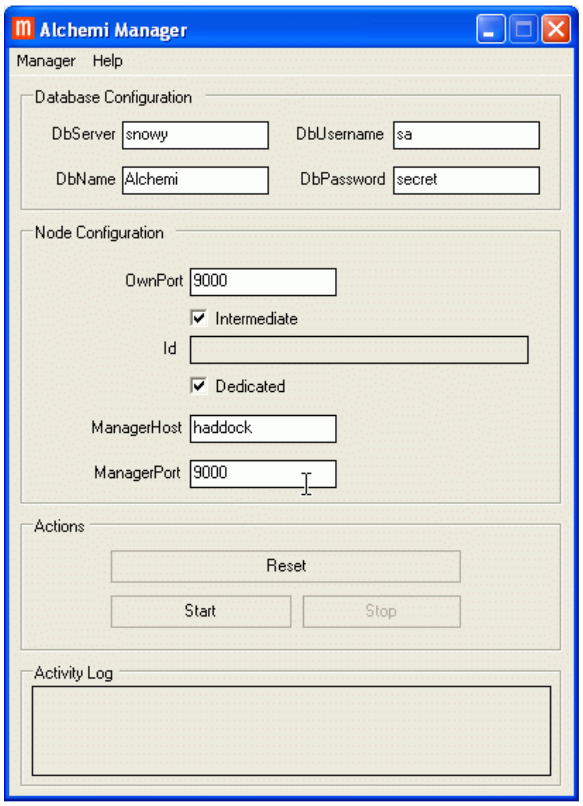
4.3 Using Alchemi

Manager
The Manager can be run from the desktop or `Start -> Programs -> Alchemi -> Alchemi Manager.`



DbServer: haddock
DbUsername: sa
DbName: Alchemi
DbPassword: secret
OwnPort: 9000
Intermediate
Id:
Dedicated
ManagerHost:
ManagerPort: 0

The database configuration settings used during installation automatically appear when the Manager is first started. For a stand-alone Manager (for a uni-level grid or the highest-level multi-level grid Manager) you only need to designate the "`OwnPort`" setting, which is the port that Manager listens on for all communication:



DbServer: snowy
DbUsername: sa
DbName: Alchemi
DbPassword: secret
OwnPort: 9000
[✓] Intermediate
Id:
[✓] Dedicated
ManagerHost: haddock
ManagerPort: 9000

Intermediate Manager Configuration
For an intermediate (lower-level) Manager that is part of a multi-level grid you need to check the "`Intermediate`" box and specify the higher-level Manager's host and port.
Since the Manager behaves like an Executor towards the higher-level Manager, you also need to specify whether it is a dedicated or non-dedicated "`Executor`" by checking/unchecking the "`Dedicated`" box. Run the manager program by clicking the "`Start`" button

Cross Platform Manager
The XPManager web service URL is of the format:

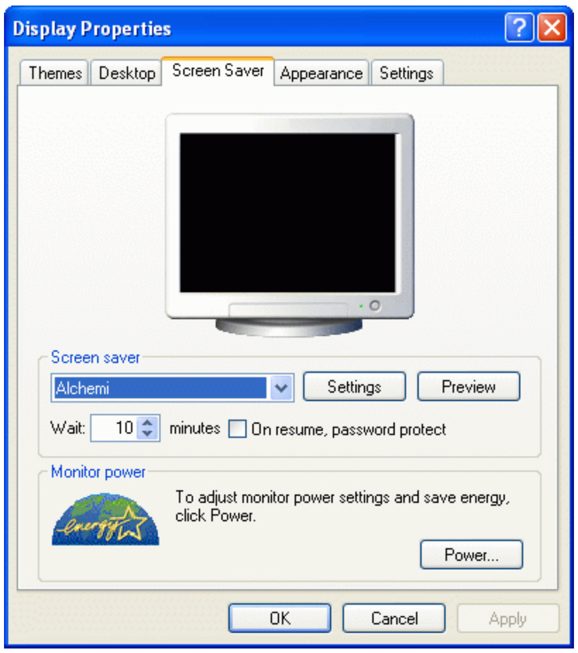
```
http://[host_name]/[installation_path]
```

The default is therefore:

```
http://[host_name]/Alchemi/CrossPlatformManager
```

The web service communicates with the Manager. The Manager must therefore be running and started for the web service to work.

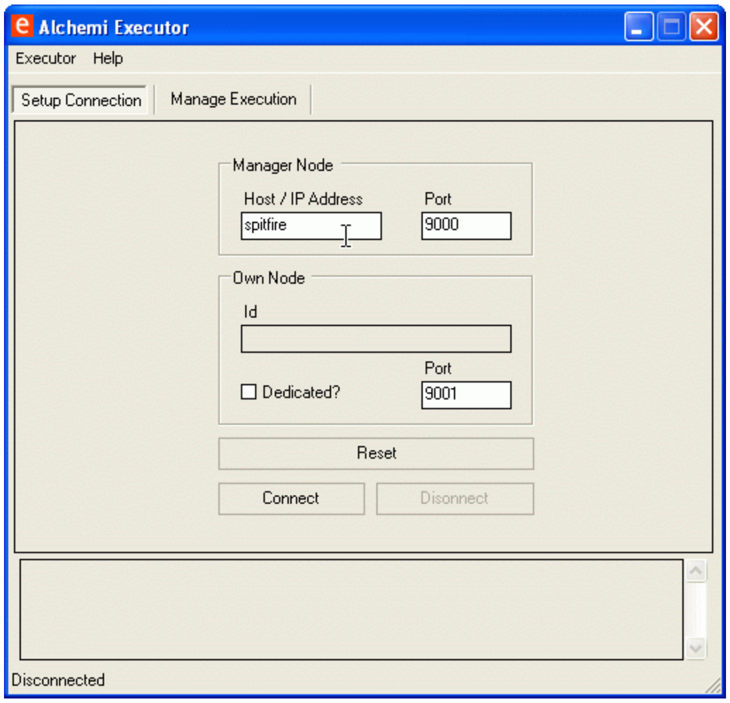
Executor
The Executor comes with a screensaver, which automatically activates the Executor. During installation, the "Screen Saver" properties are displayed with the screen saver selected.



Screen saver: Alchemi
Wait: 10 minutes ☐ On resume, password protect
Monitor power
To adjust monitor power settings and save energy, click Power.

Alchemi screen saver

The Executor is configured from the application itself.
The Executor can be run from the desktop or `Start -> Programs -> Alchemi -> Alchemi Executor`.



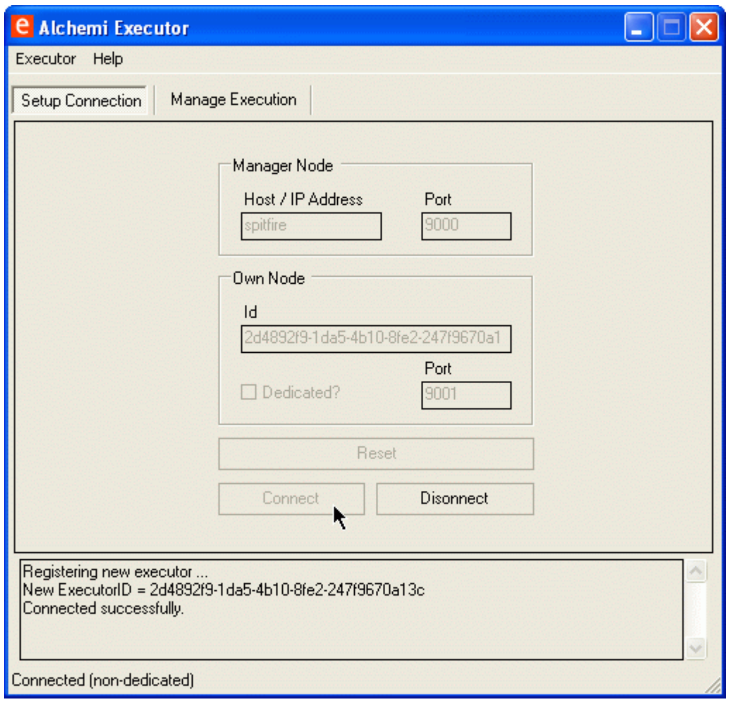
Manager Node
Host / IP Address: spitfire
Port: 9000
Own Node
Id:
☐ Dedicated?
Port: 9001
Disconnected

Executor configuration

You need to configure 2 aspects of the Executor:

- The host and port of the Manager to connect to.
- Dedicated / non-dedicated execution. A non-dedicated Executor executes grid threads on a voluntary basis (it requests threads to execute from the Manager), while a dedicated

Executor is always executing grid threads (it is directly provided grid threads to execute by the Manager).



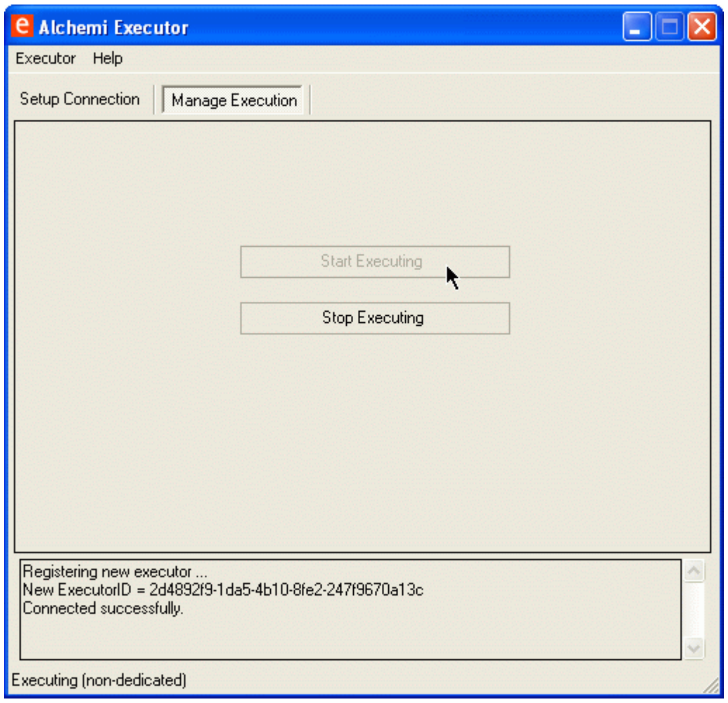
spitfire
9000
2d4892f9-1da5-4b10-8fe2-247f9670a1
9001

```
Registering new executor ...
New ExecutorID = 2d4892f9-1da5-4b10-8fe2-247f9670a13c
Connected successfully.
```

Connected (non-dedicated)

Executor connected to a Manager

Click the "`Connect`" button to connect the Executor to the Manager.
If the Executor is configured for non-dedicated execution, you can start executing by clicking the "`Start Executing`" button in the "`Manage Execution`" tab.



```
Registering new executor ...
New ExecutorID = 2d4892f9-1da5-4b10-8fe2-247f9670a13c
Connected successfully.
```

Executing (non-dedicated)

Non-dedicated execution

Note: When the Alchemi screen saver is activated, it automatically "presses" this button.

Verifying Grid Installation
You can verify successful setup of a grid by running a sample application on it. The Alchemi SDK contains a sample application "`PiCalculator`" that calculates the value of Pi to 100 digits.
Configure `PiCalculator` to point to a Manager:

- Examples `\PiCalculator\PiCalculatorExec\bin\debug\PiCalculator.exe.config`

Run it:

1) Examples `\PiCalculator\PiCalculatorExec\bin\debug\PiCalculator.exe`

For the Alchemi programming manual, please consult the Alchemi website: ***http://www.alchemi.net/***

4.4 Feedback

For further details about Alchemi please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Akshay Luther (akshayl@cs.mu.oz.au)

5 GridSim

5.1 Introduction

The GridSim toolkit supports modeling and simulation of a wide range of heterogeneous resources: single- or multi-processors, shared and distributed memory machines, such as PCs, workstations, SMPs, and clusters with different capabilities and configurations. GridSim can be used for modeling and simulation of application scheduling on various classes of parallel and distributed computing systems, such as clusters, grids, and P2P networks. The GridSim resource entities are being extended to support advanced reservation of resources and user-level setting of background load on simulated resources based on trace data. The GridSim toolkit provides facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations, and domains. It supports primitives for application composition, information services for resource discovery, and interfaces for assigning application tasks to resources and managing their execution. It also provides visual modeler interface for creating users and resources. These features can be used to simulate parallel and distributed scheduling systems such as resource brokers or Grid schedulers for evaluating performance of scheduling algorithms or heuristics. The GridSim Toolkit has been used to create a resource broker that simulates Nimrod-G for the design and evaluation of deadline and budget constrained scheduling algorithms with cost and time optimizations. It is also used to simulate a market-based cluster scheduling system (Libra) in a cooperative economy environment.

5.2 Prerequisites and Installation

The GridSim toolkit includes the java APIs to use for building simulations.
You do not need to compile GridSim source code. The JAR file is provided to compile and to run GridSim applications. The prerequisites include Java Virtual Machine 1.4.x, and the Java SDK.
The distribution includes the following jar files:

- gridsim.jar -- contains GridSim and SimJava class files
- simjava2.jar -- contains SimJava v2.0 class files only

To compile and run a GridSim application:

1) Go the directory where the GridSim Example 1 resides.
   In Unix or Linux : `cd $GRIDSIM/examples/Example01`
   In Windows : `cd %GRIDSIM%\examples\Example01`

2) Compile the Java source file
   In Unix or Linux :
   ```
   javac -classpath $GRIDSIM/jars/gridsim.jar:. Example1.java
   ```
   In Windows:
   ```
   javac -classpath %GRIDSIM%\jars\gridsim.jar;. Example1.java
   ```

3) Running the Java class file
   In Unix or Linux:
   ```
   java -classpath $GRIDSIM/jars/gridsim.jar:. Example1
   ```
   In Windows:

```
    java -classpath %GRIDSIM%\jars\gridsim.jar;. Example1
```

Note:

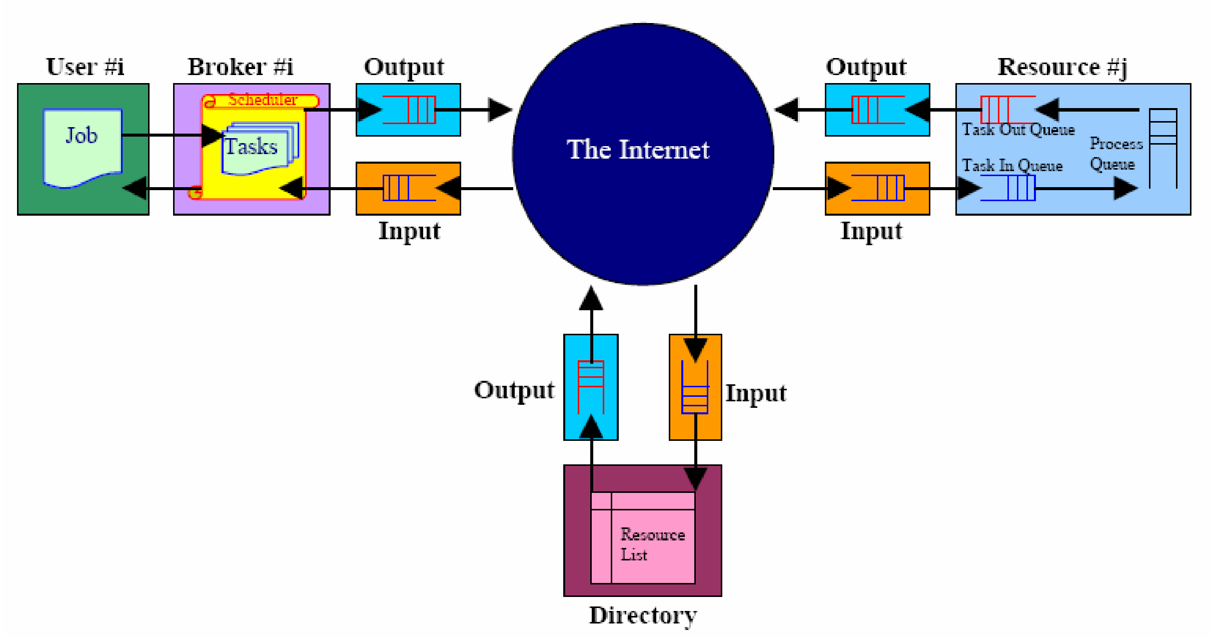
- $GRIDSIM or %GRIDSIM% is the location of the GridSim Toolkit package.
- Running GridSim of this version requires a lot of memory since there are many objects to be created. Therefore, it is recommended to have at least 512MB RAM or increase JVM heap size when running Java for large simulation experiments.
  For example:

```
    java   -Xmx300000000   -classpath   $GRIDSIM/jars/gridsim.jar:.
  Example1
```

(Max. heap size is 300MB in the above example).

5.3 Using GridSim

The Java-based GridSim discrete event simulation toolkit provides Java classes that represent entities essential for application, resource modeling, scheduling of jobs to resources, and their execution along with management. A schematic representation of interaction between GridSim entities during simulator execution is shown in the figure below.



The process of resource and application modeling for developing Grid schedulers/brokers is discussed below.

Resource Modeling:
In GridSim toolkit, we can create CPUs (also called Processing Elements (PEs)) with different MIPS (Million Instructions Per Second) or SPEC-like ratings. Then, one or more PEs can be put together to create a machine (a single CPU/SMP node). Such one or more machines can be put together to create a Grid resource. The resulting Grid resource can be a single processor, shared memory multiprocessors (SMP), or a distributed memory cluster of computers. These Grid

resources can be managed by time-shared or space shared scheduling systems depending on type of the resource. Generally, a single PE or SMP type Grid resource is managed by time-shared operating systems using round-robin scheduling policy and a cluster-like Grid resource is managed by space-shared Q-schedulers using different scheduling policies such as first-come-first-served (FIFO), back filling, shortest-job-first (SJF), and so on. For every Grid resource, the non-Grid (local) workload is estimated based on typically observed load conditions as well as the time zone of the resource. The network communication speed between user and resources are defined in terms of data transfer baud rate. When a resource entity is created, it registers resource information and contact details with the Grid Information Service (GIS) entity. This resource registration process is similar to GRIS (Grid Resource Information Server) registering with GIIS (Grid Index Information Server) in Globus system. The GIS can then be queried for list of resource in a given Grid domain to get resource handles that can be used to query resources directly for their capabilities, costs, and other configurations. Application Modeling GridSim does not define any specific application model explicitly. It is up to the developers (of schedulers and resource brokers) to define them. We have experimented with task-farming application model and we believe that other parallel application models such as process parallelism, DAGs, divide and conquer, etc., can also be modeled and simulated using GridSim. In GridSim, each independent task can be heterogeneous in terms of processing time and input files size. Such tasks/jobs can be created and their requirements can be defined through Gridlet objects. A Gridlet is a tiny GridApp that contains all information related to jobs and job execution management details such as jobs processing requirements, expressed in MIPS, disk I/O operations, the size of input files, etc. that help in computing execution time of remote resource and the size of output files. The GridSim toolkit supports a wide range of Gridlet management protocols and services that allows one to map or assign a Gridlet to a resource for execution and manage the same through out its life cycle. Each Grid user can be modeled with different characteristics/requirements such as:

- Types of job created e.g., job execution time, number of parametric replications etc.;
- Scheduling policy e.g., cost, time, or both minimization;
- Activity rate e.g., how often it creates new job;
- Time zone; and
- D- and B-factors, measured in the range [0,1], express deadline and budget affordability of the user.

A D- factor close to 1 signifies that the user is willing to set deadline of a job as long as required even when only the few slowest resources are available for the job. Similarly B-factor close to 1 signifies that the user is capable of spending as much money as required even when only the few expensive resources are available for the job. These factors are basically useful for determining deadline and budget values for a given scenario at runtime. Instead, users can also explicitly specify values for deadline and budget constraints similar to the way it is currently done in Nimrod-G.

Steps for Simulating Application Scheduling:

In this section we present high-level steps, with sample code clips, to demonstrate how GridSim can be used to simulate a Grid environment to analyze some application scheduling algorithms:

- First, we need to create Grid resources of different capability/speed like those in the real environment at different time zones and different policies (like time- or space-shared resources in World-Wide Grid (WWG) test-bed). We also need to create users with different requirements.
- Second, we need to model applications by creating a number of Gridlets (that appear similar to Nimrod-G jobs) and define all parameters associated with jobs. The Gridlets can be grouped together depending on application model for processing.
- Finally, we need to implement resource brokers. First, inquire Grid Information Service (GIS), then inquire for resource capability including cost, and then depending on scheduling

heuristics, strategy, or algorithms assign Gridlets to resources for execution. The scheduling policies can be systems-centric like those implemented in many Grid systems such as Condor-G or users-centric like Nimrod-G broker quality of services (QoS) and market-based economic models driven Deadline and Budget Constrained (DBC) time, cost, and both optimizations and policies

5.4 Feedback

This section of the manual has briefly described the concepts of how to build simlulations using the GridSim toolkit. For further details on GridSim, please refer to the paper *http://gridbus.org/papers/gridsimedu.pdf*

For further information about GridSim please contact:
Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Anthony Sulistio (anthony@cs.mu.oz.au)

6 Grid Workflow Management System

6.1 Introduction

Gridbus Workflow Engine (GWFE) facilitates users to link standalone applications and execute their workflow applications on Grids. GWFE provides a XML-based workflow language for the users to define tasks and dependencies. It also supports a just in-time scheduling system, thus allowing the resource allocation decision to be made at the time of task execution and hence adapt to changing grid environments. We also use tuple spaces approach to enable an event-driven scheduling architecture for simplifying workflow execution.

6.2 Prerequisites and Installation

- IBM TSpace
  A tspaces server is used as a workflow event server. It can be downloaded from *http://www.almaden.ibm.com/cs/TSpaces/*
- MySQL
  It is used to recode workflow execution status. It can be downloaded from *http://www.mysql.com/*
- Gridbus Market Directory
  It is used as directory service for Grid resource publication and discovery. It can be downloaded from *http://www.gridbus.org/gmd*
- Grid middleware support
  Globus toolkit 2.4 (*http://www.globus.org*)

We also use many other free Java libraries which are packaged with this distribution.

- o CoG 1.1 *http://www-unix.globus.org/cog/java/index.php*
- o Dom4j *http://www.dom4j.org/*
- o jGraph *http://www.jgraph.com/*

Installation:

- Un-tar the distribution `gridbusworkflow.tar`.
- Change into `GridbusWF` directory.
- Config property file *`WEProperties`*

  [Where to store intermediate data]
  - `STORAGEHOST` // host name
  - `STORAGEDIR` //the dicrectory
  - `STORAGEPORT` //gridftp port no for remote access

  [TSpaces]
  - `TSPACESHOST` // host name
  - `TSPACESPORT` // access port

  [Mysql]
  - `MYSQLURL` //database access url (e.g. jdbc:mysql://localhost/WE)
  - `MYSQLUSER` //user name
  - `MYSQLPASSWORD` //password

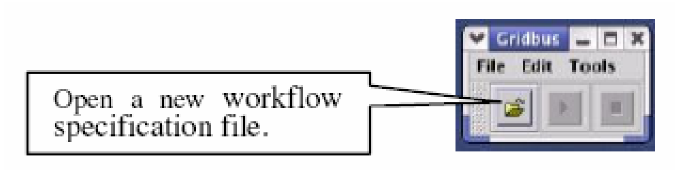
[Tuple space name for workflow engine]
`WORKFLOWEVENT` //tuple space for workflow engine (e.g. WorkflowEvent)

6.3 Using the Workflow Management System

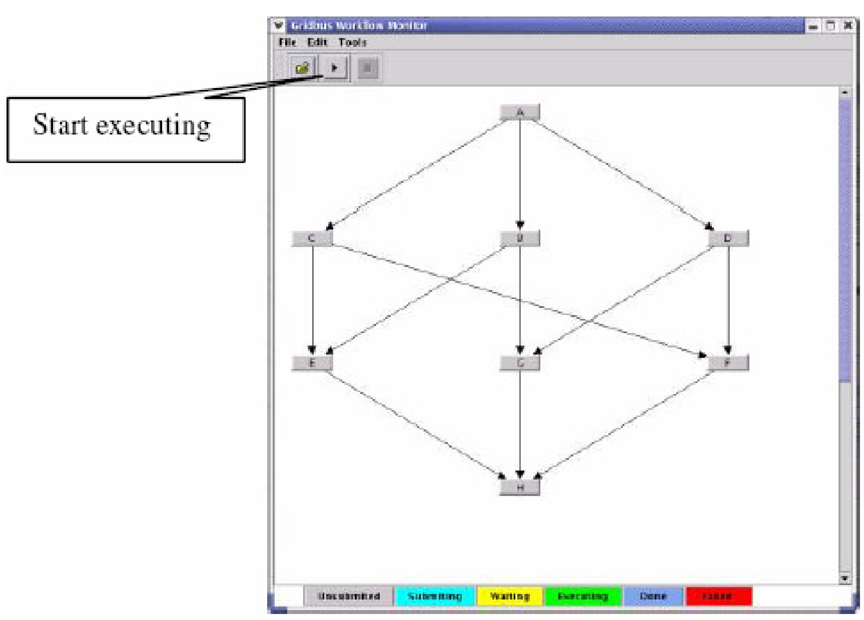
- Run workflow engine

```
[GridbusWF] .setenv.sh
[GridbusWF] java org.gridbus.workflowengine.monitor.WorkflowMonitor
```

- The main window shown in the figure below will be displayed:



- Open a workflow description file and start executing the workflow



- Monitor Workflow execution.

Some synthetic workflows and task processing programs are provided for testing purpose. Workflow description and related workflow applications can be found in example directory. In order to allow users to describe tasks and their dependencies, we defined a simple and flexible XML-based workflow language (xWFL). The workflow language provides the means to build new applications by linking standalone applications. For a detailed description of Workflow description files and the workflow language please refer to *http://gridbus.org/workflow/1.0/docs/wfwl.pdf*

6.4 Feedback

For more information about the Workflow Engine please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Jia Yu (jiayu@cs.mu.oz.au)

7 Grid Market Directory

7.1 Introduction

The Grid Market Directory (GMD) enables Grid service providers to publish their services and related costs to the public, so that consumers can browse through Web browser or query by using SOAP, to find a suitable service meeting their requirements. The GMD includes:

1. GMD Portal Manager:
   It is responsible for Web browser requests and generates web pages dynamically. The GMD Portal Manager is built on a web server.
2. GMD Query Web-services:
   It provides web-service access to the directory. Consumer's program can query GMD by using Simple Object Access Protocol (SOAP).
3. GMD Repository
   It is a database for service info.
4. GMD Query client APIs:
   Those APIs can be integrated with consumers programs; the developers of such a program only need to invoke the APIs directly without concerning presentation and transport layer technologies, such as SOAP and XML.
5. Command line client application to query the GMD

7.2 Prerequisites and Installation

- Java Virtual Machine 1.4.x
- Jakarta Tomcat
- Apache SOAP
- JDOM and
- MySQL or Other JDBC Compliant database.

The Grid Market Directory (GMD) programs are organized into two parts:

- Programs responsible for managing the portal
- Programs responsible for providing Web service for accessing GMD data.

GMD Portal Manager:

1. Basic files Included
   - All jsp pages and html pages are in `GMDPortalManager\jsp`.
   - The classes supporting jsp pages are in `GMDPortalManager\classes`.
   - Property file `GMDDB.prop` is used for Database location configuration.
2. Example for installing the GPM (GMD Portal Manager)

TOMCAT 3.2.4 configuration:

1. Make a new directory (`new_dir`) in `TOMCAT_HOME\webapps`.
2. Put all jsp and html files into `TOMCAT_HOME\webapps\new_dir`.
3. Put all classes into `TOMCAT_HOME\webapps\new_dir\WEB_INF\classes`.
4. Configure the database location information in the `GMDDB.prop`, e.g. jdbc driver name and password of accessing database.

5. Put `GMDDB.prop` into `TOMCAT_HOME\bin`.

Database tables:
Database can be MySQL or other database supporting JDBC.
There are three tables in the GMD database:

- GSPS (Grid Service Provides),
- GSPR (Grid Service Provider Registration),
- STS (Service Types);

GMD Web-service:
Basic files:

- `gmd.jar` contains all classes for Gridbus GMD web services, which is located in `GMDWebServices\server\lib`.
- Property file `GMDDB.prop` is for Database location configuration, it is the same file of the GMD Portal Manager.

Example for installing GMD web services:
TOMCAT 3.2.4:

- Put `gmd.jar` into `TOMCAT_HOME\lib`
- Config and put `GMDDB.prop` into `TOMCAT_HOME\bin` (if GMD Portal manager is not installed).

Client - Basic files:

- Client api is in the `gmdclient.jar`, which can be found in `GMDWebServices\client\lib`
- Other jar files supporting client api can be found in the same directory of GMD api jar file.
- Properties file `GMDQuery.prop` is for GMD location configuration, e.g. GMD host name and port name.
- The command-line client program (gridgmd.java) is a example for using GMD client API and in GMDWebServices\client\test.

7.3 Using GMD

How to run the command line program
Windows:

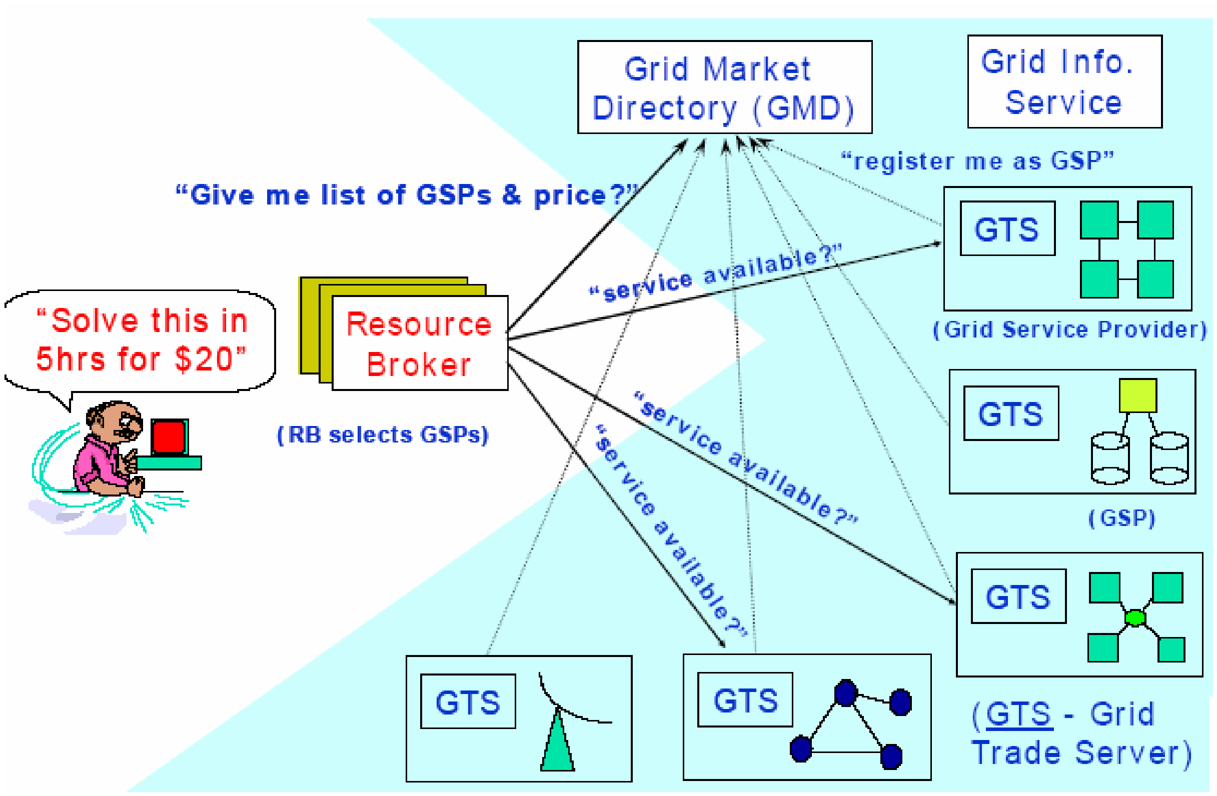
- `Set CLASSPATH=.;xerces.jar;soap.jar;activation.jar;mail.jar;jdom.jar;gmdclient.jar`
- `java gridgmd -h jarrett.cs.mu.oz.au -st globus`

The usage of the command-line program is shown below:

```
Usage: gridgmd [-h]||[-sh] [-sn] [-st] [-sp]
Options:
-a : Display all services in the Grid Market Directory
-h : GMD Host Name, the host address of the Grid Market Directory
-sh: Service Host Name, Query a service by its service host name
-sn: Service Name, Query a service by its service name
-st: Service Type Name, Query services by the service type
-sp: Query services by the service provider name
```

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.

The GMD provides web services [11] that applications can use to query the registry. 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.

For more details about the query API and web-service implementation please refer to *http://gridbus.org/papers/gmd.pdf*

7.4 Feedback

For more information about GMD, and how to deploy it, or plug-in to your own brokering mechanism, please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Jia Yu (jiayu@cs.mu.oz.au)

8 Grid Bank

8.1 Introduction

Computational Grids are emerging as a new infrastructure for Internet-based parallel and distributed computing. They enable the sharing, exchange, discovery, and aggregation of resources distributed across multiple administrative domains, organizations and enterprises. To accomplish this, Grids need infrastructure that supports various services: security, uniform access, resource management, scheduling, application composition, computational economy, and accounting. Grid Bank provides the infrastructure and services for accounting. The support of computational economy and accounting services can lead to a self-regulated accountability in grid computing.

GridBank can be thought of as a web service for Grid accounting and payment. GridBank uses SOAP over Globus toolkit's sockets, which are optimised for security. Clients use the same user proxy/component to access GridBank as they use to access other resources on the Grid. A user proxy is a certificate signed by the user, which is later used to repeatedly authenticate the user to resources. This preserves the Grid's single sign-in policy and avoids repeatedly entering the user password. Using existing payment systems for the Grid would not satisfy this policy.

8.2 Prerequisites for Installation, and Running GridBank

Requirements:

- Globus server bundle 2.x
- Java Virtual Machine 1.4.x
- MySQL

Installation and Steps for running GridBank:

1. Copy GridBank code to any installation directory. Make sure `GLOBUS_LOCATION` environment variable points to location where your Globus Toolkit is installed
2. To start setting up Globus Toolkit for use with GridBank. Generate header files and libraries of Globus I/O Module:

   ```
   $GLOBUS_LOCATION/sbin/globus-makefile-header –flavor=gcc32dbg
   globus_io
   ```

   Note: run the above command with the same flavor that you used when installing Globus Toolkit
3. Second, modify file called `globus_job_manager.c` in directory

   ```
   $GLOBUS_LOCATION/BUILD/globus_gram_job_manager-2.1.
   ```

   If you have different version of Globus, then just copy and paste resource usage code (search for "`GRIDBANK MODIFICATION`" keywords). Then execute in the same directory:

   ```
   make
   make install
   ```

4. This modification to Globus Toolkit will write job requests file and resource usage files to `/tmp` directory after job(s) finished executing. GridBank Charging Module operation depends on this file. The name of the file is the client's certificate name (with all '/' replaced by '_') followed by `.rur` extension.
5. Run Globus normally (e.g. `globus-job-run`) to make sure the resource usage file is generated.
   GridBank code assumes that you have MySQL installed in `/usr` directory (requires `/usr/include/mysql` and `/usr/lib/mysql` in order to compile as is; you can have MySQL installed in another directory, but you'd have to change `Makefile` in `gridbank/server/protocols` to point to the right directory).
6. Compile `soapcpp` and `xerces-c` libraries that come with GridBank (see instructions inside directories).
7. Create file in `/etc/grid-security` called "`template-accounts`":
8. Open the file and add local (system) account names. Each account name must by followed by new line character (just like in `grid-mapfile`).
9. Create GridBank database by executing following commands (partial paths are given): Replace "`globus_location`" variable in all subdirectories of gridbank with the path to your Globus installation (i.e. `GLOBUS_LOCATION`).
10. Run the following commands:

```
cd gridbank/server/accounting
mv Makefile MakefileNormal
mv MakefileCreateDB Makefile
make
su
./createdb -admin <Administrator Certificate Name> -bank <bank
number> -branch <branch number>
exit
mv Makefile MakefileCreateDB
mv MakefileNormal Makefile
rm createdb
rm *.o
```

    Note: this assumes that MySQL server is running as root and must have root-only access (for security reasons). `<Administrator Certificate Name>` is the (Globus) X509v3 Certificate Subject Name of the initial administrator who can create other administrators and perform account management operations (i.e. open account, deposit, withdraw, etc.). `<bank number>` is usually "1" for GridBank, and `<branch number>` is the number of your GridBank branch and has to be unique. GridBank account numbers consists of the following: 2 digits for bank number, 4 digits for branch number, and 8 digits for account number. E.g. 01-0001-00000001. The branch number has to be unique. So each GridBank branch is identified by the first 6 digits, e.g. 01-0001. So to create database for branch 01-0012 you need to execute

```
./createdb -admin "bla bla" -bank 1 -branch 12
```

11. Replace "`globus_location`" variable in all Makefiles in all subdirectories (ie. `gb_io, server, gbpm, gbcm`) with the path to your Globus installation (i.e. the value of `$GLOBUS_LOCATION`).
12. Compile GridBank Server:

```
cd gridbank/server
```

`sh install` (or `csh install` depending on your shell)

This should create `'gbserver'` executable `in gridbank/server` directory. Run `'gbserver'` as root:

```
su
./gbserver -dbname gridbank<bank no.>_<branch no.>
```

Server is started by specifying the name of the database. For example, if your branch is 01-0012, then execute

```
./gbserver -dbname gridbank1_12
```

13. Compile GridBank Charging Module in `(gridbank/gbcm)`. Make sure that `'gbcm.conf'` file contains the right configuration parameters. The first parameter, "`gbcm tcp port`", is the port number to run GridBank Charging Module server. The default is set to 3000. That's what the GridBank Payment Module expects, but can be reconfigured. If you change this parameter, you also need to change "`default gbcm port`" parameter in `gridbank/gbpm` directory as well. "`charging enabled`" is either `0` or some other integer (usually `1`). Value `0` disables charging, and will only report resource usage. "`gb server address`" specifies the URL of the GridBank server. Note that by default server runs on port 2500. The rest of parameters specify hardware characteristics of the resource. The parameters and their corresponding values are self-explanatory (and should be taken from hardware specs). This should create `'gbcm'` executable in the same directory. Run `'gbcm'` as root.
14. Compile GridBank Payment Module in `(gridbank/gbpm)`. This should create `'gb-pre-pay'` executable in `gridbank/gbpm` directory. Use this command to pre-pay for the resource (i.e. to set up a local account) and then use Globus commands normally to submit job(s).E.g. to run /bin/date command on machine 'somehost', execute:

    ```
    ./gb-pre-pay -amount 1.2 -destacc 01-0001-00000002
    somehost.csse.uwa.edu.au
    ```

    (Make sure you run `$GLOBUS_LOCATION/bin/grid-proxy-init first`)

    ```
    $GLOBUS_LOCATION/bin/globus-job-run somehost.csse.uwa.edu.au
    /bin/date
    ```

    Don't forget to set up your `'gbpm.conf'` file.
    "`account`" is the GridBank account number of the client. Each client obtains an account by asking administrator to create one for them. When administrator opens a new account, account number is returned and must be saved as this parameter. "`gb server address`" is the URL of the GridBank server. "`default gbcm port`" is the port number that GridBank Charging Module runs on. By default, it's 3000.
15. There is a GUI which you can run:
    ```
    java GBClient
    ```

Make sure that you have `'libgbapi.so'` located in `gridbank/gbpm` directory in your environment. You can set up `LD_LIBRARY_PATH` environment to point to this lib.Also make sure you run `$GLOBUS_LOCATION/bin/grid-proxy-init` first.

The GUI allows GridBank clients (users) to request account statements. It allows administrators to create other administrators, open and close accounts, deposit and withdraw. The deposit and withdraw operations are the means by which currency is created in the GridBank. The amount of currency that should be created must correspond to resource reservations. This area is under research. For details about GridBank implementation refer to
http://www.csse.uwa.edu.au/~barmouta/gridbankarticle.pdf

8.3 Feedback

For further details please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Alex Barmouta (barmouta@csse.uwa.edu.au)

9 Gridscape

9.1 Introduction

Gridscape is a tool that enables the rapid creation of interactive and dynamic test-bed portals (without any programming effort). Gridscape primarily aims to provide a solution for those users who need to be able to create a grid test-bed portal but do not necessarily have the time or resources to build a system of their own from scratch. The design aims of Gridscape are that it should:

- Allow for the rapid creation of Grid test-bed portals;
- Allow for simple portal management and administration;
- Provide an interactive and dynamic portal;
- Provide a clear and user-friendly overall view of Grid test-bed resources; and
- Have a flexible design and implementation such that core components can be leveraged, it provides a high level of portability, and a high level of accessibility (from the browsers perspective).

Gridscape itself consists of two key individual components:

- a web application and
- a an administrating tool

that are implemented in Java by following MVC (Model-View-Controller) based, Model-2 type architecture.

9.2 Prerequisites and Installation

Before we can deploy and start working with our web portal, there are some other supporting technologies which need to be installed. These technologies are Java, Jakarta Tomcat and Jakarta Struts.

- Java Virtual Machine 1.4.x and SDK (J2SE)
- Jakarta Tomcat 4.1
- Jakarta Struts 1.1.0

Note: The Struts class and jar files has been included with Gridscape package for your convenience.

Installing GridScape Web Application
Once you have Jakarta Tomcat and Struts installed and working, the GridScape template web application can be deployed.

- Download the GridScape Web Archive (`gridscape.war`).
- Place the GridScape Web Archive into the Tomcat `webapps` directory.
- The application will be deployed automatically when Tomcat is running.

If you have Tomcat running, you should now be able to view the template portal online.

Installing GridScape Admin Tool
In order for you to easily customise your testbed portal you should install the GridScape Admin Tool.

- Download the GridScape Admin Tool archive.
- Extract files into a new folder.
- The admin tool should be ready to run.

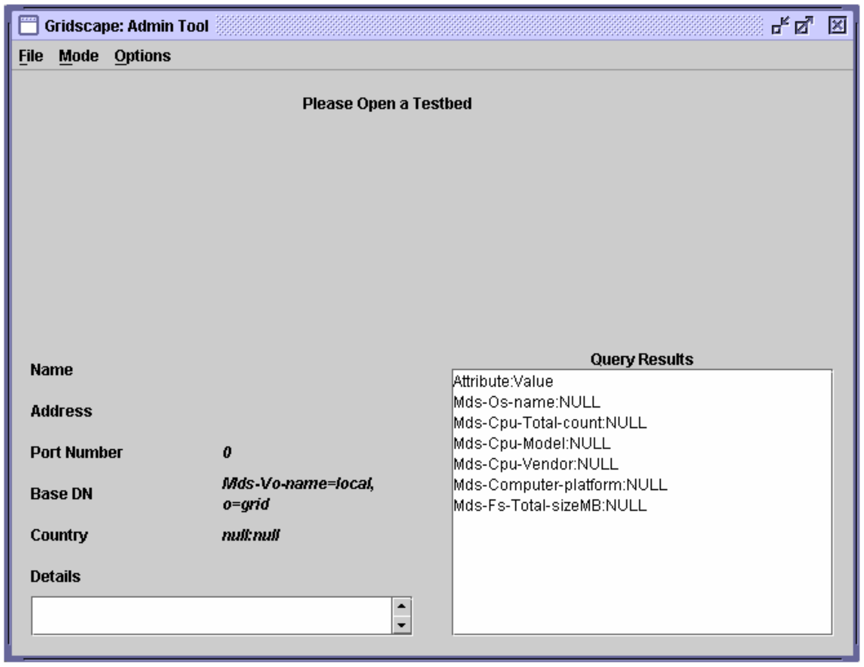
9.3 Using GridScape

Using the Admin Tool

Now that we have successfully installed the GridScape template Web Application and the Admin Tool, we can begin to create our own test bed portal.

Launch GridScape Admin Tool

To start with, launch the GridScape Admin Tool. The first time this is run you will be asked to open a test bed:



Please Open a Testbed

Name	
Address	
Port Number	*0*
Base DN	*Mds-Vo-name=local, o=grid*
Country	*null:null*
Details	

Query Results

```
Attribute:Value
Mds-Os-name:NULL
Mds-Cpu-Total-count:NULL
Mds-Cpu-Model:NULL
Mds-Cpu-Vendor:NULL
Mds-Computer-platform:NULL
Mds-Fs-Total-sizeMB:NULL
```

GridScape Admin Tool

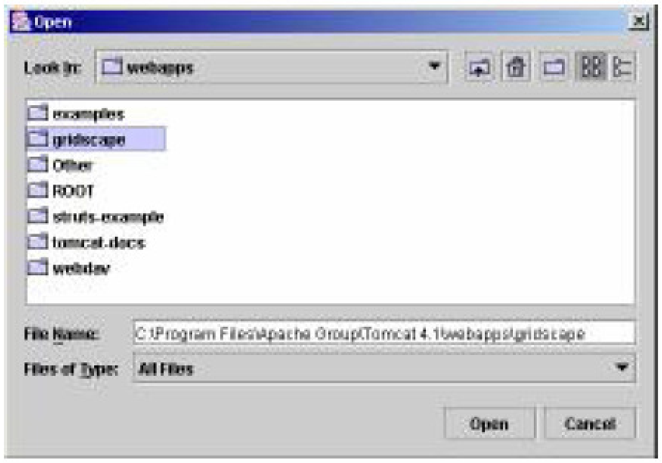
Opening a test bed

We now need to open a test bed in order for us to start our work. You may have an existing test bed which you can open for editing, but for our first portal we want to edit the template provided with GridScape.

- In the “`File`” menu, select the “`Open`” item.

Selecting the “`Open`” menu item will bring up a dialog, as shown below, to allow you to locate the test bed portal you wish to edit. In this case it will be the GridScape template.

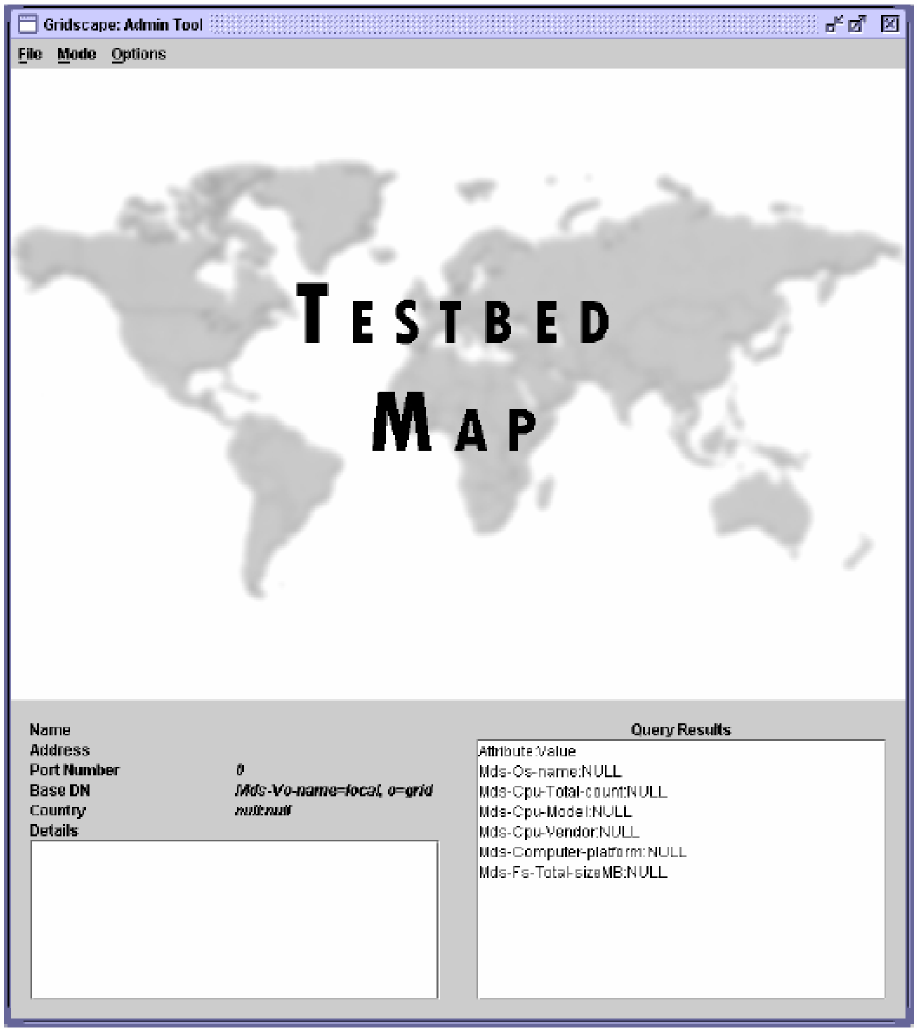
- Locate the GridScape web application which was deployed under Tomcat.
- Select “`Open`”.



File Name: C:\Program Files\Apache Group\Tomcat 4.1\webapps\gridscape
Files of Type: All Files

Open Dialog

The Admin Tool will open the web application and the interface will change to display the details of the test bed you have elected to open. Shown below is the opened template web application, ready for customisation.



Name	
Address	
Port Number	*0*
Base DN	*Mds-Vo-name=local, o=grid*
Country	*null:null*
Details	

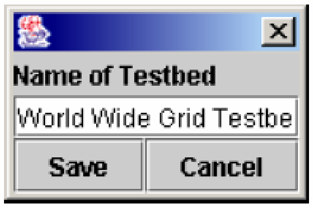
```
Attribute:Value
Mds-Os-name:NULL
Mds-Cpu-Total-count:NULL
Mds-Cpu-Model:NULL
Mds-Cpu-Vendor:NULL
Mds-Computer-platform:NULL
Mds-Fs-Total-sizeMB:NULL
```

Opened Testbed Template

Changing the Testbed Name

The first step we will take in customising the template test bed portal is to change its name.

- In the "`Options`" menu, select the "`Testbed Name`" item.
- Enter the name of your test bed. We have entered the name for the World Wide Grid Test bed portal.
- Select the "`Save`" option.



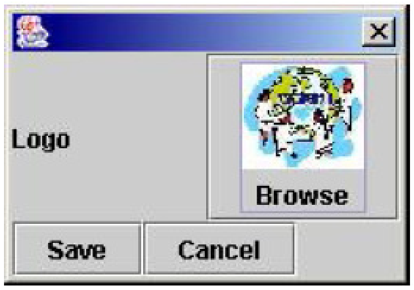
Testbed name dialog

Once you have saved the new name, the web application will be updated immediately.

Changing the Test bed Logo

Next we will choose a logo for our test bed which will be displayed next to the Test bed name on our test bed portal.

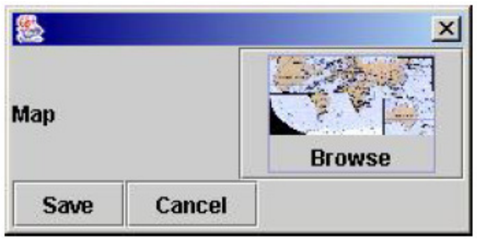
- In the "`Options`" menu, select the "`Test bed Logo`" item.
- Select the "`Browse`" option. A dialog will be displayed allowing you to locate a suitable image to be used as your test bed logo.
- Find a suitable image.
- Select the "`Set`" option. The figure below shows the logo selected for our World Wide Grid Test bed portal.
- Select the "`Save`" option on the remaining dialog.



Once you have saved the new logo, the web application will be updated immediately.

Changing the Test bed Map

Now we will select the map to be used for our test bed portal. This is the map on which the locations of Grid resources will be displayed.



Testbed Map Dialog

In the “`Options`” menu, select the “`Testbed Map`” item.

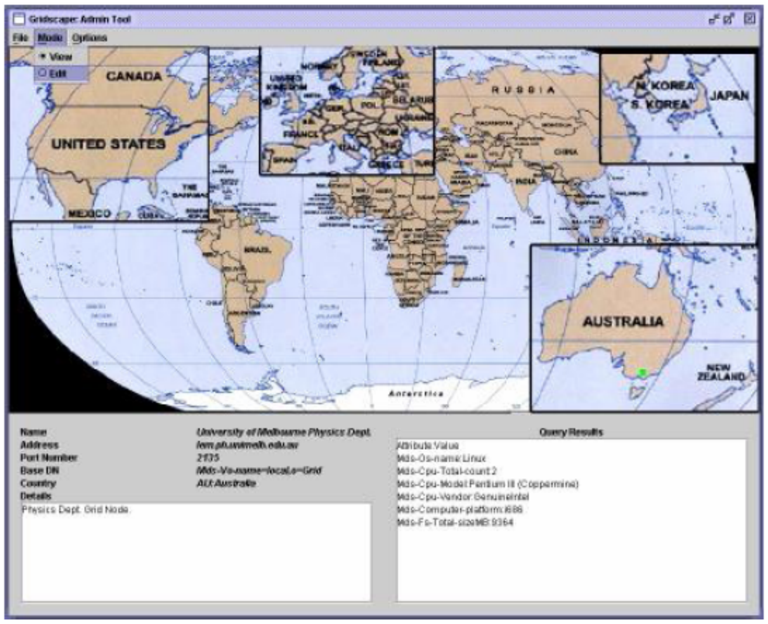
- Select the “`Browse`” option. A dialog will be displayed allowing you to locate a suitable image to be used as a map for your test bed.
- Find a suitable image.
- Select the “`Set`” option. The figure below shows the map selected for the World Wide Grid Test bed portal.
- Select the “`Save`” option on the remaining dialog.

Once you have saved the map, the Admin Tool should update to display the new map and the web application will be updated immediately.

Changing Modes

Besides being able to simply modify things such as the Name, Logo and Map for our test bed portal, we still need to deal with all of our Grid resources which make up our test bed. In GridScape we discuss our Grid resources in terms of Locations, where a Location describes not only a single Grid resource, but its physical location as well. The GridScape Admin Tool has been designed to operate in two different modes - View mode and Edit mode, which allows us to interact with Locations in different ways. The figure below shows changing the mode from View to Edit.

- View Mode: In view mode you can select individual Locations to query. Once a Location has been selected its attributes will be displayed in the “`Query Results`” panel.
    - o To enter View mode, select the “`View`” item from the “`Mode`” menu.
- Edit Mode: In edit mode you can add new Locations to your test bed, edit Locations which already exist, and remove and Locations you have previously added.
    - o To enter Edit mode, select the “`Edit`” item from the “`Mode`” menu.



Name	University of Melbourne Physics Dept.
Address	lem.ph.unimelb.edu.au
Port Number	2135
Base DN	Mds-Vo-name=local,o=Grid
Country	AU: Australia
Details	

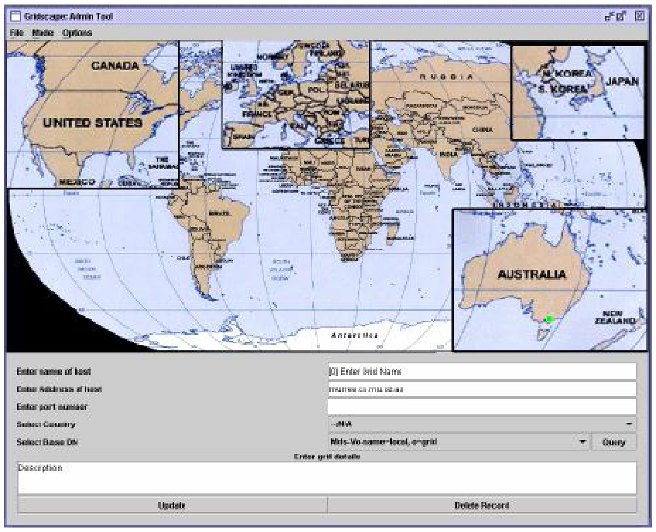
Physics Dept. Grid Node.

```
Attribute:Value
Mds-Os-name:Linux
Mds-Cpu-Total-count:2
Mds-Cpu-Model:Pentium III (Coppermine)
Mds-Cpu-Vendor:GenuineIntel
Mds-Computer-platform:i686
Mds-Fs-Total-sizeMB:9364
```

Adding a new Location

The next step in creating your test bed portal is to add your Grid resources (Locations).

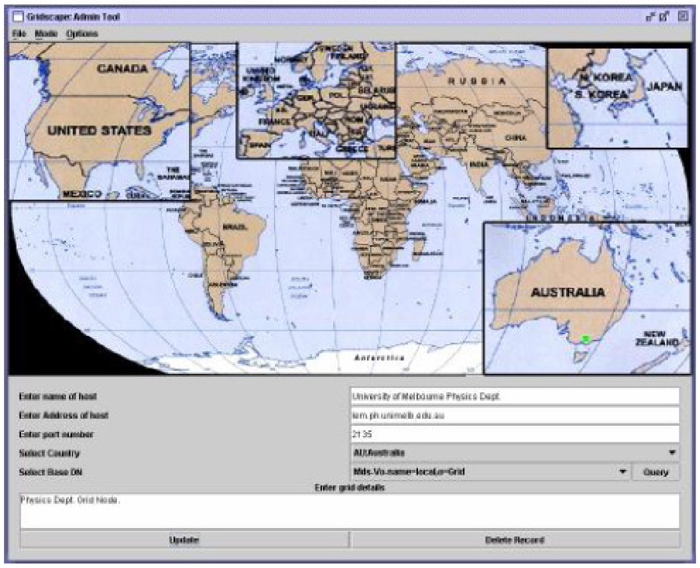
- Enter Edit Mode
- Left-click the mouse on an area of the map where you wish to position your Location. The figure below shows the result of this action. A new Location has been added to the test bed with default values.



Enter name of host: [0] Enter Grid Name
Enter Address of host: mumei.cs.mu.oz.au
Enter port number:
Select Country: --/N/A
Select Base DN: Mds-Vo-name=local, o=grid
Enter grid details
Description

Enter the name of the Location in the space provided.

- Enter the address of the Location in the space provided. This address will be used to query the Location.
- Enter the port corresponding to the address used to query the Location.
- Select the country in which this Location is located.
- Select an appropriate Base DN (for MDS purposes). The default setting should be appropriate in most instances. However, if this is not the case you may select "`Query`" to show a list of available options.
- Enter a short description about the Location.
- Choose the "`Update`" option to store the information just entered about this Location.



Enter name of host: University of Melbourne Physics Dept.
Enter Address of host: lem.ph.unimelb.edu.au
Enter port number: 2135
Select Country: AU/Australia
Select Base DN: Mds-Vo-name=local,o=Grid
Enter grid details
Physics Dept. Grid Node.

This process should be continued until all the resources/Locations for your test bed have been added.

Moving a Location

If you notice that one of your Locations is not positioned correctly on your map, you can easily move the Location to its correct place.

- Enter "`Edit`" mode.
- Find the Location you wish to move.
- Left-click the mouse on the Location and drag to the desired position.
- Release the mouse button.

All your Locations may be moved around as described until you are happy with your test bed.

Editing a Location

Now that we have added all the Locations to our test bed we may want to edit some of the details – perhaps we made a spelling error or the address of one of our Locations has changed. In such cases we may wish to edit the details of one of our Locations.

- Enter "`Edit`" mode.
- Find the Location you wish to edit.
- Left-click the mouse on the Location.
- Make necessary changes to any of the fields.
- Select "`Update`" to store the changes.

Deleting a Location

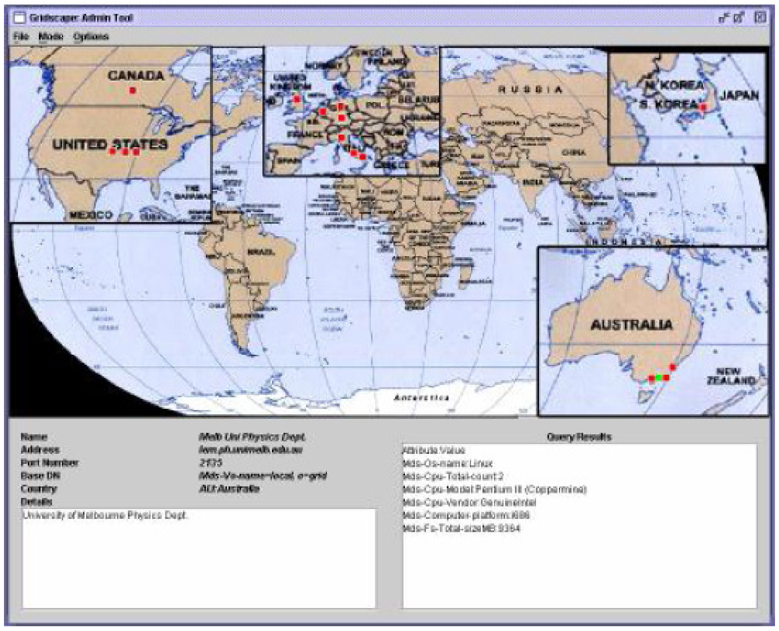
We now know how to Add, Move and Edit Locations within our test bed, but what about removing unwanted Locations? Whether you have accidentally added a Location you no longer want, or if you just need to remove a Location which is no longer needed, removing Locations is straightforward.

- Enter "`Edit`" mode.
- Find the Location you wish to delete.
- Left-click the mouse on the Location.
- Select the "`Delete Record`" option.

Querying a Location

Our test bed portal is now complete. We can now query various Locations if we want to view some information about it. Querying a Location will fetch attribute information and display it in the results panel.

- Enter "`View`" mode.
- Find the Location you wish to query.
- Left-click the mouse on the Location. The figure below shows the results of querying one of the Melbourne University Physics Department Location within the World Wide Grid Test bed portal.



Name Melb Uni Physics Dept.
Address lem.ph.unimelb.edu.au
Port Number 2135
Base DN Mds-Vo-name=local, o=grid
Country AU:Australia
Details
University of Melbourne Physics Dept.

Query Results
```
Attribute:Value
Mds-Os-name:Linux
Mds-Cpu-Total-count:2
Mds-Cpu-Model:Pentium III (Coppermine)
Mds-Cpu-Vendor:GenuineIntel
Mds-Computer-platform:i686
Mds-Fs-Total-sizeMB:9364
```

Query Results

Saving Changes

Now that we have added and modified all of the Locations for our test bed portal we will need to save them.

- In the "`File`" menu, select the "`Save`" item.

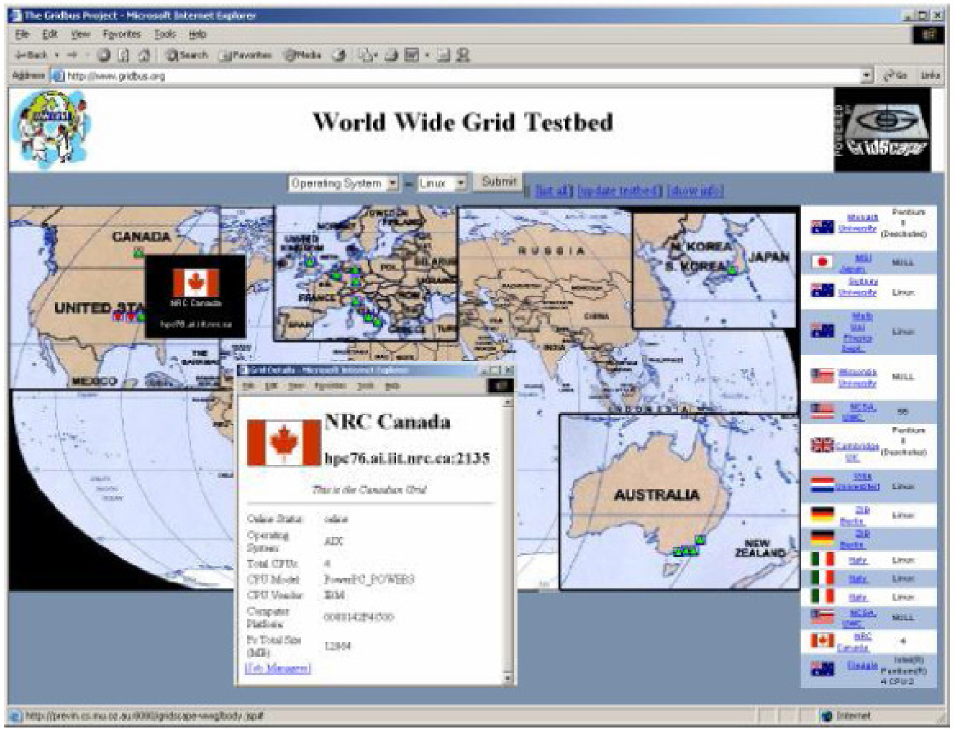
If we do not select the save option, our Locations will be lost and will not appear online in our web application and will not be remembered for the next time we run the Admin Tool.

Finishing

Once we are happy with our test bed portal we can now exit the Admin Tool.

- In the "`File`" menu, select the "`Exit`" item.

The test bed portal creation is complete! We can now go and visit our web test bed portal online.



NRC Canada

hpc76.ai.iit.nrc.ca:2135

This is the Canadian Grid

Online Status	online
Operating System	AIX
Total CPUs	4
CPU Model	PowerPC_POWER3
CPU Vendor	IBM
Computer Platform	00081428 4C00
Fs Total Size (MB)	12864

[Info Managers]

Monash University	Pentium II (Dedicated)
AIST Japan	NULL
Sydney University	Linux
Melb Uni Physics Dept.	Linux
Wisconsin University	NULL
NCSA, UIUC	95
Cambridge UK	Pentium II (Dedicated)
VBA University	Linux
ZIB Berlin	Linux
ZIB Berlin	
Italy	Linux
Italy	Linux
Italy	Linux
NCSA, UIUC	NULL
NRC Canada	4
Gadgie	Intel(R) Pentium(R) 4 CPU 2

World Wide Grid (WWG) Test bed Portal Generated Using Gridscape

9.4 Feedback

For further details please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Hussein Gibbins (hag@cs.mu.oz.au)

10 G-Monitor

10.1 Introduction

G-Monitor is a portal developed to interface with the Grid Resource Broker (GRB) and provide the user with the ability to not only monitor, control, and steer execution of application jobs on Global Grids but to provide functionality such as Grid level resource discovery, grid economy, scheduling algorithms based on grid economy.
G-Monitor provides the user with a ubiquitous interface which can be used in any location to access any broker system the user desires. The user only needs access to a web browser and with it they are able to fully utilize the functionality provided by G-Monitor. As G-Monitor is a web-based portal the user does not have to worry about setting up SSH or exporting X displays, which can all be made more complicated depending on their firewall settings.
G-Monitor can interact with either the Gridbus or Nimrod G brokers GRB (Grid Resource Broker). These brokers use low-level services provided by low-level middleware such as Globus and Legion. 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. With the use of a web browser the user is able to fully utilize the functionality of G-Monitor outlined in the Main feature section.
G-Monitor comprises of a series of perl scripts. These scripts provide a means for the user to retrieve grid information from the nimrod server via a web interface. The perl scripts require Apache web server to be running and take advantage of the CGI module.
The G-Monitor system receives requests from the user web browser, which it then parses and makes a socket connection to the appropriate Nimrod server. Using this connection it retrieves the information from the Nimrod server and generates HTML for the user to view via their browser.

10.2 Prerequisites and Installation

Server-side requirements:

- gd1_4_tar.gz
- apache_1.3.27.tar.gz
- gnuplot 3.7.1 compiled with gif support
- perl 5.005 +
- perl FTP module
- perl Telnet module
- telnet daemon
- ftp daemon (to allow experiment files to be transferred to the broker)

G-Monitor has been successfully tested to work with the following browsers:

- IE 5
- Netscape 4.77

Installation Instructions:

1. Install apache:

```
gzip -cd apache_1.3.27.tar.gz | tar -xv
cd apache_1.3.27/
./configure
make
make install
```

2. Install gd:
```
 gzip -cd  gd1_4_tar.gz | tar -xv
   cd gd1.4/
   make
   cp libgd.a /lib/
   cp gd.h  /usr/include/
```

3. Install gnuplot:
```
 gzip -cd gnuplot-3.7.1/ | tar -xv
   ./configure --with-gd
   make
   make install
```

4. `/usr/local/apache/bin/apachectl start`

5. Navigate browser `to http://127.0.0.1/` - to check apache is up.

6. `gzip -cd gmonitor2.tar.gz | tar -xv`

7. ensure that permissions are correct!! eg, if apache runs under the user "nobody" make sure all files have ownership to that user

```
   chown -R nobody.nogroup /usr/local/apapche/cgi-bin/gmonitor/*
   chown -R nobody.nogroup /usr/local/apapche/htdocs/gmonitor/*
```

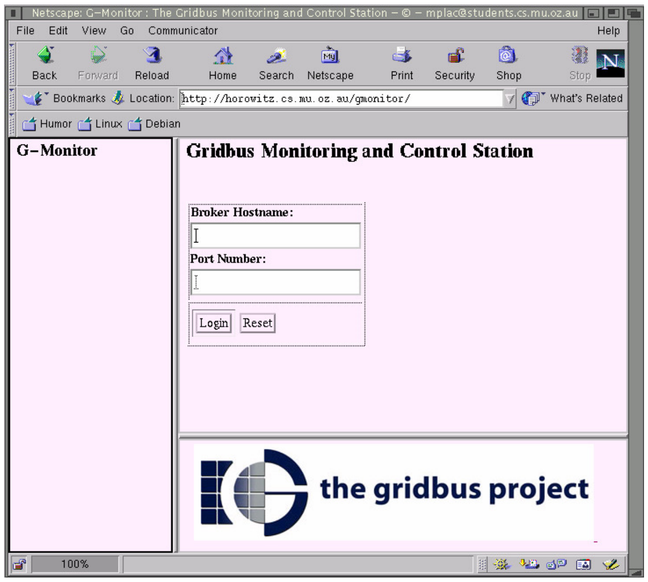
8. load up browser IE 5, netscape/communicate 4.77 and point your browser to:
`http://<Gmonitor server hostname>/gmonitor/`

10.3 Using G-Monitor

Gmonitor provides a web interface for the nimrod brokering system. It allows the user to remotely monitor and control a grid system. Gmonitor enables the user to:

- set quality of service parameters (Deadline,Budget,Optimisation,start, stop shutdown)
- Monitor/Control Jobs Information (Job name, status, remarks,grid node, Execution Time)
- Resources (Server name, Host name, Service cost, status, remarks)
- Experiment status (Deadline,Budget,Job status,host status)

Logging in:
Open a new web browser and navigate to "`http://server name/gmontior/`". You should be presented with the login screen as shown below:


Netscape: G-Monitor : The Gridbus Monitoring and Control Station – © – mplac@students.cs.mu.oz.au
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://horowitz.cs.mu.oz.au/gmonitor/ What's Related
Humor Linux Debian
Login Reset
the gridbus project
100%

G–Monitor

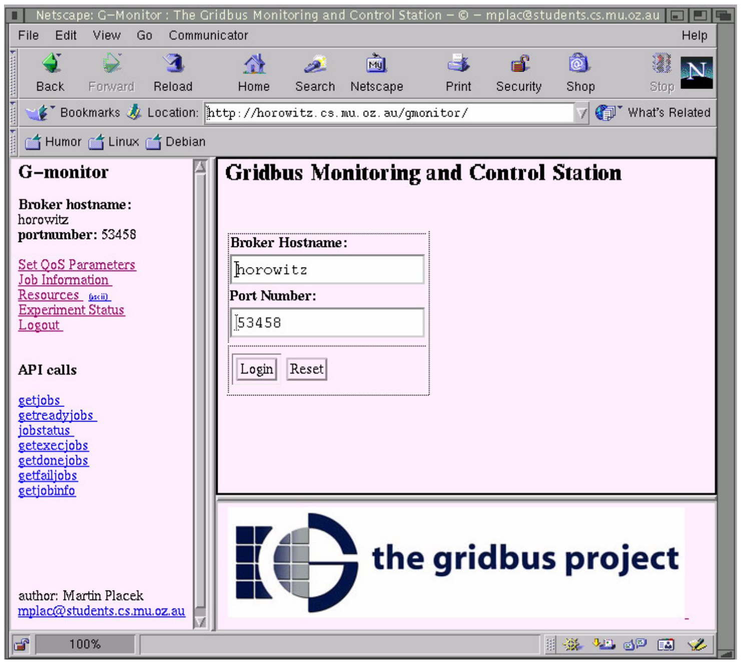
Gridbus Monitoring and Control Station

Broker Hostname:

Port Number:

Login screen

Once a Broker Host-name and port number is specified press login and you should find that a menu of options is presented on the left hand side.


Netscape: G-Monitor : The Gridbus Monitoring and Control Station – © – mplac@students.cs.mu.oz.au
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://horowitz.cs.mu.oz.au/gmonitor/ What's Related
Humor Linux Debian
Login Reset
the gridbus project
100%

G–monitor

Broker hostname: horowitz
portnumber: 53458

Set QoS Parameters
Job Information
Resources (ascii)
Experiment Status
Logout

API calls

getjobs
getreadyjobs
jobstatus
getexecjobs
getdonejobs
getfailjobs
getjobinfo

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

Gridbus Monitoring and Control Station

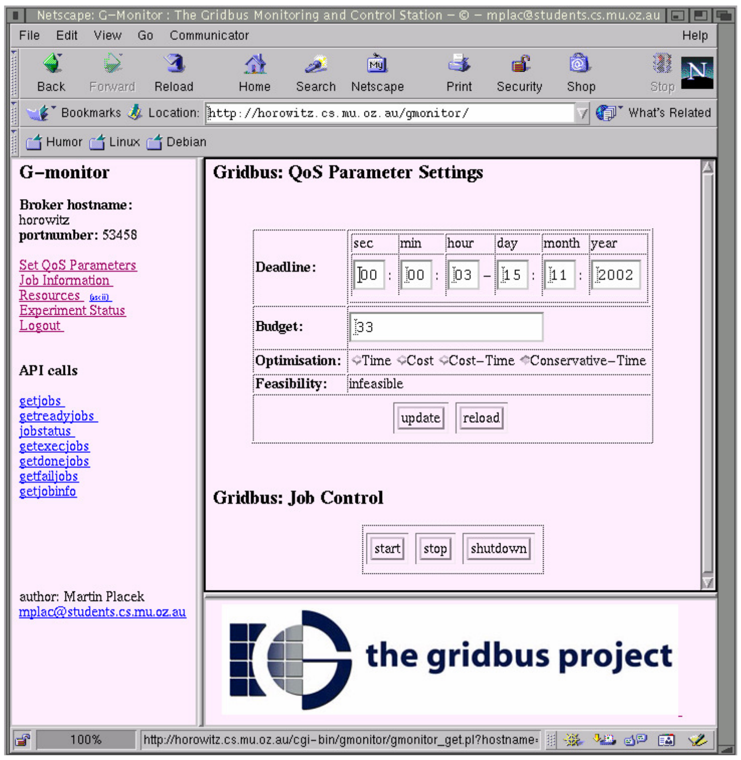
Broker Hostname: horowitz

Port Number: 53458

Logged in

Set QoS Parameters

By clicking on the first item we are presented with the page shown below. This page enables the user to set a Deadline, Budget and Optimisation, by changing the values and clicking on the "`update`" button. The second part of the page titled "`Gridbus: Job Control`" enables the user to "`start`", "`stop`" and "`shutdown`" the nimrod server.


Netscape: G–Monitor : The Gridbus Monitoring and Control Station – © – mplac@students.cs.mu.oz.au
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://horowitz.cs.mu.oz.au/gmonitor/ What's Related
Humor Linux Debian

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: QoS Parameter Settings

	sec	min	hour	day	month	year
Deadline:	00	00	03	15	11	2002

Budget:	33
Optimisation:	Time Cost Cost–Time Conservative–Time
Feasibility:	infeasible

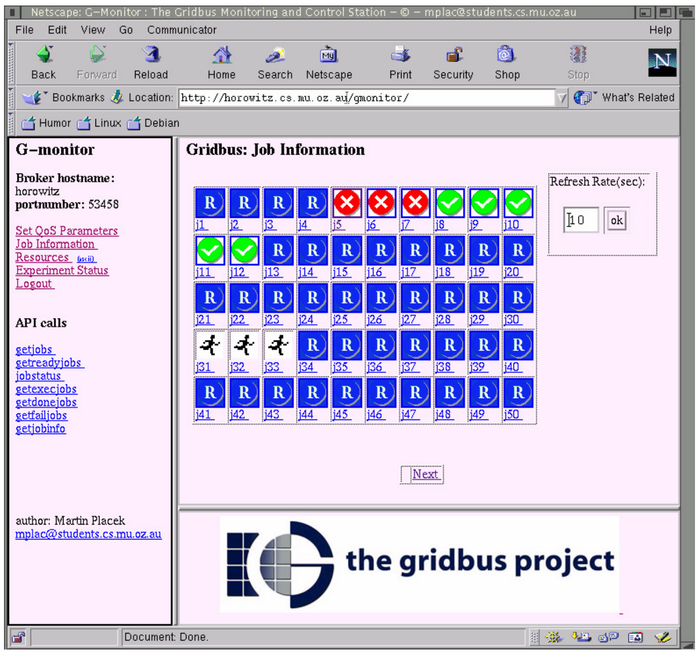
Gridbus: Job Control

start stop shutdown
the gridbus project
100% http://horowitz.cs.mu.oz.au/cgi-bin/gmonitor/gmonitor_get.pl?hostname=

QoS Parameters

Job Information

By clicking on the "`Job Information`" button, we are presented with the figure below, which is a summary of all the jobs currently available on the grid system. We can see that there are more jobs present and these can be viewed by pressing the next button. This page has a refresh rate, which is set to 10 seconds by default. The user has the option to change the refresh rate by entering another value and clicking the "`ok`" button.


Netscape: G–Monitor : The Gridbus Monitoring and Control Station – © – mplac@students.cs.mu.oz.au
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://horowitz.cs.mu.oz.au/gmonitor/ What's Related
Humor Linux Debian

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: Job Information

R j1	R j2	R j3	R j4	✗ j5	✗ j6	✗ j7	✓ j8	✓ j9	✓ j10
✓ j11	✓ j12	R j13	R j14	R j15	R j16	R j17	R j18	R j19	R j20
R j21	R j22	R j23	R j24	R j25	R j26	R j27	R j28	R j29	R j30
🏃 j31	🏃 j32	🏃 j33	R j34	R j35	R j36	R j37	R j38	R j39	R j40
R j41	R j42	R j43	R j44	R j45	R j46	R j47	R j48	R j49	R j50

Refresh Rate(sec): 10

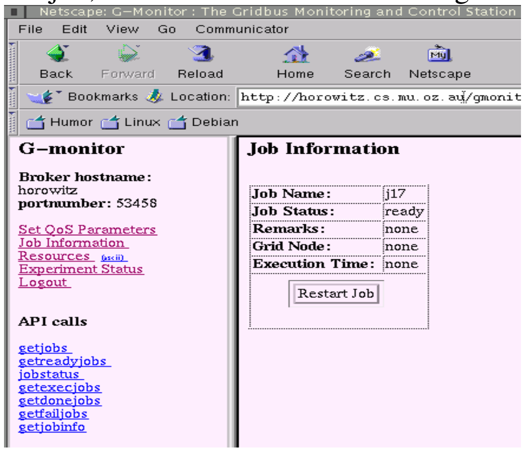
Job Information.

The icon associated with each job represents the job's status which can be one of "Ready, Running, Completed or failed".

Icon	Job Status
Ⓡ	ready
🏃	running
✅	completed
❌	failed

Icon - Job status association.

If you click on a particular job you will be presented with the job's attributes as show in (Diagram 6). The user is able to restart the job, this has the effect of re-setting its status to "ready".


Netscape: G–Monitor : The Gridbus Monitoring and Control Station
File Edit View Go Communicator
Back
Forward
Reload
Home
Search
Netscape
Bookmarks
Location: http://horowitz.cs.mu.oz.au/gmonit
Humor
Linux
Debian

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

Job Information

Job Name:	j17
Job Status:	ready
Remarks:	none
Grid Node:	none
Execution Time:	none

Restart Job

Job Attributes.

Resources

Clicking on the Resources link you will find that you are presented with a list of all the workstations registered on the grid.

Experiment Status

This page provides a summary of how the experiment is progressing. It provides the user with information on the spent and remaining Budget and Deadline, current status of all the registered jobs with the system and a list of all the hosts and their job assignment.

10.4 Feedback

For further details about G-Monitor please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)
Martin Placek (mplac@cs.mu.oz.au)

11 ExcelGrid

11.1 Introduction

ExcelGrid is a .Net plug-in to Microsoft Excel helps in extending Excel from the desktop to enterprise and global grids as to reap the benefits of improved job execution speed and getting processing done faster. It provides a front-end to a grid via Excel spreadsheet and performs user-defined computations on enterprise grids created using Alchemi and global grids built using Gridbus coupled with Globus, UNICORE and also Alchemi technologies. It allows users to run jobs on remote computers, using an easy-to-use GUI, and retrieve the results via the standard Excel spreadsheet interface. This section of the manual explains the way to use ExcelGrid to run jobs on remote grid nodes. It is intended to help users get started with ExcelGrid, by walking through the steps involved in running a parameter-sweep application on the grid.

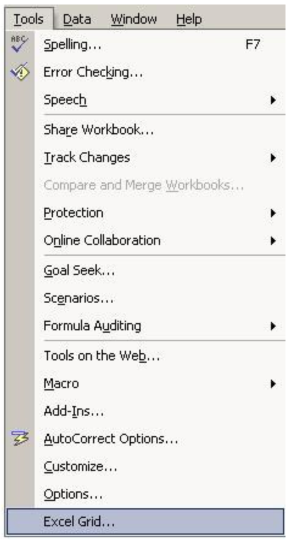
11.2 Prerequisites and Installation

Requirements:

- Microsoft .Net Framework v1.1
- Microsoft Excel XP (2002) or higher
- On the server side, an Alchemi Manager / Gridbus Broker installation is required. ExcelGrid works with Alchemi v0.8 and Gridbus broker v1.2

Installation:

To install ExcelGrid, run the setup.exe found in the "`setup`" directory of the distribution and follow the onscreen instructions. The setup program is a standard Windows Installer package, and no additional configuration is needed for ExcelGrid. After installation, ExcelGrid appears as a menu option in the Excel "`Tools`" menu, as shown in the figure below.

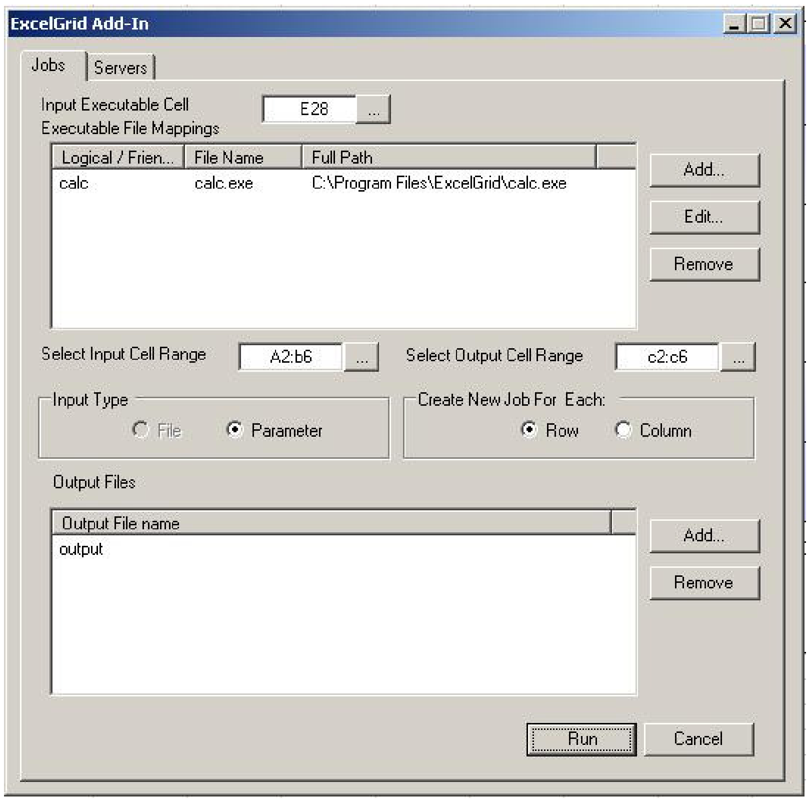


ExcelGrid in the Tools Menu

11.3 Using ExcelGrid

Using ExcelGrid with Alchemi v.0.8:

1. Start up Excel, and select "ExcelGrid" from the "tools" menu.
   The input form shown will appear, allowing for selection of various parameters that are using in the job composition.
2. Select the inputs to ExcelGrid.
   The inputs to ExcelGrid are:
   a. The address of the cell-range where input data is found
   b. The address of the cell-range where output data is to be placed
   c. The address of the cell where the name of the executable / job is found
   d. Whether the input parameters are all files or values (in this version, only value parameters are supported)
   e. Whether to create a new job for every row or one for every column of input
   f. The username, password and hostname, port of the middleware server running on the same or a remote machine
   g. Names of any output files produced (the output from the first file is placed in the output cells).



Input Executable Cell: E28

Executable File Mappings

Logical / Frien...	File Name	Full Path
calc	calc.exe	C:\Program Files\ExcelGrid\calc.exe

Select Input Cell Range: A2:b6

Select Output Cell Range: c2:c6

Input Type: File [✓ Parameter]

Create New Job For Each: [✓ Row] Column

Output Files

Output File name
output

Input Form

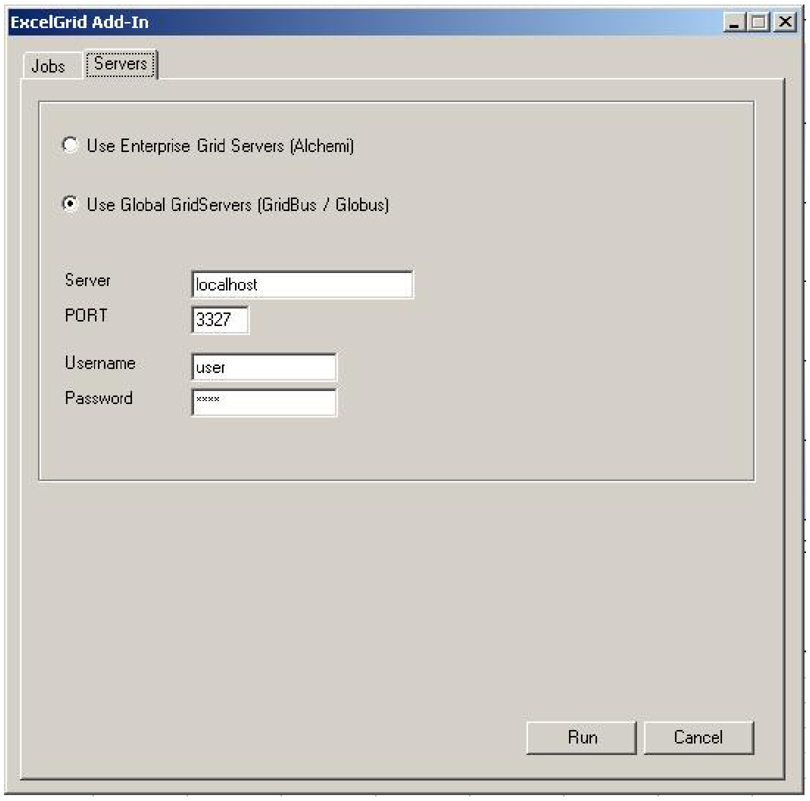
The form shown above is used to specify all the input parameters mentioned in step 3. An executable file mapping is just a friendly name for an executable file. So, an executable file located at say,

```
C:\Program Files\ExcelGrid\calc.exe
```

could be simply called "calc". And the excel spreadsheet would then simply contain "`calc`" as the name of the executable for the job, and ExcelGrid will use the mapping specified, to

locate the actual file. As show in the figure, ExcelGrid is pointed to look for the name of the executable, in the cell addressed: `E28`.
So, ExcelGrid looks for the name found in `E28`, in its list of executable file mappings, and resolves the actual physical executable to run on the remote node.

The input cell range is selected by the user, and ExcelGrid creates jobs from the input specified. For example, in the above form, the input cell range is specified as `a2:b6`. So, ExcelGrid reads the values found in the cells `a2:b6` which is an array of 5 rows and 2 columns.



( ) Use Enterprise Grid Servers (Alchemi)

(●) Use Global GridServers (GridBus / Globus)

Server	localhost
PORT	3327
Username	user
Password	****

Server selection

Based on the option select for “Create New Job for Each Row / Column”, ExcelGrid decides how to create grid jobs.

- if the “row” option is selected, then 5 jobs with 2 parameters each are created
- if the “column” option is selected, then 2 jobs with 5 parameters each are created

For Alchemi, the user should also specify one or more output files which the job produces, and ExcelGrid will read the first output file, to obtain the output from Alchemi, and place it in the address of the output cells specified, in the cell range `c2:c6`, in the above example. For the current version we assume that each job produces one output, which is placed in the addresses specified.

3. Select the middle-ware platform and specify security credentials. The figure above shows the tab where the grid-middleware supported by ExcelGrid is shown. One of the options is selected, and the server host: port, and username, password are specified in this form.
4. Once all the inputs are specified, click "Run" and ExcelGrid will then prepare the jobs, try to establish a connection to the middleware specified, and submit the jobs to run on the remote grid nodes.
5. As the jobs execute, the progress form shown in the figures below, displays the status of job execution. As the results come in, they are placed in the output cells.

Using ExcelGrid with Gridbus Broker v.1.2:

Running ExcelGrid with the Gridbus broker follows the same procedure as outlined above, with the following differences:

1. For Gridbus broker, the current version expects the executable to be present on the remote node and the broker will read the output and provide it to ExcelGrid. So, any output files specified are simply ignored.
2. We provide a broker plug-in which needs to be running on the server-side, to be able to receive requests from ExcelGrid. This is found in the "`broker`" directory of the distribution.

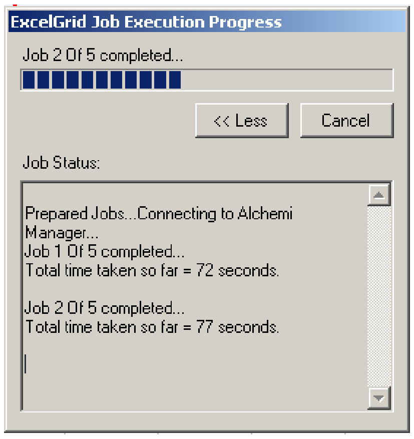
   To run the broker plug-in, set the classpath to the broker and start the plug-in (`ConnectServer.class`) like any other java program:

   For example:

   ```
   java ConnectServer
   ```

   This will then start the broker plug-in and listen for clients on a port which is output on the console. This port number should be used as the port number in ExcelGrid when running with the broker. Running the broker plug-in requires a full installation of the Gridbus broker in addition to the broker plug-in library (`gbconnector.jar`).

   For more details on what is needed to run the broker please refer to the Gridbus broker 1.2 documentation. (*http://www.gridbus.org/broker/)*
3. In the current version, no username/password credentials are needed to run the broker, and when running ExcelGrid with the broker middleware, any username/password entered are simply ignored.

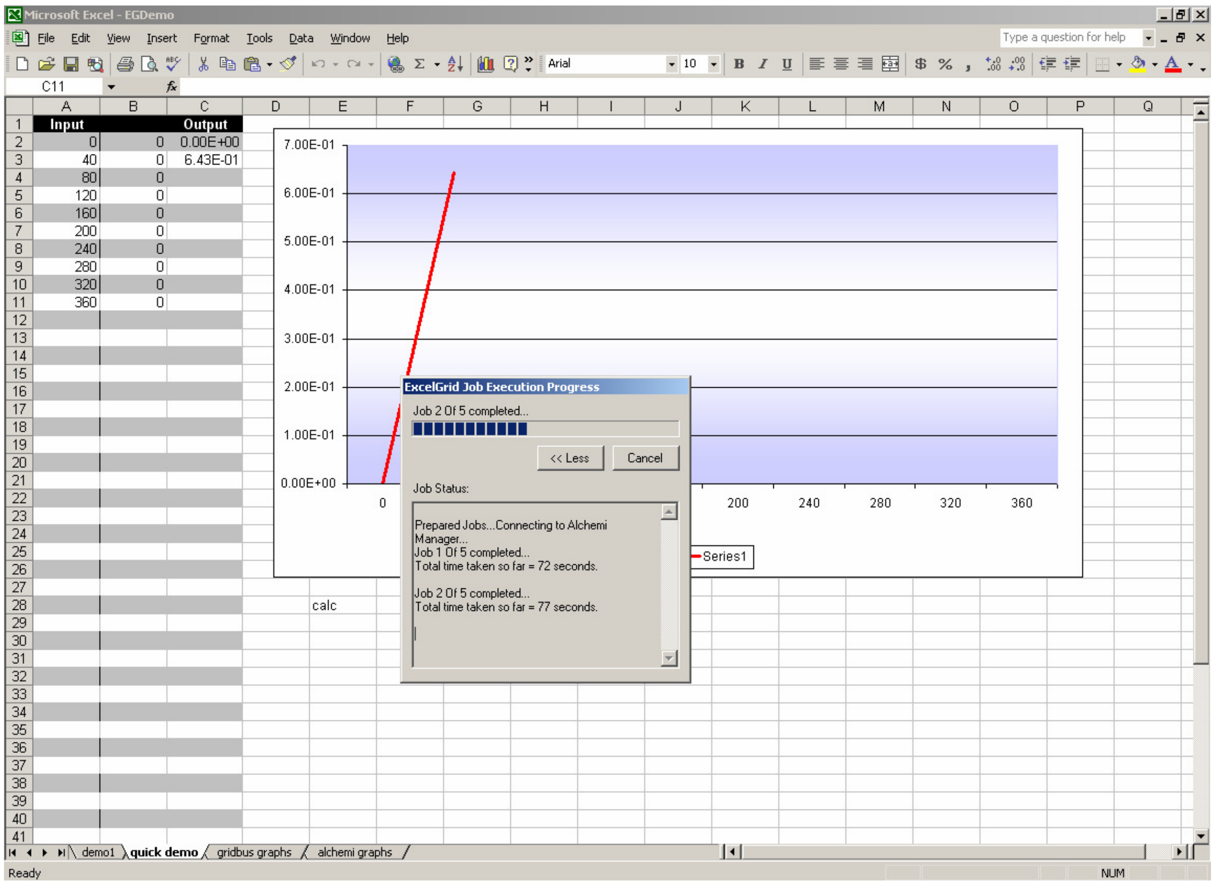


```
Job 2 Of 5 completed...

Prepared Jobs...Connecting to Alchemi
Manager...
Job 1 Of 5 completed...
Total time taken so far = 72 seconds.

Job 2 Of 5 completed...
Total time taken so far = 77 seconds.
```

Job Progress



Input		Output
0	0	0.00E+00
40	0	6.43E-01
80	0	
120	0	
160	0	
200	0	
240	0	
280	0	
320	0	
360	0	

Job 2 Of 5 completed...

Job Status:

```
Prepared Jobs...Connecting to Alchemi
Manager...
Job 1 Of 5 completed...
Total time taken so far = 72 seconds.

Job 2 Of 5 completed...
Total time taken so far = 77 seconds.
```

ExcelGrid gathering results

11.4 Feedback

We hope this manual has helped you to get started with ExcelGrid and experiment with running a few grid applications. We would be happy to provide help and support, and invite you to share your experiences, suggestions and opinions about using and extending ExcelGrid to make it more useful. For more information please contact:

Dr. Rajkumar Buyya (raj@cs.mu.oz.au)

Krishna Nadiminti (kna@unimelb.edu.au).