

Supporting Teaching of Grid Computing

Author: Jack Wootton

Supervisor: Gabor Terstyanszky

Date: October 18, 2006

Keywords: Grid Computing, Teaching, P-Grade, Gridsphere, Linux

Abstract: Teaching Grid computing on a real Grid introduces a risk of damage to Grid resources, caused by inexperienced users submitting executables to remote execute machines. This project presents a Grid teaching solution that provides a safe execution environment for Students to learn Grid computing, while ensuring knowledge is transferable from the teaching environment, to a live Grid.

This report is submitted in partial fulfillment of the requirements for the M.Sc. Internet Computing at the University of Westminster.

Acknowledgement

The author acknowledges the kind input and support of Szabolcs Illes, Gabor Terstyanszky and Thierry Delaitre.

Table of Contents

1.0 Introduction	1
2.0 Related research	3
2.1 What is Grid Computing?	3
2.2 Grid Architecture	3
2.3 Who uses Grid?	5
2.4 Condor and Condor G	5
2.5 Globus Resource Allocation Manager	6
2.6 Pre WS GRAM	7
2.7 WS GRAM	7
2.8 Grid enabled web portals	7
2.9 Collaboration and teaching	8
2.10 Virtual workspaces	9
2.10.1 Dynamic (Unix) Accounts	10
2.10.2 Virtualisation	10
2.10.3 Virtual Workspace deployment	11
2.10.4 Virtual clusters	12
3.0 Simulators	14
3.1 GridSim	14
4.0 Testbeds	15
4.1 GridLab Testbed	15
4.2 Grid Infn Laboratory for Dissemination Activities Testbed	15
5.0 Configuration of Grid architecture	16
6.0 Requirements Analysis	17
6.1 Expectations	17
6.2 Problem Domain	18
6.3 Requirements	20
6.3.1 Tutors Functional Requirements	20
6.3.2 Students Functional Requirements	21

6.3.3 Non Functional Requirements	21
6.3.4 Interface requirements	21
6.4 Use case analysis	23
7.0 Design	25
7.1 Installation of multiple head nodes on a single cluster	25
7.1.2 Advantages	26
7.1.3 Disadvantages	26
7.2 Running multiple gatekeepers on a single head node	28
7.2.1 Advantages	28
7.2.2 Disadvantages	29
7.3 Hostname mapping	29
7.3.1 Advantages	30
7.3.2 Disadvantages	31
7.4 Solution	31
7.5 Unimplemented and of considerable relevance	36
7.5.1 Scenarios	36
7.5.2 Sequence Diagrams	38
7.6 Implemented and of slight relevance	39
7.6.1 Download proxy certificate	39
7.6.2 Submit workflow	40
7.6.3 View workflow output	40
7.7 Implemented and of considerable relevance	40
7.7.1 Add Grid configuration	40
7.7.2 Add Grid resource URL	40
7.7.3 Add portal account.	40
8.0 Implementation	42
8.1 Resources	42
8.2 JSR Portlet	42
8.3 Database configuration	44
8.4 JSR Portlet View	45

8.5 Solution Architecture	47
8.5.1 Adding a virtual resource	47
8.5.2 Using virtual resources	54
8.5.3 Remove virtual resource	55
8.6 Solution configuration	56
9.0 Testing	59
10.0 Conclusion	63
11.0 References	65
Appendix A	68

Index of Figures

Figure 2.1 The relationship between Condor, Condor G, GRAM and Grid clients.	6
Figure 2.2 XML schema describing a workspace [22].	11
Figure 2.3 Relationship between a Grid job, workspace metadata schema, Grid resources and WMS implementation methods.	12
Figure 2.4 Graphical representation of the workspace metadata schema, extended to describe virtual clusters [24].	13
Figure 3.1 Terminal output after running Example01 on GridSim, where ‘PE’ is a computer processor.	14
Figure 6.1 Grid teaching problem domain.	19
Figure 6.2 Project of Grid teaching problem domain with solution.	20
Figure 6.3 Configuration portlet user interface.	22
Figure 6.4 Use case diagram outlining requirements of Grid teaching solution.	23
Figure 7.1 The relationship between Condor, Condor G, GRAM and Grid clients, applied to multiple submit machine.	27
Figure 7.2 The relationship between Condor, Condor G, GRAM and Grid clients, applied to multiple gatekeepers.	29
Figure 7.3 The relationship between Condor, Condor G, GRAM and Grid clients, applied to a hosts file lookup.	31
Figure 7.4 Sequence diagram detailing logical steps for use case ‘Adding Virtual Resource’.	38
Figure 7.5 Sequence diagram detailing logical steps for use case ‘Removing Virtual Resource’.	39
Figure 7.6 Class diagram of solution: hostname mapping configuration.	41
Figure 8.1 Screenshot of Initial portlet view.	47
Figure 8.2 Portlet view to add new Grid Configurations.	48
Figure 8.3 Portlet view to add new Grid resource URLs to a Grid configuration.	48

Figure 8.4 Default view of Virtual Resource portlet with Grid configurations.	49
Figure 8.5 Default view of Virtual Resource portlet with Grid configurations and Grid resources.	50
Figure 8.6 Screenshot of Virtual Resources Portlet: Newly created virtual resource.	53
Figure 8.7 P-Grade workflow with four nodes.	54
Figure 8.8 Selecting Grid configuration and resource URL for a node.	55
Figure 8.9 Screenshot of portlet view after the removal of a virtual Grid resource.	56
Figure 8.10 Grid teaching solution deployment diagram.	58

Index of Tables

Table 6.1 Problem elements of the Grid teaching problem domain.	18
Table 7.1 Port, IP and Name values applied to the multiple submit machine candidate solution.	25
Table 7.2 Port, IP and Name values applied to the multiple gatekeeper candidate solution.	28
Table 7.3 Port, IP and Name values applied to the hosts file candidate solution.	30
Table 7.4 Summary of the main characteristics dictating the quality of each candidate solution.	32
Table 8.1 Skelton Gridsphere project structure generated using Ant.	43
Table 8.2 Virtual Resource Portlet UI Bean action methods.	45

1.0 Introduction

Grid computing involves the coordination of multiple distributed processes such that they function as one system. Computational Grids are used to achieve this. Computational Grids are inherently complex, and introducing Grid computing to new users is a non trivial task. Grid computing operates through the use of remote machines accepting executable code, which they process. Grid computing defines far more than the organisation and architecture of computational networks. How Grid is used, the policies under which it may be used, access control and the even distribution of computational resources are all within the remit of Grid Computing. The multifaceted nature of Grid ensures that it remains a hotbed of research. As the technologies underpinning Grid computing mature, so Grid becomes more usable and the applications of Grid broaden; an example being the relatively new commercial interest in Grid computing [18].

Increasingly Grid computing is finding itself in mainstream undergraduate computer science education [32]. It is undoubtedly a requirement of any technology that it be introduced to new users so that adoption of that technology is maintained and research efforts continue. The distributed nature of Grid computing, however, introduces a risk that inexperienced users of Grid may damage the shared, physical Grid resources. It is this risk that forms the context of this project. The purpose of this research project is to elicit a solution that counters the risk associated with inexperienced users of Grid submitting executable code to remote machine for processing.

Section 2.0 introduces Grid computing and the technologies that are utilised in making Grid computing possible. The organisation of Grid users, as well as past and present Grid architectures is also discussed. Section 2.4 introduces Grid software and toolkits that help make Grid computing more productive and automate common Grid computing tasks, such as proxy generation and remote job submission.

Grid computing has been implemented with two architectures, resource oriented and service oriented. These are both discussed in section 2.2. The role of the web portals in Grid computing is discussed in section 2.8.

Collaboration is a central pillar of Grid computing and is one of the foremost distinguishing features between High Performance Computing, and Grid Computing. There are substantial and varied problems involved in Grid computing, these are outlined, along with the relationship between collaboration and teaching in section 2.9. Various

toolkits and software solutions exist that either present themselves as a direct solution to the problem of teaching Grid, or can be adapted to form a solution. These are evaluated in section 7.0.

The requirements that must be fulfilled in solving the Grid teaching problem are introduced in section 6.0 followed by a detailed description of the problem domain. Implementation of the researched solution is detailed in full, in section 8.0. A full code listing is available in appendices A.

2.0 Related research

This chapter introduces Grid computing and the areas of Grid computing that present themselves as possible solutions to the problem of teaching Grid computing.

2.1 What is Grid computing?

The term ‘Grid computing’ refers to the use of decentralized computing resources to solve complex scientific or engineering problems. [14] surmise that Grid Computing is a concept that allows for coordinated resource sharing and problem solving in dynamic, multi-institutional groups. Increasingly such multi-institutional groups are commercial and no longer restricted to academic or scientific domains [6]. The dynamic nature of Grid computing introduces incompleteness into definitions that were once sufficient. Dynamic in the context of Grid computing refers to both the diverse and rapidly changing compute resources available, as well as the current and potential users of Grid. Nevertheless [14] provide a sufficient definition for us.

Grids are themselves not a single technology, but rather an amalgamation of existing, proven technologies including, but not limited to Web Services, FTP, and certificates. Such an infrastructure affords Grid Computing and associated Grids scalability. Resilience is a key feature of Grid computing and is provided by monitoring services and brokers, while computational performance (including storage capacity) is provided by the union of computing resources to form a ‘Grid’.

2.2 Grid architecture

Grid architecture should allow for coordinated resource sharing and problem solving between the users [14]. The aforementioned assertion is often referred to as ‘The Grid Problem’. The current (service orientated) architecture of Grid allows for interoperability through common protocols and collaboration through Virtual Organisations, with resources exposed as consumable services. To better understand Grid as it is now, it is advantageous to look at the history of Grid architecture, gaining an appreciation that Grid architecture has evolved to reach its current implementation, and will undoubtedly continue evolving though the wider adoption of Grid and the maturation of Grid related technologies.

The words ‘collaboration’ and ‘sharing’ did not always feature so commonly in Grid computing literature, reflecting architectures of the time. In 1998 [16] defined Grid as “a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities”. Clearly emphasis was placed on the role of resources in Grid rather than collaboration and resource sharing.

Grid architectures fall into one of two categories:

1. Resource oriented.
2. Service oriented.

The definition offered in 1998 characterises resource orientated grid, whereas the definition offered at the start of this document refers to service orientated grid. The evolution of Grid architecture is clearly documented through the comparison of [16] and [17], where the former discusses architecture in terms of clusters, intranets, internets and the latter in terms of services.

Service Oriented Architecture (SOA) is a paradigm for making computational resources available to consumers. Consumers can be realised in many ways: other services, humans and programs can all be service consumers. The adoption of SOA by Grid has resulted in the Open Grid Services Architecture (OGSA). OGSA is a layered architecture, with each layer representing some functionality with the aim that higher level services and applications use lower level layers to become part of a grid and therefore share resources [19].

SOA consist of three primary actors:

1. Service provider – the business logic of the service being offered.
2. Service registry – a method to locate services based on criteria provided by the potential service consumer.
3. Service consumer – user of the service being offered.

Web services realise SOA through *stateless* services, whereas Grid computing realises SOA through *stateful* services implemented using the Web Service Resource Framework (WS-RF) [9]. The WS-RF is supplied as part of the Globus Toolkit 4 (GT4). The latest stable release from Globus is version 4.0.3 [11]. A grid service at it’s most atomic is simply be a web services implementing the WS-RF.

2.3 Who uses Grid?

Grids are used by Virtual Organisations (VOs). VOs can be individual users and institutions that collaborate in problem solving, linked by common resource sharing rules [14]. Common resource sharing rules are distinct from the idea of a formal contract, as such VOs may increase and decrease in size allowing individual members to come and go. They may also disband altogether once their objectives are completed. Definitions of VOs are deliberately fuzzy since they may be formed for a variety of reasons, vary in size and longevity and require diverse resources. Members of VOs are often geographically distinct from each other, for example, students from different universities may join to form a VO [4].

2.4 Condor and Condor G

Condor [5] is software that organises and keeps track of users' Grid jobs as they are submitted to Grid resources. Condor provides users with the functionality to:

- View the queue of submitted jobs, also known as the 'submit queue'.
- Submit new jobs.
- Record previously submitted jobs.
- View information about completed jobs.

The aforementioned capabilities are provided by Condor job management software, known as Condor G. As well as *job* management, Condor provides *resource* management functionality, keeping track of:

- Resources available to run Grid jobs.
- The organisation of resources such that are used efficiently for the potentially long submit queue.
- Resources unavailable to run Grid jobs.

'Submit machines' is the name given to machines with Condor job management installed. 'Execute machines' is the name given to machines with Condor G (resource management) installed. A single machine can run both Condor and Condor G, however it is common to follow the architecture in figure 2.1. Condor G is so called since it uses a Globus developed protocol to start the Grid job on the execute machine, which may be remote to the submit machine. Condor G uses the Globus toolkit to offer:

- Authentication – mutual authentication method ensuring that users are permitted to use real Grid resources and that the resources themselves can be trusted.
- Program execution – execution of the Grid job sent by a Grid user.
- Remote program execution - execution of the Grid job sent by a Grid user on a remote compute machine.

- Data transfer – staging of any data the Grid job may require during execution as well as the executable itself. Within the context of virtualisation in Grid, whole virtual machine images may be staged onto a physical resource running a hypervisor.

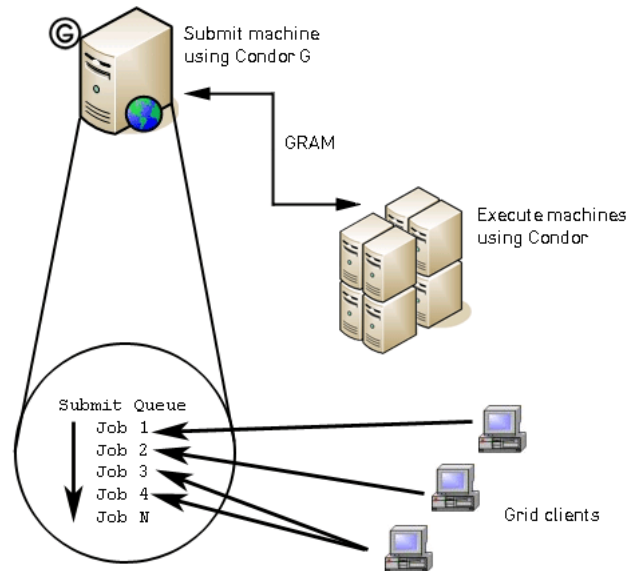


Figure 2.1: The relationship between Condor, Condor G, GRAM and Grid clients.

2.5 Globus Resource Allocation Manager

Distributed computing relies on remote job submission and control of jobs running remotely. The Globus Resource Allocation Manager (GRAM) is a protocol belonging to the Globus toolkit that provides a standard interface for requesting and utilizing remote system resources. GRAM therefore simplifies the use of remote systems in Grid computing. GRAM itself is not job scheduler, but is instead used to interface job schedulers; it is this interface that has been standardized through the use of GRAM. Without GRAM, developers would be required to deal with a large number of protocols and interfaces.

GRAM has been split into two flavors:

1. Pre Web Service (WS) GRAM
2. WS GRAM

The use of Pre WS GRAM has been deprecated by Globus as a result of the integration of SOA into Grid computing. However the Globus Toolkit 4 (GT4) comes packaged with both flavors for backwards compatibility.

2.6 Pre WS GRAM

A gatekeeper is the first point of contact for a grid job (submitted by a grid user). It is the gatekeeper that handles the request from the client as well as creating a job manager for the grid job. It then becomes the role of the job manager to monitor the grid job as it runs remotely on an execute machine. When the job completes it is normal for the job manager to terminate. For the purposes of this project, it is enough to know that a GRAM Gatekeeper acts as a server, listening for job submissions on ports specified during installation. It is worth noting that it is the responsibility of the job manager, not the gatekeeper to communicate any job state changes back to the user. A submit machine may have multiple gatekeepers running, although they must all run as `root`.

Authentication is another job handled by the gatekeeper using the Grid Security Infrastructure (GSI) for mutual authentication. However the discussion of GSI is outside the scope of this project. Once authentication procedures complete, the gatekeeper will map a Grid user to a local user account, allowing the executable to be run securely and with correct permissions.

2.7 WS GRAM

WS GRAM exposes standardised job execution interfaces as a set of Grid services. Gatekeepers and Job Managers are replaced by their service counterparts:

- Managed Job Factory Service (MJFS)
- Managed Job Service (MJS)
- File Stream Factory Service (FSFS)
- File Stream Service (FSS)

The services previously listed work in combination to provide an interface for job submission, monitoring and control.

2.8 Grid enabled web portals

Enabling Grid for non Grid users is a challenging task. The entry barrier for access to Grid computing is high, caused by the complexities of distributed computing and the middleware that supports it. Grid enabled web portals have proven a successful method for lowering this barrier [25]. Portals interface a broad spectrum of Grid related functions including, but not limited to: data management, job submission, information services, application interfaces, collaboration, workflows and workflow visualizations. However the true spectrum of grid functionality that can be covered with the application of a Portal is infinite, since through portal development any functionality can be exposed and added to a Portal.

Portals not only expose existing Grid functionality in a user friendly manner, but serve to add new functionality that further reduces the entry barrier to Grid computing for non Grid users. For example, users of a portal may be presented with customised 'views' of the portal, reflecting their requirements. In this example such functionality is provided by portal persistence management combined with a portal login process.

While Grid enabled portals have received much attention from developers, solutions have suffered from being tightly coupled with the problem domain to which they were applied. Consequently the usefulness of portals as reusable code elements was limited [20]. In reaction to this problem, frameworks [13], [34] have been developed to promote a modular and standardised approach to exposing Grid functionality.

Finally, on this subject, it is worth noting that [27] nominate the relatively low cost of delivering functionality as an important factor behind the adoption of portals by the Grid community, simultaneously reminding us that providing a single point of access to multiple services and technologies is nothing new. Examples such as Yahoo [35] and MSN [28] are cited.

2.9 Collaboration & teaching

Grid resources ensure that people have access to information at the right time, improving decision making and collaboration [6]. Collaboration in Grid computing is evidently manifested through the sharing of resources; however it also manifests itself through the sharing of knowledge. Academic environments contribute to knowledge transfer, and it is essential that Grid computing finds itself in mainstream higher education to cement its use in a variety of contexts.

Learning distributed computing is a challenge due to difficulties in orchestrating meaningful demonstrations to students [33]. Difficulties in teaching Grid computing are exasperated by:

1. Inherent complexity of Grid computing - complex subject areas are naturally more challenging to teach.
2. Security requirements relating to membership and job execution - certificate authorization and Grid membership is often a complex and lengthy process.
3. Large number of resources required to teach distributed computing - it is impractical to create enough Grid resources purely for student use.
4. Large number of potential distractions to core learning.
5. Use of a real Grid (e.g. National Grid Service) is unsafe - inexperienced users may unintentionally overload Grid resources or cause security breaches. Examples of inexperienced users are students and any new user of Grid.

Any useful teaching environment should engage with problems 1-5. In summary a Grid teaching environment should:

- Provide isolation from real Grid resources.
- Simplify non trivial tasks.
- Simulate as closely as possible a real Grid.

Candidates include:

- Virtual Workspaces.
- Simulators.
- Grid testbeds.
- Configuration of Grid architecture.

We now discuss each potential solution area.

2.10 Virtual workspaces

Grid computing involves the sharing of heterogeneous Grid resources between VOs, each with potentially conflicting resource requirements. Such shared use often results in under-utilisation of Grid resources caused by a mismatch between resources offered and the application requirements [15] of the shared users. However sharing is not the only cause of resource under utilisation. [34] draw attention to the potential incompatibility of a cluster's installed libraries across VOs as another difficulty with current Grid use. Virtual Workspaces (VW) are an attempt to address the following three problems with Grid clusters:

- Lack of performance isolation.
- Little control over resource sharing.
- Fine grained usage difficult to enforce.

VWs act as an abstraction layer from physical Grid resources and allow execution environments to be dynamically created by VOs such that the execution environment will suit, exactly, the specific needs of each VO. A Grid client may then define a VW driven by their own specific requirements [22]. Importantly distribution of Grid resources between multiple VOs is maintained.

The Globus Alliance [10] developed the Workspace Management Service (WMS) which provides an interface to common VW functionality. Since it is a service developed for use with Grid computing, naturally it complies with the GT4 implementation of the Web Service Resource Framework (WSRF) model. Through the WMS VOs can create, manipulate and destroy workspaces.

Virtual Workspaces result in Grid ‘jobs’ (executables to be run on Grid resources) being mapped to a workspace rather than a resource. Providing the job maps successfully to a workspace and the workspace resource description can be honored by a resource, the job will run.

VWs can be realised in two ways, firstly through Dynamic Accounts, secondly using virtualisation technologies.

2.10.1 Dynamic (Unix) Accounts

Implementing the WMS through Dynamic Accounts (DA) works either by adding user accounts on a Unix system (using the Unix `useradd` command), or requesting an account from a pool of accounts. Independent of the method chosen, the objective is to map a Grid identity onto a local Unix account. Once the account is in use, the Unix operating system provides:

- Control of resource allocation through the use of Unix commands: `setrlimit`, `quota` and `chroot`.
- Control over account lifetime.
- Auditing.
- Isolation through user groups and user accounts.

The ‘backend’ of the WMS implementation uses Unix commands, accounts, groups and tools to create or load a workspace configuration, as well as customise it once it is loaded.

2.10.2 Virtualisation

Dynamic Accounts provide a relatively coarse grained implementation of VW and do little to provide a custom execution environment for the Grid client. Virtual Machines (VM) can be used to implement VWs through the WMS by installing Grid resources with a Virtual Machine Monitor (VMM). Images containing representations of RAM, disk and other resource attributes can be loaded by the VMM. The custom (Grid) execution environment can then be used. VMs provide excellent isolation and offer fine grained control over resource customisation, as well as providing the Grid client with the perception of using a single isolated machine. One of the most appealing attributes of VMs is the ability to serialize the state of the VM, such that it may be reloaded on a different physical resource and resume processing. [37].

Significant barriers remain in the adoption of virtualisation technologies in Grid, namely staging of the VM image and any data and resources required during job execution. However, assuming staging of resources has been completed, use of VMs cause only small performance degradation (approximately 5%) [22].

2.10.3 Virtual Workspace deployment

It has been described how the WMS can be implemented using either DAs or Virtualisation. But how does a Grid client describe its resource requirements such that the execution environment may be created to realise the resource requirements? A number of methods have been experimented [23], [34], [15], [1], [24].

An accepted [22], [24] method is to create an XML schema that describes the workspace, how it should be deployed and the quota of resources required. Workspace description schema are called *workspace metadata schema*.

```

<xs:simpleType name="categoryType">
<xs:restriction base="xs:string">
<xs:enumeration value="dynamic account"/>
<xs:enumeration value="vm"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="stateType">
<xs:restriction base="xs:string">
<xs:enumeration value="shutdown"/>
<xs:enumeration value="paused"/>
<xs:enumeration value="running"/>
<xs:enumeration value="corrupted"/>
</xs:restriction>
</xs:simpleType>
<xs:element name="virtualWorkspace">
<xs:complexType>
<xs:sequence>
<xs:element name="category" type="categoryType" default="vm"/>
<xs:element name="state" type="stateType" default="shutdown"/>
<xs:element name="EPR" type="wsa:EndpointReferenceType"/>
<xs:element name="creationTime" type="xs:dateTime" minOccurs="0"/>
<xs:element name="lifeCycle" type="xs:integer" minOccurs="0"/>
<xs:element name="lastModified" type="xs:dateTime" minOccurs="0"/>
<xs:element ref="hw:hardware" minOccurs="0"/>
<xs:element ref="net:network" minOccurs="0"/>
<xs:element ref="sw:software" minOccurs="0"/>
<xs:element ref="cap:capability" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>

```

Figure 2.2: XML schema describing a workspace [22].

A VM image that fulfills the workspace described by the workspace metadata schema (above) is retrieved from a pool of suitable VM images, and staged onto the resource. Alternatively an existing VM image is cloned; the new image is then staged onto the physical resource.

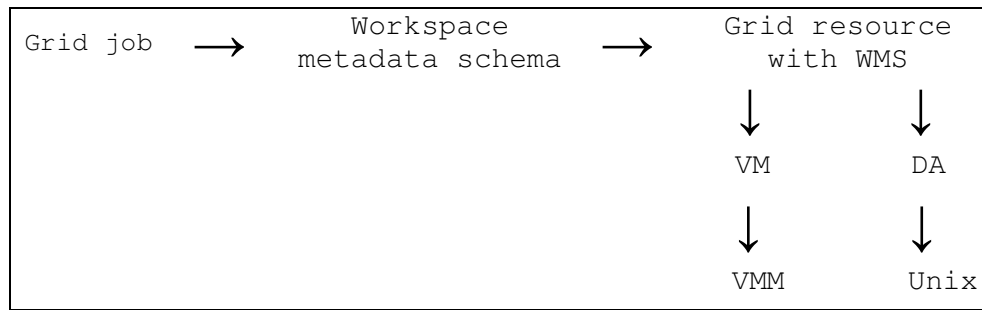


Figure 2.3: Relationship between a Grid job, workspace metadata schema, Grid resources and WMS implementation methods.

2.10.4 Virtual clusters

So far we have discussed only atomic workspaces implemented using VMs or DAs mapped onto a single Grid resource. [15] propose extensions to the workspace metadata schema and changes to the WMS, such that virtualisation of entire clusters may be achieved. Virtual clusters work by means of:

- Collections of atomic workspaces.
- A method of describing collections of atomic workspaces.
- A method of managing collections of atomic workspaces.

The workspace metadata schema is extended to include aggregations of atomic workspaces. Workspaces now become *aggregate workspaces*. An aggregate workspace contains sets of workspaces; each set contains homogeneous workspace configurations. It is through the combination of sets within an aggregate workspace, that a heterogeneous cluster may be described using a single workspace metadata schema. Importantly the ability of aggregate workspaces to define heterogeneous clusters allows both service and worker nodes to be described.

In order to manage aggregate workspaces, the WMS is extended to handle local IP address assignment to each node. In addition, the service is extended to utilise a database which records the availability and state of physical resources that may be used to deploy a workspace. The database may then be queried so that suitable physical resources are selected for the deployment of the workspace.

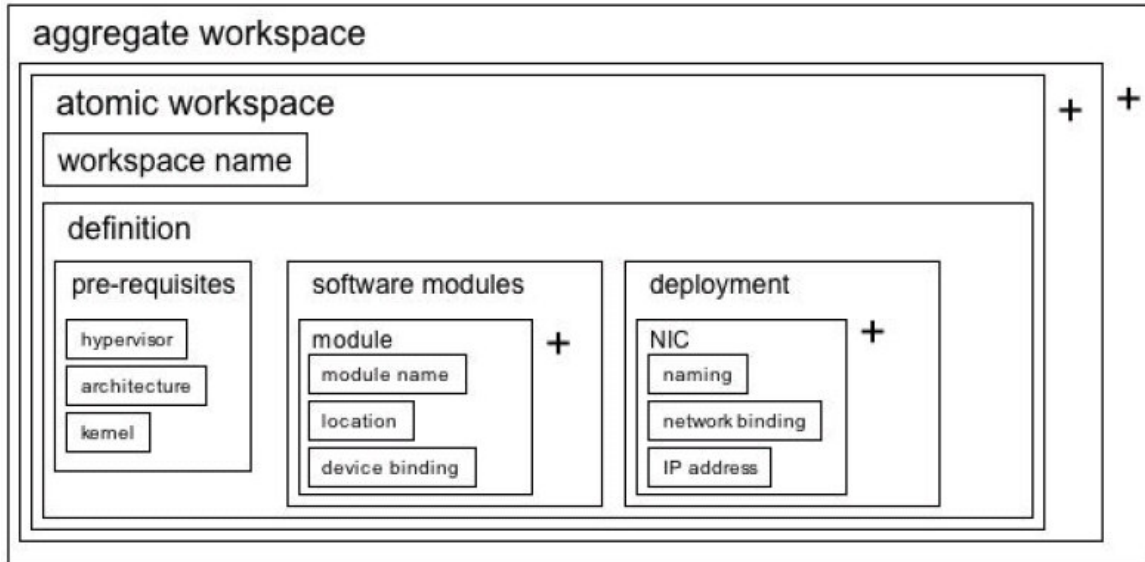


Figure 2.4: Graphical representation of the workspace metadata schema, extended to describe virtual clusters [24].

3.0 Simulators

3.1 GridSim

The GridSim toolkit simulates heterogeneous resources, including single and multiprocessors, shared and distributed memory and clusters with distinct configurations [32]. Evaluation of algorithms, specifically scheduling algorithms on Grid clusters can be performed using GridSim. A number of examples are packaged with GridSim, and it is these examples that provide a potential Grid teaching solution.

```
Starting example of how to create one Grid resource
Initializing GridSim package
Initialising...
Starting to create one Grid resource with 3 Machines ...
Creates a Machine list
Creates a PE list for the 1st Machine
Creates 4 PEs with same MIPS Rating and put them into the PE list
Creates the 1st Machine that has 4 PEs and stores it into the Machine list

Creates a PE list for the 2nd Machine
Creates 4 PEs with same MIPS Rating and put them into the PE list
Creates the 2nd Machine that has 4 PEs and stores it into the Machine list

Creates a PE list for the 3rd Machine
Creates 2 PEs with same MIPS Rating and put them into the PE list
Creates the 3rd Machine that has 2 PEs and stores it into the Machine list

Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid
resource
Finish the 1st example
```

Figure 3.1: Terminal output after running Example01 on GridSim, where 'PE' is a computer processor [GridSim].

The GridSim toolkit could be extended, with little effort, to prompt for user input when creating resources and clusters, thus allowing students to create simulated Grid resources. From this perspective teaching the idea of multiple distributed resources being utilised for a Grid job could be articulated through GridSim. Advanced students could also extend their learning by programming with the GridSim toolkit; however, it is likely that the students programming would improve at the expense of their understanding of Grid Computing.

4.0 Testbeds

4.1 GridLab Testbed

The GridLab Testbed [12] provides an environment for development, testing, evaluation, and deployment of new components for use in a Grid environment. GridLab's Testbed could be used as a Grid teaching environment. However GridLab Testbed is designed as a test environment for real Grids, and therefore is highly similar to a real Grid and consequently fails to solve many of the problems a real Grid presents.

For the following reasons GridLab Testbed is not suitable for a teaching environment:

- New user needs a digital certificate signed by a trusted Certificate Authority (CA).
- Certificates signed by Globus CA are not valid.
- Credentials must be placed in students `~/.globus` directory, therefore requiring a Unix account for each student.
- Each student must be listed in the Grid-map file, adding to administration tasks.
- GSISSH client must be installed. This could be overcome by using GSISSH client available on the NGS website [30]

4.2 Grid Infn Laboratory for Dissemination Activities Testbed

The Grid Infn Laboratory for Dissemination Activities (GILDA) Testbed was released for use in 2004. One of the primary objectives was to serve as a method for knowledge transfer [8], so that training sessions for new Grid users may be built around the GILDA testbed.

GILDA is comprised of:

- GILDA Testbed – a series of Grid resources focused in Italy including resource brokers, data managers, monitoring tools, compute machines and storage machines.
- Grid Demonstrator and Grid Tutor – a web portal based on the GENIUS web portal [7] allowing job submission from a predefined list of Grid.
- GILDA certification authority – a Certificate Authority (CA) issuing certificates that may be used on the GILDA testbed, excluding real Grids.
- GILDA VO – a virtual organisation, membership of which must be explicitly requested, but open to any member of GILDA.
- Grid monitoring system.
- GILDA mailing list (`gilda@inf.it`).

The GILDA testbed provides an entire grid dedicated to new users and makes for a seemingly highly suitable teaching environment. Furthermore tools such as the Grid Demonstrator and Grid Tutor provide pre-written programs that serve to reduce potential risk of damage to compute machines as well as ensuring new users is not distracted by trivial tasks such as writing executables to run on Grid resources.

5.0 Configuration of Grid architecture

Through the combination of existing Grid technologies and architectures a teaching environment may be created, the operation of which appears similar to a real Grid. There are three possible configurations of Grid architecture that may lead to a solution. Each of the three configurations will be applied to the architecture in figure 2.1 in determining the appropriateness of the potential solution. The three configurations are:

- C1. The installation of multiple head nodes on a single cluster.
- C2. The installation of multiple gatekeepers on a single head node.
- C3. The use of (existing) operating system level tools that assist in the development of a Grid teaching environment.

It is outside the scope of this chapter to extract a solution from the aforementioned configurations. Section 7.0 attempts to clarify how configurations C1, C2 and C3 can provide a solution.

6.0 Requirements analysis

This chapter outlines the requirements that any candidate solution should fulfill in providing a successful Grid teaching solution.

6.1 Expectations

There exist a number of potential solutions to the problem of teaching Grid. As we have seen solutions may be built around:

- Virtual Workspaces, section 2.10.
- Simulators, section 3.0.
- Grid testbeds, section 4.0.
- Configuration of Grid architectures, section 5.0.

However the aforementioned solution domains all suffer from either one or both of the following two problems:

1. They are inaccessible to small or resource limited institutions.
2. They are overly complex and not suitable for undergraduate students.

With the exception of GILDA, there exists no Grid teaching solution that makes use of well established components that are either freely accessible or come pre installed on common computer systems. A successful Grid teaching environment should be financially viable and have minimal physical resource requirements. It is also important however that the teaching environment simulates a real Grid as closely as possible. It is the mismatch between these two requirements that define the boundaries of the problem domain.

The teaching environment should present an easy transition to real Grid, meaning knowledge and skills learnt by students in the teaching environment should be easily transferable to a real Grid environment.

Any solution, where possible, should provide flexibility in the look and feel of the teaching interface. The implication being that a Grid teaching solution should be independent of the interface used by students and staff.

6.2 Problem domain

Name	Description
Virtual Grid	An environment that mimics a real Grid such that Grid jobs are run and results may be retrieved while there is no risk of damage to real Grid.
Virtual Resource	A Grid resource URI that is mapped to a single physical resource.
Real Grid	A resource that forms part of a Real Grid.
Real Resource	A Grid resource URI that represents a real physical resource that is unique to the URI.
Physical resource	A physical component of a Grid resource, including compute machines and storage machines.
Configuration Portlet	A portlet providing a friendly interface for on demand resource creation and destruction.
Tutor	The user of the Configuration Portlet.
New Grid users (Student)	The user of the Simulation Grid.
Grid enabled web portal	A portal that exposes Grid functionality in a user friendly manner.
Real Grid	An existing fully functional Grid, for example, the National Grid Service (NGS) [29].

Table 6.1: Problem elements of the Grid teaching problem domain.

Figure 6.1 is a diagrammatic representation of the Grid teaching problem, showing the relationship between Grid users, Grid enabled web portal, real Grid and physical grid resources within the problem domain. This is a research project, which aims to find a solution to the problem of teaching Grid.

‘New Grid users become a type of ‘Existing Grid users as soon as they are in possession of a valid Grid proxy certificate, and as such have access to the same Grid resources experienced users do. ‘New Grid users also have access to the same grid enabled web portal as experienced users. A solution should not seek to change this last assertion,

since the use of a Grid enabled web portal is an integral part of learning Grid. A trend that is likely to strengthen as the development of grid enabled web portals matures.

The use of a Grid enabled web portal is both part of the problem and the solution. It is part of the problem since it allows Grid resources to be shared between many different types of user. Moreover, it lowers the entry barrier to Grid, allowing increasingly inexperienced users to run Grid jobs. It is, however, part of the solution, since any teaching solution should ease the transfer of knowledge from a learning environment, to a live environment. Grid enabled web portals satisfy this requirement excellently.

In finding a solution to the problem of teaching Grid it is anticipated that other, secondary problems such as the non-trivial authentication process required to gain a valid Grid certificate, will also be addressed. A successful solution should deal with the problem of risk to underlying physical Grid resources. It is projected that a suitable solution to the Grid teaching problem would modify the problem domain, such that it is suitable for new Grid users, shown in figure 6.2.

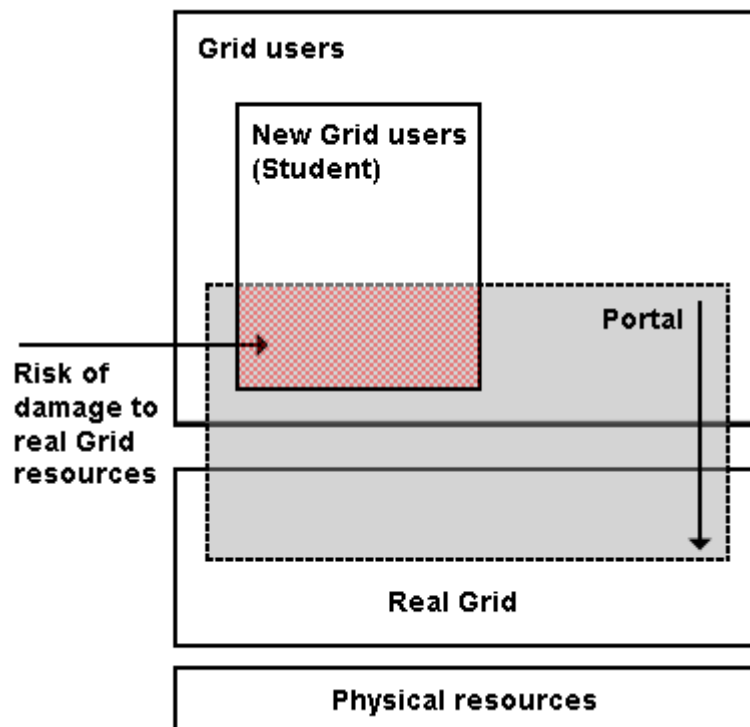


Figure 6.1: Grid teaching problem domain.

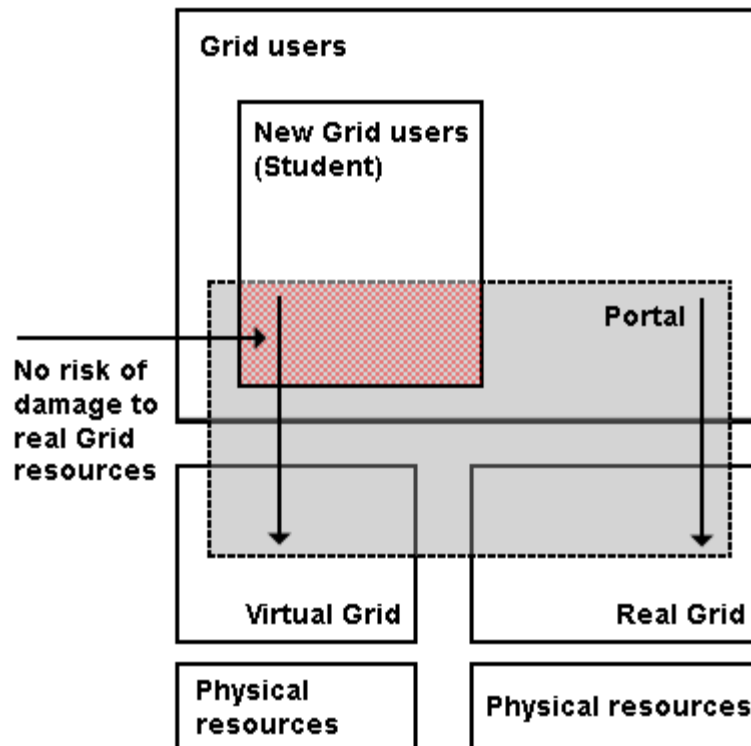


Figure 6.2: Project of Grid teaching problem domain with solution.

6.3 Requirements

6.3.1 Tutor's Functional Requirements

R1. Allow for the configuration of a Simulation Grid such that resources may be made either available or unavailable.

R2. Allow for on-demand Virtual Resource creation.

R3. Allow for on-demand Virtual Resource destruction.

R4. Allow virtual resources to be named identically to their real Grid equivalents.

R5. Virtual Resources should be configurable through the P-Grade Portal by tutors.

R6. P-Grade Portal should be used as the teaching interface.

R7. The teaching environment should be easy and inexpensive to setup.

R8. Simulation Grid maintenance and administration time should be minimal.

6.3.2 Student's Functional Requirements

R9. Provide students with an environment that allows job submission and job processing away from a real Grid.

R10. Allow for the generation of a proxy certificate such that the Student may use teaching Grid.

R11. Allow for the generation of workflows.

R12. Allow for the retrieval of workflow results.

6.3.3 Non Functional Requirements

R13. The solution should be work run in a Linux/Unix operating system environment.

R14. The solution should run in a Tomcat container.

6.3.4 Interface Requirements

Use interface design for both Student's and Tutor's greatly impacts on the success of any potential solution developed. The importance of user friendly Grid interfaces has already been discussed in section 2.8. It is important to continue the efforts such as those of Gridsphere [13] that aim to lower the entry barrier to Grid computing. This is of particular importance since a teaching tool, will inheritably be used by experienced users who are new to Grid.

Good user interface design helps to avoid the employment of complicated programming solutions in verifying user input, further reinforcing the important role user interface design plays in defining a solution to the Grid teaching problem. For example restricting user input through user interface design is infinitely better than programming solutions to validate user input and then provide meaningful feedback. The interface specification is defined in figure 6.3.4.

R15. The user interface should provide students with transferable knowledge in using Grid enabled web portals.

R16. The user interface should minimise the possibility of error when creating and destroying virtual Grid resources.

Virtual Resources			
Current Grids	xxx.xxxxxxx.xx.xx	▼	Load
Current grid resources	???.?????????.???.??	▼	Virtualise
Virtual grid resources	xxx.xxxxxxx.xx.xx	▼	Remove
Description of functionality			

Figure 6.3: Configuration portlet user interface.

6.4 Use case analysis

From requirements, a use case can be produced detailing the relationships that exist between users of the system and requirements.

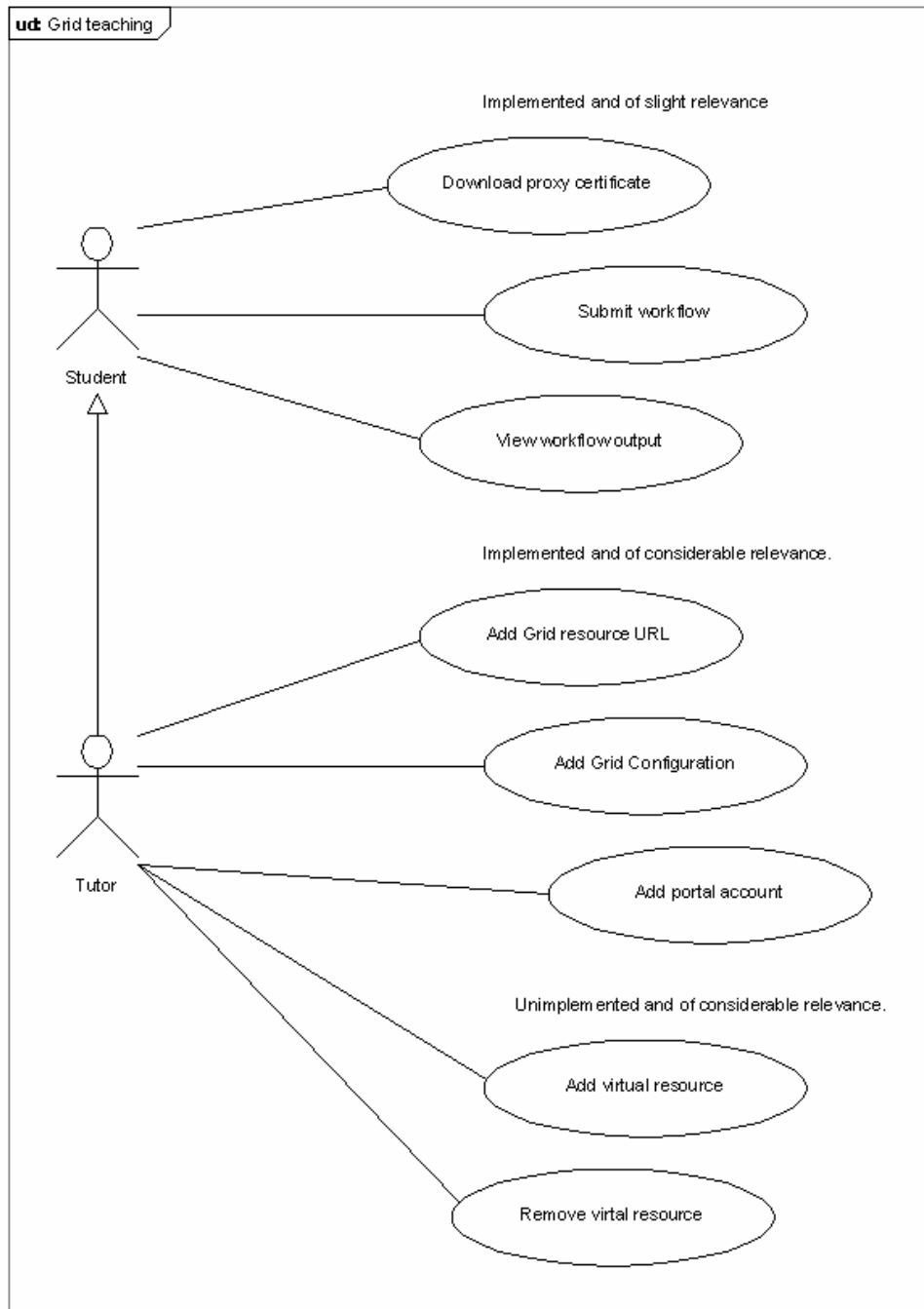


Figure 6.4: Use case diagram outlining requirements of Grid teaching solution.

The use case diagram in figure 6.4 is comprised of eight use cases and two actors. The two actors are

- Student - the individual who is being taught Grid computing.
- Tutor - the individual who is teaching Grid computing using the Grid teaching solution.

There are three categories of use case:

1. Unimplemented and of considerable relevance – the use case has yet to be implemented. Finding a practical solution to uses cases in this category forms the epicenter of this research project.
2. Implemented and of considerable relevance – the use case has already been implemented and its correct operation is of primary importance to the use cases in the ‘Unimplemented and of considerable relevance’ category.
3. Implemented and of slight relevance – the implementation of the use case is already complete and its correct operation is of slight relevance to the uses cases in the ‘Unimplemented and of considerable relevance’ category.

‘Download proxy certificate’, ‘Submit workflow’ and ‘View workflow output’ are implemented and of slight relevance. ‘Add Grid resource URL’, ‘Add Grid configuration’ and ‘Add portal account’ are implemented and of considerable relevance. ‘Add virtual resource’ and ‘Remove virtual resource’ are unimplemented and of considerable relevance.

The use cases belonging to the category ‘Unimplemented and of considerable importance’ will be developed into design scenarios while the others will be discussed so as to provide an introduction into their purpose, however there is no requirement to develop them into scenarios or UML diagrams.

7.0 Design

This chapter attempts to extract a suitable Grid teaching solution from three candidate solutions. In determining the suitability of each potential solution, figure 2.1 is applied, systematically, to each grid configuration outlined in sections 7.1, 7.2 and 7.3.

7.1 Installation of multiple head nodes on a single cluster

Installing multiple submit machines, each with one gatekeeper, on a single cluster provides users with the illusion that they have access to multiple resources. Making individual submit machines either available or unavailable on the cluster would act as a basic configuration mechanism, resulting in grid jobs either completing successfully or failing. Grid jobs would be submitted through a Grid enabled web portal. The portal would be configured to list all available submit machines on the cluster by default.

Gatekeepers would be identified by the IP addresses of the machine to which it is installed. A friendly domain name could also be used. However domain names would be restricted by:

- Cost – domain names can become expensive if they are in demand.
- Administration – the administrative cost involved in purchasing and setting up of domain names makes them highly troublesome to integrate into an ‘on-demand’ resource creation policy. That is to say, it would be difficult to allow new resources to be created for the purposes of teaching, and then to immediately assign, to those resources, a new domain name.
- Naming convention - the illusion of using a real Grid would be compromised since, for example, the resource address `grid-compute.oesc.ox.ac.uk` is already assigned to Oxford’s cluster and therefore could not be used to name teaching resources.

Resource	Port	IP address	Name
R1	2119	161.74.69.117	node01.grid-teaching.wmin.ac.uk
R2	2119	161.74.69.118	node02.grid-teaching.wmin.ac.uk
R3	2119	161.74.69.119	node03.grid-teaching.wmin.ac.uk

R4	2119	161.74.69.120	node04.grid-teaching.wmin.ac.uk
----	------	---------------	---------------------------------

Table 7.1: Port, IP and Name values applied to the multiple submit machine candidate solution.

Table 7.1 assumes four resources are required to create a meaningful teaching resource. If the resources are to be publicly available then the solution requires four public IP addresses. All gatekeepers would run on the default port of 2119, as such it would not be a requirement for resources to be post-fixed with a port number.

7.1.1 Advantages

- Realistic, as students would be using a real cluster.
- Failure of a submit machine disrupts teaching.

7.1.2 Disadvantages

- Physical access to a cluster required.
- Expert knowledge of Grid middleware required.
- Grid resources are static and may only be switched on and off.
- New resources cannot be added dynamically.
- Resources cannot be removed dynamically.
- Recreates Grid architecture, therefore resource heavy and contrary to the notion of a separate Grid teaching resource.
- Physical access to cluster required to simulate the varying availability of resources.

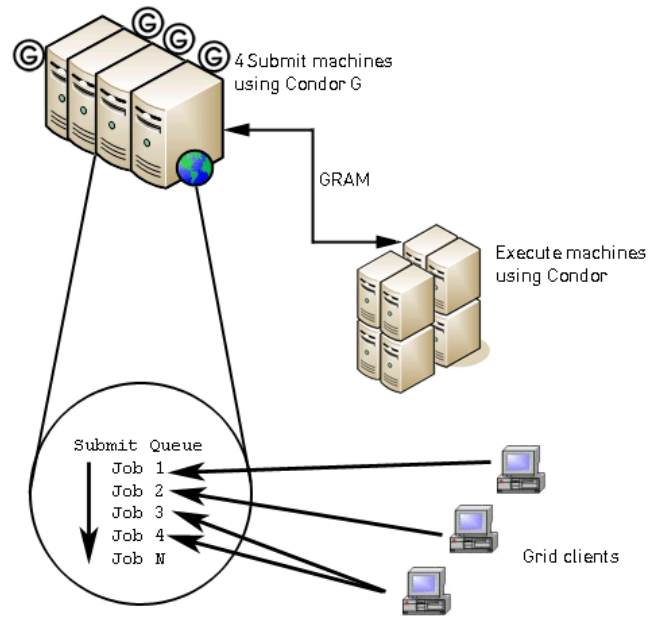


Figure 7.1: The relationship between Condor, Condor G, GRAM and Grid clients, applied to multiple submit machine.

7.2 Running multiple gatekeepers on a single head node

It has previously been explained in this paper that a single submit machine may run multiple gatekeepers. This provides a potential solution. Gatekeepers have two basic requirements.

1. They must run as `root`.
2. They must run on port that is unique to them.

Grid jobs would be submitted through a Grid enabled web portal. The portal would be configured to list all available submit machines on the cluster by default.

Resource	Port	IP address	Name
R1	2119	161.74.69.117	node01.grid-teaching.wmin.ac.uk
R2	3239	161.74.69.117	node01.grid-teaching.wmin.ac.uk
R3	2319	161.74.69.117	node01.grid-teaching.wmin.ac.uk
R4	5558	161.74.69.117	node01.grid-teaching.wmin.ac.uk

Table 7.2: Port, IP and Name values applied to the multiple gatekeeper candidate solution.

Table 7.2 assumes four resources are required to create a meaningful teaching resource. All IP addresses remain the same, as do the names of the teaching resources. A consequence of the lack of distinction between resource names means that the port number of each gatekeeper must be explicitly provided when specifying a teaching resource. For example `node01.grid-teaching.wmin.ac.uk:3239` would be the full address of teaching resource R2. Nevertheless, it would still be possible to submit a Grid job to `node01.grid-teaching.wmin.ac.uk`, providing a gatekeeper was running using the default port 2119.

The availability of virtual Grid resources would be modified by gatekeepers being started and stopped on the head node.

7.2.1 Advantages

- Number of gatekeepers limited only by number of free ports.
- Failure of a gatekeeper does not disrupt teaching.

7.2.2 Disadvantages

- Resource names unrealistic.
- Port number must be explicitly provided.
- Failure of a submit machine disrupts teaching.
- Number of virtual resources restricted by free ports.

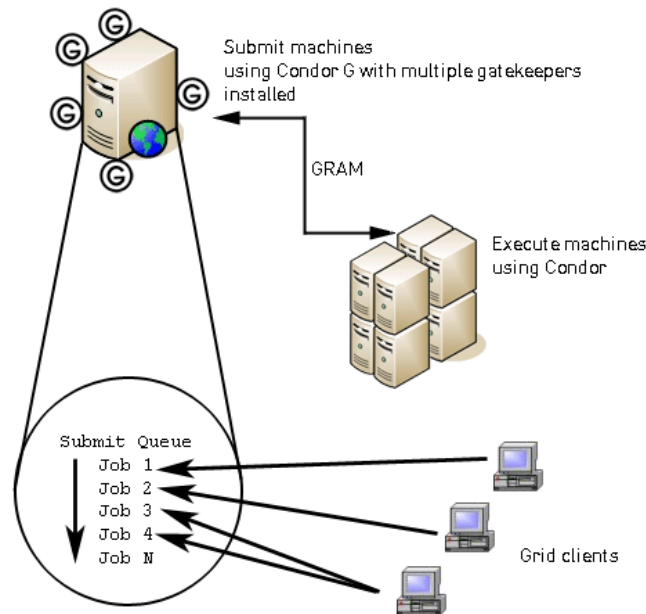


Figure 7.2: The relationship between Condor, Condor G, GRAM and Grid clients, applied to multiple gatekeepers.

7.3 Hostname mapping

A `hosts` file is a simply formatted system file that acts as a local Domain Name Service (DNS). Both Microsoft Windows and Linux have a `hosts` file following the syntax of: IP-Address, followed by at least one space, followed by a hostname. For example, an entry of:

```
209.73.186.238 www.google.com
```

would map the name `www.google.com` to the address `209.73.186.238`. If a user opened their favorite web browser and entered `www.google.com`, they would see the Yahoos! [35] Website, since, in this example the IP address is that of `www.yahoo.com`. It is this feature that provides a potential Grid teaching solution.

A gatekeeper would run on the default port of 2119 and would therefore be included implicitly when submitting a Grid job to a submit machine. The resource names have no

restrictions and may be names that are already in use on a real Grid. The hosts file would be checked before an external DNS lookup is performed. The entry

```
127.0.0.1 grid-data.man.ac.uk
```

would map Manchester's [26] Grid resource name to the local machine. Default port of 2119 would be used as it is not possible to specify IP addresses in combination with port numbers in hosts files.

Configuring the availability of virtual resources could be achieved through adding and removing entries on the hosts. For example, removing the entry

```
127.0.0.1 grid-data.man.ac.uk
```

would result in the Grid resource, grid-data.man.ac.uk, becoming unavailable. The Student's Grid job would fail. The solution is lightweight; there is no physical limitation to the number of virtual Grid resources. Configuring 25 virtual Grid resources would result in 25^2 , or 625 different virtual Grid combinations.

Resource	Port	IP address	Name
R1	2119	161.74.69.117	grid-data.rl.ac.uk
R2	2119	161.74.69.117	grid-data.man.ac.uk
R3	2119	161.74.69.117	grid-compute.leeds.ac.uk
R4	2119	161.74.69.117	grid-compute.oesc.ox.ac.uk

Table 7.3: Port, IP and Name values applied to the hosts file candidate solution.

Table 7.3 assumes four resources are required to create a meaningful teaching resource. Only one submit machine, running one gatekeeper is required. There is no restriction on the name of teaching resources.

7.3.1 Advantages

- Any resource name including names already assigned to Grid resources may be used.
- Only one submit machine required.
- Only one gatekeeper running, therefore lightweight solution.
- No limit on the number of resource that may be created.
- Resources may be added and removed at any time.
- Names may be assigned to resources as they are created.
- Lightweight solution.

7.3.2 Disadvantages

- Failure of a gatekeeper disrupts teaching.
- Failure of a submit machine disrupts teaching.

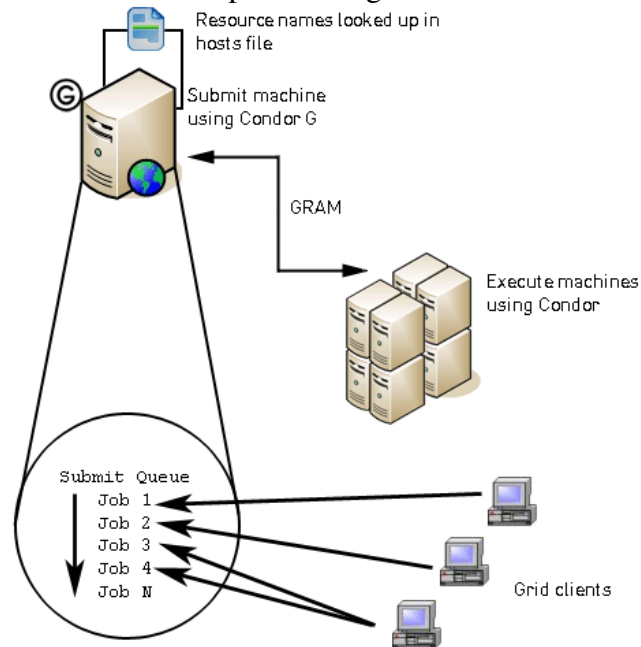


Figure 7.3: The relationship between Condor, Condor G, GRAM and Grid clients, applied to a hosts file lookup.

7.4 Solution

Through the application of configurations C1, C2 and C3 (introduced in section 7.1, 7.2 and 7.3) to the architecture shown in figure 2.1, a fair assessment of the potential successfulness and flexibility of each configuration has been conducted. Table 7.4 compares the following characteristics of each candidate solution:

- Realism – The accuracy of the simulation as an end user would experience.
- Security – The risk of damage to underlying physical resources or other Grid jobs.
- Extendibility – The potential for further work or customisation of the solution if it involves third party software.
- Feasibility – Time required to successfully develop the solution as well as technical expertise and resources required.
- Robustness – The degree to which the solution can function correctly.

	Realism	Security	Extendibility	Feasibility	Robustness
Workspace management service	Realising the WSM service through virtualisation technologies would result in a realistic Grid experience since real Grid would be used for running Grid jobs. There is no simulation in this solution.	The use of virtualisation technologies such as hyperivisors (virtual machine managers) provide excellent job execution isolation and security. Realising the WSM service through dynamic accounts less so.	Virtual machine technology is notoriously complex; it would be difficult to extend any work without substantial resources and time.	Even to run as few as four virtual machines, would require significant computational power. It is difficult to justify such expenditure purely on a Grid teaching environment. However the excellent job isolation abilities of virtual machines would possibly allow the simultaneous execution of live Grid jobs.	Virtual machine technology has experienced resurgence as a result of the potential use in Grid computing. However the technology is complex leading to instability when compared to simpler solutions.
Simulators	The simulators assessed in this project are not design to offer a realistic Grid experience in terms of human interaction. They are designed to evaluate the use of programs, executables and algorithms on Grid. For this reason simulators struggle to	Simulators are unlikely to cause security problems since GridSim and OptorSim are both self contained Java programs that run on a single computational resource. The distributed nature of the simulator is	Both OptorSim and GridSim are well packaged Java programs. The source code to GridSim is freely available, making it highly extendible.	Both simulators are free and easily to install. Source code for GridSim is available. For this reason the simulators introduced in this project are feasible.	Both simulators were installed and tested using example programs that came packaged. No problems were experienced using the sample programs and installation was error free. A longer period of testing would be required in

	provide a realistic Grid experience.	mimicked within the program code.			ascertaining the robustness of OptorSim and GridSim. Initial results suggest both are robust.
Grid testbeds	GridLab Testbed and GILDA were both highly realistic as they are both implemented as a live Grid. GILDA has multiple clusters cross Italy and throughout the world. Authorization was realistic as was virtual organisation membership.	Both GridLab Testbed and GILDA are implemented using real Grid architectures, toolkits and software. For this reason they are exposed to the same security risks as live Grid. The only real difference being that damage to Grid resources would only affect other educational users and not commercial Grid jobs being processed.	It would be difficult to extend either GILDA or GridLab since they are large distributed architectures.	Little configuration or organisation is required to use GridLab or GILDA. However the authorization process for GILDA membership was lengthy and complex.	Both testbeds offer good robustness since they are comprised form many distributed resources.
Multiple head node configuration	The installation of multiple head nodes on a cluster would provide	It would still be possible for Grid jobs to damage the	The physical resources used for Grid job execution	Installing multiple head nodes on a cluster is not feasible since, firstly a	Very robust since the solution is comprised of

(C1)	for a realistic Grid experience, the architecture is very similar to a live Grid. Execution time of Grid jobs may degrade due to the limited computational power of a single cluster.	physical resource; however this would not impact on the live Grid.	could be configured differently to extend the project.	cluster is required and secondly Grid architecture would be recreated, requiring a high level of expertise.	multiple execute machines as well as multiple submit machines. Failure of a gatekeeper, head node or processing machine would not disrupt teaching of Grid.
Multiple gatekeeper configuration (C2)	The installation of multiple gatekeepers on a single submit machine would provide for a realistic Grid experience. The same justifications for the multiple head node configuration are given. Execution time of Grid jobs would be degraded significantly due to the processing limits of a single submit machine. This could be overcome by attaching the submit machine to a cluster.	It would still be possible for Grid jobs to damage the physical resource; however this would not impact on the live Grid.	The physical resources used for Grid job execution could be configured differently to extend the project.	The installation of multiple gatekeepers running on unique ports is feasible and uncomplicated to configure.	Fairly robust since failure of a gatekeeper would not result in disruption to teaching of Grid. Failure of the submit machine would.
Hostname mapping	Hostname mapping configuration would	It would still be possible for Grid	The physical resources used for	Highly feasible as only a simple hosts file	Not robust at all since failure of a

configuration (C3)	provide excellent realism for a new Grid user since the names of resources could be identical to those used on a live grid.	jobs to damage the physical resource; however this would not impact on the live Grid.	Grid job execution could be configured differently to extend the project.	requires updating in order to change the grid teaching environment. All Linux machines have a <code>hosts</code> file by default.	gatekeeper, submit machine or execute machine would disrupt teaching of Grid.
---------------------------	---	---	---	---	---

Table 7.4: Summary of the main characteristics dictating the quality of each candidate solution.

The hostname mapping solution meets all requirements presented in section 6.3 and as such will be implemented as the solution to the problem of grid teaching.

Scenarios are the first step in detailing the logical steps involved in developing use cases labeled ‘Unimplemented and of considerable relevance’, which include:

- Adding virtual resource.
- Removing virtual resource.

7.5 Unimplemented and of considerable relevance

7.5.1 Scenarios

Identifier:	SC1
Name:	Add virtual resource
Description:	Tutor adds resource URL to list of virtualized resources.
Actor(s):	Tutor
Principle actor:	Tutor
Preconditions:	The Tutor has a Portal account with a role of ‘super’ or ‘admin’.
Flow of events:	<ol style="list-style-type: none"> 1. The tutor logs into the Grid Enabled Web Portal. 2. The tutor selects a portlet that provides the function of adding a virtual resource. 3. The tutor selects grid from which to load resource URLs. 4. Grid resource URLs loaded. 5. Tutor select resource URL to virtualise. 6. Resource URL is mapped to the IP address of the submit machine.

	7. Resource URL is added to repository of existing virtualised resource URLs.
Alternative flow:	1,2,3,4 and 5 occur according to the normal flow of events. 6. The mapping between the resource URL and the submit machine's IP address already exists. 7. The resource URL is not mapped to the submit machines IP address.
Post-conditions:	The operating system's hosts file contains a mapping between the newly virtualised resource and the submit machine.
Exceptions:	

Identifier:	SC2
Name:	Remove virtual resource
Description:	Tutor adds resource URL to list if virtualized resources.
Actor(s):	Tutor
Principle actor:	Tutor
Preconditions:	The Tutor has a Portal account with a role of 'super' or 'admin'.
Flow of events	1. The tutor logs into the Grid Enabled Web Portal. 2. The tutor selects a portlet that provides the function of adding a virtual resource. 3. The tutor selects resource URL of previously virtualised resources. 4. The mapping between the submit machine IP address and the virtual resource URL is removed.
Alternative flow:	
Post-conditions:	The operating system's hosts file no longer contains a mapping between the once virtualised resource and the submit machine.
Exceptions:	1,2,3 and 4 occur according to the normal flow of event. 5. The persistent storage does not contain the resource URL to be removed. 6. An error message is displayed informing the user to contact their administrator, or to inspect the hosts file manually.

7.5.2 Sequence diagrams

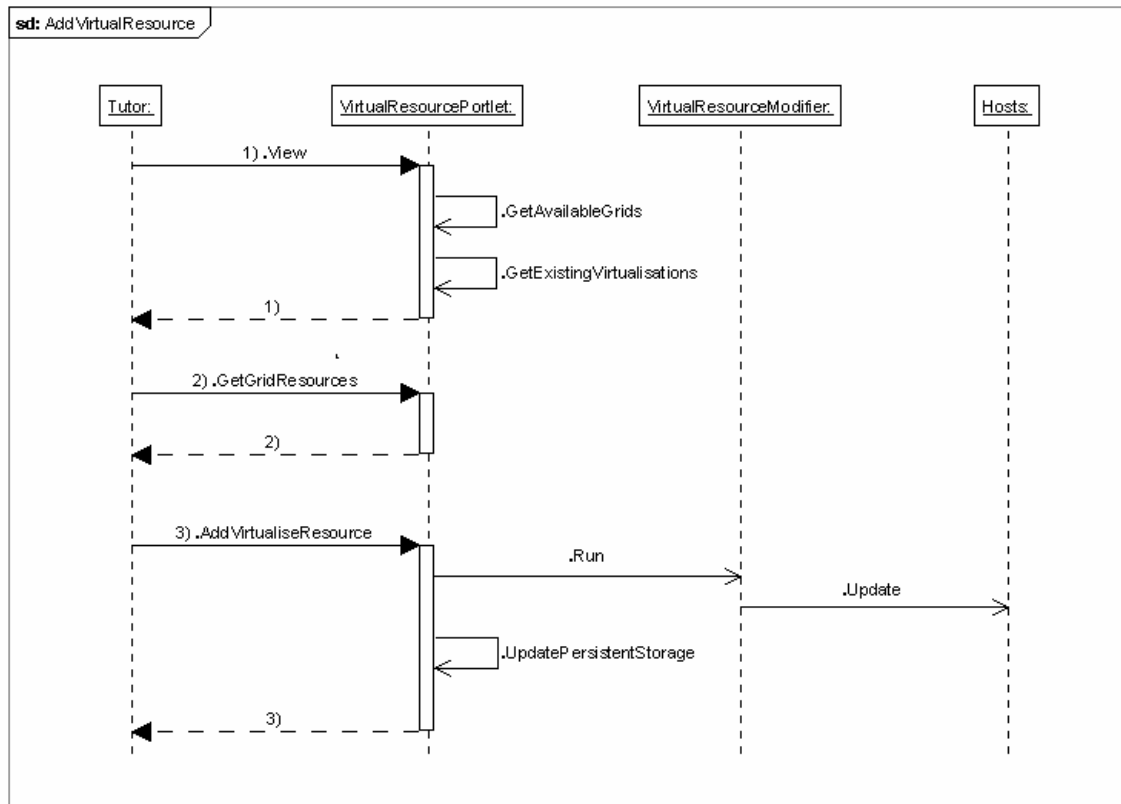


Figure 7.4: Sequence diagram detailing logical steps for use case ‘Adding Virtual Resource’.

The tutor adds virtual resources using a portlet. During portlet initialisation two functions (`GetAvailableGrids`, `GetExistingVirtualisations`) will run:

1. `GetAvailableGrids`: The names of all Grids attached to the grid enabled web portal are retrieved and the user interface is updated appropriately.
2. `GetExistingVirtualisations`: The names of all current virtualised Grid resource URLs are retrieved and the user interface is updated appropriately.

Functions `GetAvailableGrids` and `GetExistingVirtualisations` will not be triggered explicitly, instead they run when the portlet is initialised, resulting in a constantly updated and prepared user interface. Once both functions complete the tutor may select a Grid and retrieve the resource URLs attached to that Grid. Finally the tutor may select one of the retrieved resource URLs, and subsequently request that it be virtualised. The newly virtualised resource URL is then added to persistent storage. The `VirtualResourceModifier` updates the operating system’s `hosts` file by adding a mapping between the resource URL and the submit machines IP address.

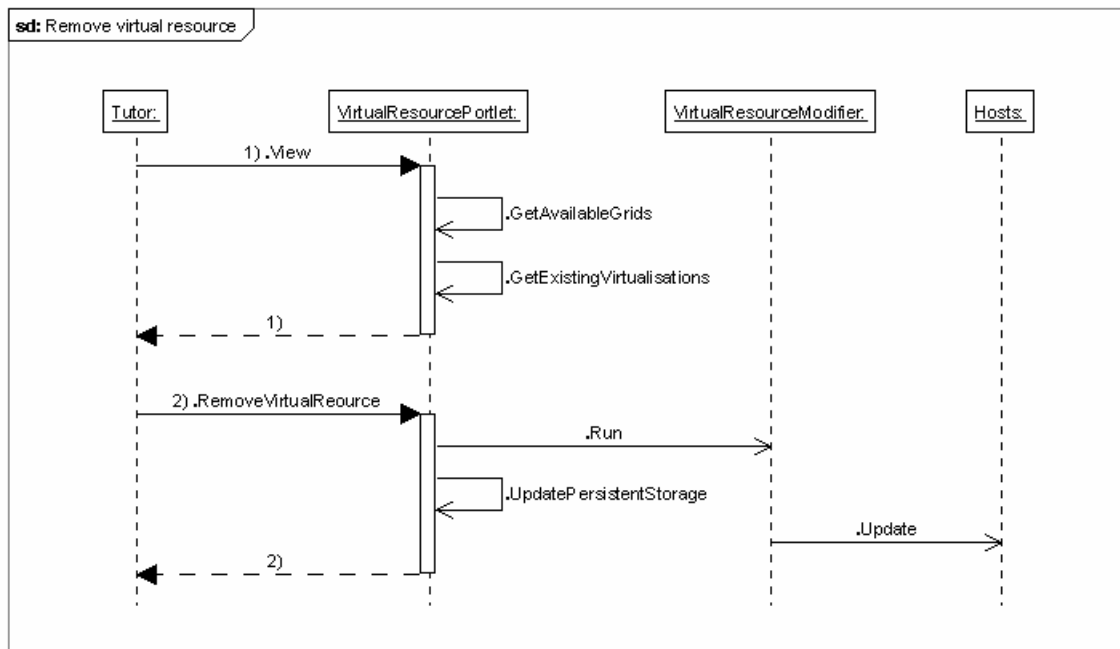


Figure 7.5: Sequence diagram detailing logical steps for use case ‘Removing Virtual Resource’.

The tutor removes virtual resources using a portlet. As with sequence diagram ‘Adding virtual resource’, functions `GetAvailableGrids` and `GetExistingVirtualisations` will run during portlet initialisation. Removing a virtual resource is less complex than adding one since the tutor is not required to load resource URLs. The tutor may immediately select from a list of existing virtualised resource URLs the one they wish to remove. The `VirtualResourceModifier` then updates the operating system’s `hosts` file by removing the mapping between the resource URL and the submit machines IP address.

Use cases belonging to the categories ‘implemented and of considerable relevance’ and ‘implemented and of slight relevance’ will now be outlined.

7.6 Implemented and of slight relevance

7.6.1 Download proxy certificate

The system currently accepts certificates issued by the UK e-Science certificate Authority and those CAs with which the UK e-Science CA has agreements. Proxy certificates are downloaded from a proxy server, for example `myproxy.grid-support.ac.uk` using a portlet. However there is no restriction on the proxy server to be specified, allowing the Tutor to specify a proxy server that issues proxy certificates purely for submit machines

on the virtual grid. Such certificate would not validate on a live Grid such as the NGS, thus ensuring the integrity of the live Grid.

7.6.2 Submit workflow

Workflows can be created through the use of P-Grade [31], a Grid portal installed on GEMMLCA/P-GRADE portal nodes.

7.6.3 View workflow output

Workflow output, or results, can be retrieved using P-Grade.

7.7 Implemented and of considerable relevance

7.7.1 Add Grid configuration

Grid configurations can be added by Tutors providing they have access to an account with either admin or super-user permissions. Grid configurations cannot be changed by Grid users. P-Grade can be used for this.

7.7.2 Add Grid resource URL

Resource URLs can be added to a Grid configuration by any user. P-Grade can be used for this. Default may also be set by administrators' and super-users. P-Grade can be used for this.

7.7.3 Add portal account

Portal accounts can be added by the Tutor providing they have access to an account with either admin or super-user permissions. Normal users cannot add, remove or update portal accounts.

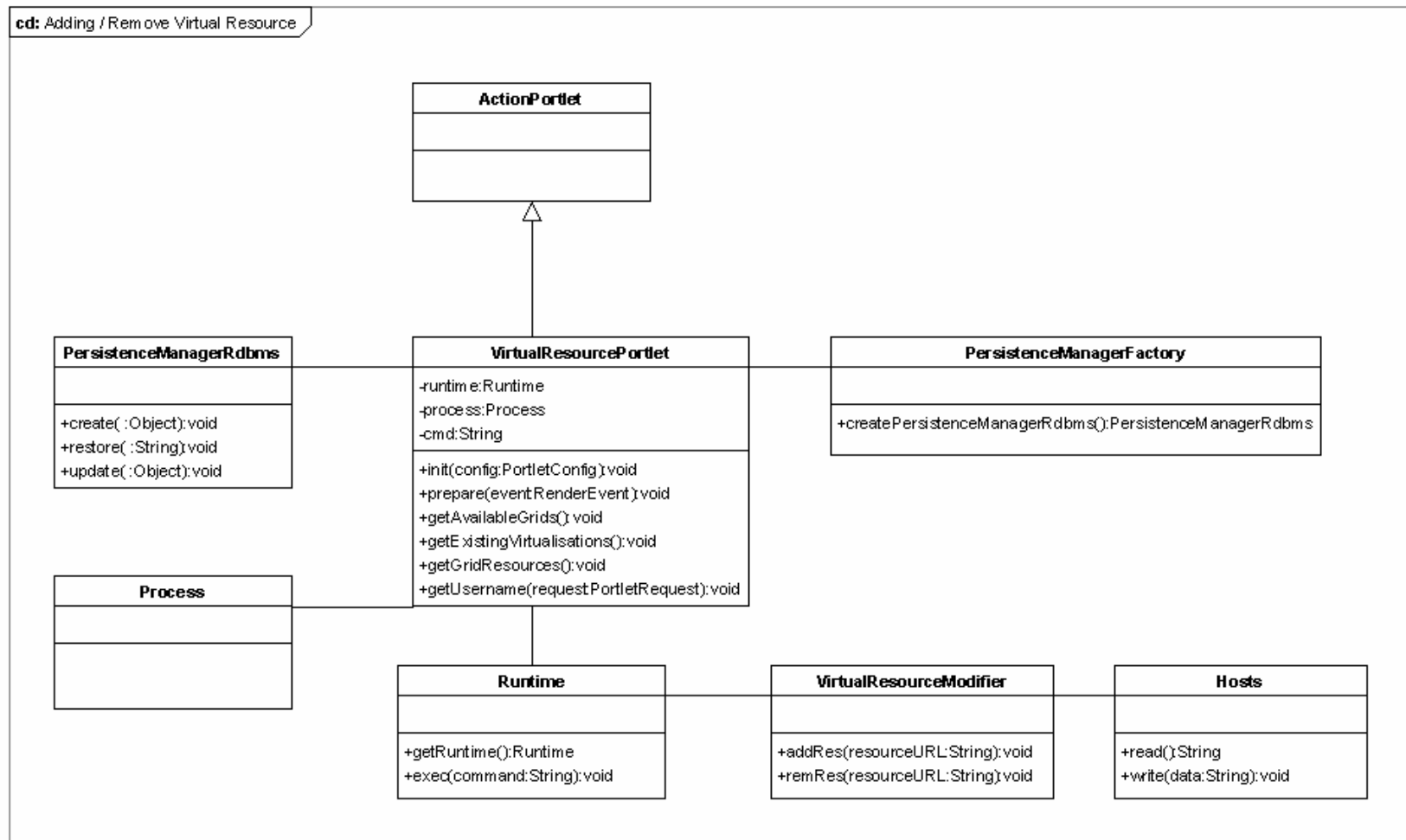


Figure 7.6: Class diagram of solution: hostname mapping configuration.

8.0 Implementation

Requirements R2 and R3 (section 6.3.1) were implemented in solving the Grid teaching problem.

8.1 Resources

The following resources were required in the implementation of requirements R2 and R3:

- Gridsphere 2.0.2
- Hibernate Query Language (HQL)
- Ant 1.6.5
- Tomcat 5.0.28
- J2EE
- JSP
- Java User Interface Beans
- Sztaki P-Grade API
- Linux
- Stream Editor (SED)
- Bash Shell

8.2 JSR Portlet

A Java Specification Request 168 [21] standards compliant portlet was developed to allow for dynamic virtual resource creation and dynamic virtual resource destruction. The portlet was implemented using the Gridsphere [13] portal framework, Ant [2] and the Tomcat web server [3]. Ant was installed on Linux to deploy the portlet into a Tomcat web server container. It should be highlighted that although Tomcat was used in this project, any servlet container meeting the Servlet Specification 2.4 [38] may be used.

The portlet used a Java Server Page and Java User Interface Beans to dynamically construct a user interface that:

- Displayed all existing Grid configurations.

- Retrieved all Grid resource URLs attached to Grid configurations.
- Created new virtual Grid resource URLs.
- Destroyed existing virtual Grid resource URLs.
- Saved current virtual Grid resource URLs to persistent storage which transcended the lifetime of the portlet.

Using Ant a new portlet project was created with the following directory and file structure automatically created:

File or Directory	Description
<code>build.xml</code>	Contains targets for the default code structure to compile and deploy the portlet in Gridsphere to a servlet container using Ant. This code did not require any modifications
<code>/lib</code>	Directory to store third party libraries that the Virtual Resource portlet required. In this case the Sztaki P-Grade portal libraries were included since they are used in the portlet.
<code>src</code>	Directory to store all Java based source code. To ensure the correct functioning of Ant when deploying Gridsphere portlets it is necessary to ensure all portlets exists in a subdirectory of <code>src</code> named <code>portlets</code> .
<code>webapp</code>	Directory containing three directories <ol style="list-style-type: none"> 1. <code>html</code> (location of <code>html</code>). 2. <code>jsp</code> (location of <code>jsp</code> modules & views). 3. <code>WEB-INF</code> (discussed next).
<code>webapp/WEB-INF</code>	Directory containing the <code>jsr</code> portlet specification, including two compulsory files that are specified as part of the <code>jsr 168</code> : <ol style="list-style-type: none"> 1. <code>web.xml</code> 2. <code>portlet.xml</code>

Table 8.1: Skelton Gridsphere project structure generated using Ant.

Three files required modification to allow integration into the portal:

`portlet.xml` was modified to include the location of the portlet containing the functionality fulfilling requirements R2 and R3 (section 6.3). The following line was added:

```
<portlet-class>
    org.gridisphere.gsexamples.portlets.Resource
</portlet-class>
```

`layout.xml` was modified to ensure the portlet appeared correctly in the portal. The location of the Virtual Resource portlet was also specified:

```
<portlet-frame label="resourcename">
    <portletclass>
        org.gridisphere.gsexamples.portlets.Resource
    </portlet-class>
</portlet-frame>
```

`group.xml` was modeled on an automatically generated template named `group.sample.xml`, and modified to ensure that only users with the correct permissions could use the Virtual Resource portal and therefore create and destroy virtual Grid resources. The following lines were added:

```
<portlet-role-info>
    <portlet-class>
        org.myorg.portlets.adminportlet.AdminPortlet
    </portlet-class>
    <required-role>ADMIN</required-role>
</portlet-role-info>
```

8.3 Database configuration

Gridsphere is preinstalled with a Java Relation Database Management System (RDBMS). The underlying RDMS of Gridsphere defaults to HSQL. HSQL is a database implemented purely in Java. Gridsphere's default database configurations were adequate for the implementation of configuration C3. Underlying database configurations are defined in the file `hibernate.properties` located in `WEB-INF/persistence`.

```
hibernate.dialect=net.sf.hibernate.dialect.HSQLDialect
hibernate.connection.username=sa
hibernate.connection.password=
hibernate.connection.url=jdbc:hsqldb:@3RDPARTY_WEBAPP@/WEB-
INF/database/gridsphere
hibernate.connection.driver_class=org.hsqldb.jdbcDriver
```

8.4 JSR Portlet View

Portlet development can become cumbersome and complex since portlets that extend `GenericPortlet` are restricted to a finite set of methods in which to implement any processing or logic. Usually this is `doView(RenderRequest, RenderResponse)`. To avoid the degeneration of portlet methods into unmanageable `if/else` statements checking the value of a specific attribute or action, the `ActionPortlet` was extended, replacing `GenericPortlet`.

The user interface used a simple tag library and beans combination to ensure all control remained in the portlet, thus enforcing the MVC paradigm that the Gridsphere framework promotes. The JSP only presents a view of the portlet. Development with Java User Interface Beans is straightforward since they strongly resemble standard XHTML tags.

`ActionSubmit` UI Beans specify action attributes that map directly onto methods of a portlet providing the portlet extends `ActionPortlet`. Three action attributes listed in table 8.2 represent three different methods, two of which relate directly to requirements R2 and R3, they are:

Action value	Requirement	Description
<code>loadResources</code>	N/A	Sends a request to retrieve all Grid resource URLs attached to a specific Grid configuration.
<code>add</code>	R2	Adds a resource URL (retrieved from the <code>loadResources</code> action) to the operating systems <code>hosts</code> file.
<code>remove</code>	R3	Removes a resource URL (retrieved from persistent storage) from the operating systems <code>hosts</code> file.

Table 8.2: Virtual Resource Portlet UI Bean action methods.

```

1.   <ui:form>
2.   <ui:table>
3.   <ui:tablerow>
4.   <ui:tablecell>Select Grid:</ui:tablecell>
5.   <ui:tablecell>
6.   <ui:listbox beanId="gridsListBox"></ui:listbox>
7.   </ui:tablecell>
8.   <ui:tablecell>
9.   <ui:actions submit action="loadResources" value="Load
resources"/>
10.  </ui:tablecell>
11.  </ui:tablerow>
12.  <ui:tablerow>
13.  <ui:tablecell>Select resource:</ui:tablecell>
14.  <ui:tablecell>
15.  <ui:listbox beanId="resourcesListBox"></ui:listbox>
16.  </ui:tablecell>
17.  <ui:tablecell>
18.  <ui:actions submit action="add" value="Virtualise resource"/>
19.  </ui:tablecell>
20.  </ui:tablerow>
21.  <ui:tablerow>
22.  <ui:tablecell>Current VGRS:</ui:tablecell>
23.  <ui:tablecell>
24.  <ui:listbox beanId="vgrsListBox"></ui:listbox>
25.  </ui:tablecell>
26.  <ui:tablecell>
27.  <ui:actions submit action="remove" value ="Remove virtual
resource"/>
28.  </ui:tablecell>
29.  </ui:tablerow>
30.  </ui:table>
31.  </ui:form>

```

Code view of resource.jsp. Code not formatted as bold serves only to maintain an organised layout and does not impact on the functionality discussed in this chapter.

When the portlet is rendered for the first time then a unique method with no attached action is called, meaning it was not triggered from a Java UI Bean. In this example the method is `prepare`. It simply forwards the request to the jsp file `resource.jsp` to display the Virtual Resource interface to the Tutor. While JSPs can exist in many directories the method `setNextState` assumes they have been placed in `webapp/jsp`, a convention followed in the implementation of this research project.

```

1.   private static final String DISPLAY_PAGE = "resource.jsp";
2.
3.   . . . .
4.
5.   public void init(PortletConfig config)
6.   {
7.       DEFAULT_VIEW_PAGE = "prepare";
8.   }
9.
10.  public void prepare(RenderFormEvent event)
11.  {
12.      . . . .
13.          setNextState(event.getRenderRequest(), DISPLAY_PAGE);
14.      . . . .
15.  }

```

Since all functionality can be presented in one view (`resource.jsp`) the portlet simply loads the same view once each portlet action has completed. Upon logging in the Tutor may select the Virtual Resource portlet, the default view of which is displayed in figure 8.1.

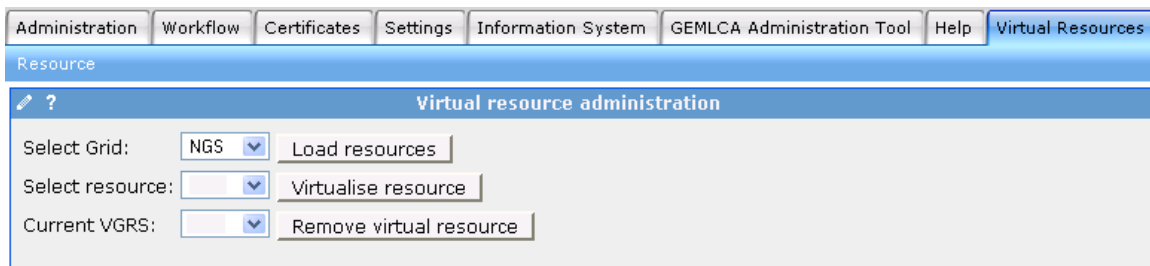


Figure 8.1: Screenshot of Initial portlet view.

8.5 Solution Architecture

Figure 8.10 presents a detailed view of the solution architecture. The implementation of requirements R2 and R3 is now discussed.

8.5.1 Adding a virtual resource

Virtual resources are stored as URLs, any valid Grid resource may also be a valid virtual resource. Two preconditions must be fulfilled before a Grid resource may be selected, by a Tutor, to be part of a virtual grid.

- P1. A grid configuration must be created using the P-Grade portal.

P2. A grid configuration must have at least one resource attached.

Precondition P1 is fulfilled through the P-Grade's 'Settings' portlet (figure 8.2).

The screenshot shows the 'Settings' portlet with the 'Settings' tab selected. The main content area is titled 'GRID configurations (DEFAULT configuration!)'. It contains a table with columns: Name, Information System (sub-columns: Type, Host, Port, BaseDn), and [Actions].

Name	Information System				[Actions]
	Type	Host	Port	BaseDn	
NGS		N/A			Resources Delete
Uow		N/A			Resources Delete
Virtual		N/A			Resources Delete

Below the table is a form to add a new configuration:

Name: Grid: Information System: N/A

Figure 8.2: Portlet view to add new Grid Configurations.

The first precondition (P1), is satisfied by adding three Grid configurations to the portal; NGS, UoW and Virtual. The second precondition (P2) states that a grid configuration must have at least one Grid resource attached. For demonstration purposes six resources were added to the Virtual Grid configuration (figure 8.3). At this stage there is nothing to distinguish a standard Grid configuration and attached resources, from the rather abstract idea of a virtual Grid configuration and virtual resources. The naming of a Grid configuration as 'Virtual' is purely to make it easier to keep track of throughout this chapter, and is perhaps a little confusing at this stage.

The screenshot shows the 'Settings' portlet with the 'Settings' tab selected. The main content area is titled 'Resources for 'Virtual' GRID (DEFAULT configuration!)'. It contains a table with columns: URL, Job manager, and [Actions].

URL	Job manager	[Actions]
grid-compute.cpc.wmin.ac.uk	jobmanager-fork	Delete
grid-compute.leeds.ac.uk	jobmanager-fork	Delete
grid-compute.oesc.ox.ac.uk	jobmanager-fork	Delete
grid-data.man.ac.uk	jobmanager-fork	Delete
grid-data.rl.ac.uk	jobmanager-fork	Delete
grid.lancs.ac.uk	jobmanager-fork	Delete

Below the table is a form to add a new resource:

URL: Job manager:

Figure 8.3: Portlet view to add new Grid resource URLs to a Grid configuration.

In order to differentiate between standard Grid configurations and virtual Grid configurations the Virtual Resources portlet is used. The `prepare` method of the portlet performs the task of retrieving Grid configurations using P-Grade libraries. The Portlet's default view is updated using the `ActionPortlet` to include the newly added Grid configurations (NGS, UoW, Virtual).

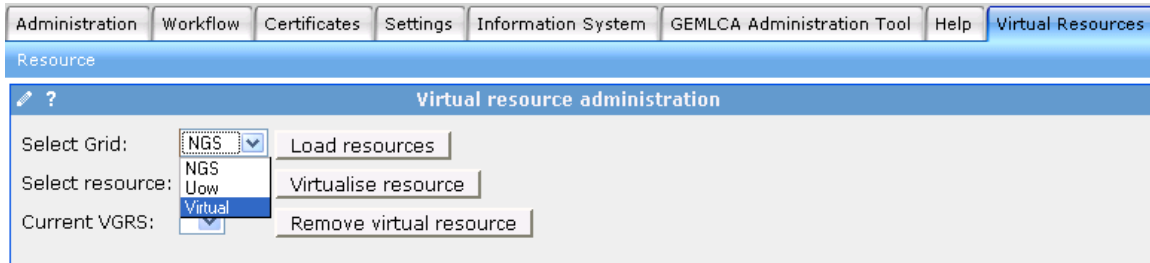


Figure 8.4: Default view of Virtual Resource portlet with Grid configurations.

```

1.  public void prepare(RenderFormEvent event)
2.  {
3.  /***** Start *****/
4.
5.  . . . .
6.
7.  /***** Grid resources *****/
8.
9.  // Load Grid Configurations
10. String[] availableGrids = GridConfigs.getGridNames();
11.
12. // Get reference to UI Bean
13. ListBoxBean gridsListBox= event.getListBoxBean("gridsListBox");
14. gridsListBox.clear();
15.
16. // Populate UI Bean with Grid configurations
17. for( int i=0;i< availableGrids.length; i++)
18. {
19.     ListBoxItemBean item = new ListBoxItemBean();
20.     item.setValue(availableGrids[i]);
21.     gridsListBox.addBean(item);
22. }
23.
24. . . . .
25. /***** Done *****/
26. }

```

A request to view resources attached to a Grid configuration may be performed by selecting the 'Load Resources' button (figure 8.4). Loading a Grid configuration's attached resources is the responsibility of the function `loadResources`, which utilises

P-Grade libraries to retrieve Grid resources. The interface is updated with the newly retrieved resources attached to the Virtual Grid configuration. Selecting a Grid resource (retrieved using the `loadResources` function), the Tutor may request it be virtualised by clicking the ‘Virtualise resource’ button. This process is displayed in figure 8.5.

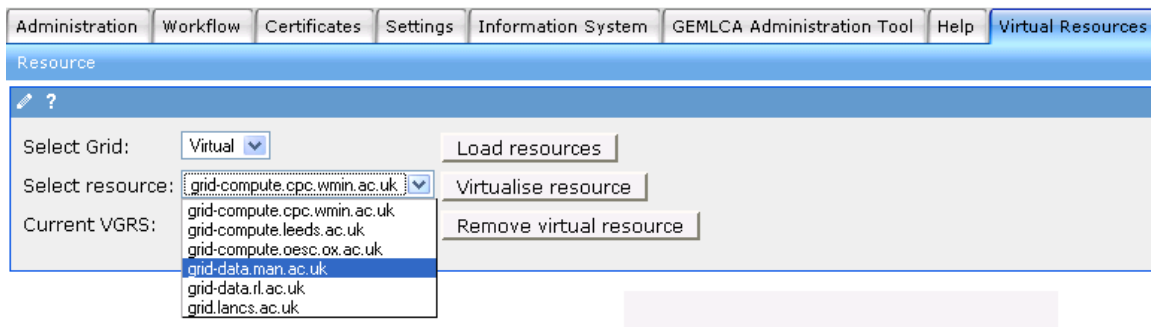


Figure 8.5: Default view of Virtual Resource portlet with Grid configurations and Grid resources.

```
1.    public void loadResources(ActionFormEvent event) throws
PortletException
2.    {
3.        ActionRequest req = event.getActionRequest();
4.
5.        // Retrieve UI info
6.        ListBoxBean gridsListBox= event.getListBoxBean("gridsListBox");
7.        ListBoxBean resourcesListBox=
event.getListBoxBean("resourcesListBox");
8.
9.        // Clear existing elements
10.       resourcesListBox.clear();
11.
12.       // Load name of grid from which to load resources
13.       String gridName = gridsListBox.getSelectedValue();
14.
15.       // Name of current user
16.       String username = getUsername(req);
17.
18.       // Actually load resource urls
19.       PGradeSettingsBean pgsb = new PGradeSettingsBean(username,
gridName);
20.       SZGJobManagerConfiguration[] jmConfigs =
pgsb.getJMConfigurations();
21.
22.       // Populate UI Bean with resource urls
23.       for( int i = 0; i < jmConfigs.length; i++)
24.       {
25.           ListBoxItemBean item = new ListBoxItemBean();
26.           item.setValue(jmConfigs[i].getContactString());
27.           resourcesListBox.addBean(item);
28.
29.       }
30.
31.       // Forward request to resource.jsp
32.       setNextState(event.getActionRequest(), DISPLAY_PAGE);
33.   }
```

In order for a Grid resource to be virtualised it must be added to the operating system's `hosts` file. Programmatically, this means a URL being passed as a command line argument to the shell script `addRes.sh`. The code to run a shell script from a portlet is now presented.

```
1.     public void add(ActionFormEvent event) throws
PortletException
2.     {
3.         ActionRequest req = event.getActionRequest();
4.
5.         /* Retrieve gui elements */
6.         ListBoxBean resourcesListBox =
event.getListBoxBean("resourcesListBox");
7.         String resourceName = resourcesListBox.getSelectedValue();
8.
9.         /* Build the command */
10.        cmd = ("/etc/addRes.sh " + resourceName);
11.
12.        try
13.        {
14.            /* Actually execute the shell script */
15.            process = runtime.exec(cmd);
16.
17.            /* Load existing resource urls */
18.            vgrsList = (VgrsList) pm.restore("from " +
VgrsList.class.getName());
19.
20.            String currentUrls = vgrsList.getUrl();
21.
22.            /* Add new resource url */
23.            currentUrls += resourceName + ",";
24.            vgrsList.setUrl(currentUrls);
25.
26.            /* Actually update */
27.            pm.update(vgrsList);
28.        }
29.        catch(Throwable t)
30.        {
31.            . . . .
32.        }
33.        . . . .
34.    }
```

addRes.sh inserts a Grid resource URL into the operating system's hosts file. Presuming the shell script runs successfully, the operating system's hosts file would gain a new entry of '127.0.0.1 <grid resource URL>'. It is at this point the Grid resource becomes a Virtual Grid resource, since it is now mapped to the loopback IP address of 127.0.0.1.

```

1.  #!/bin/bash
2.
3.  # Create temporary hosts file, because sed overwrites buffer
4.  cat /etc/hosts > /etc/temp_hosts
5.
6.  # Insert loopback IP, a space, and command line argument
7.  # Use temporary hosts file as input
8.  # Redirect std out to hosts
9.  sed -e '/vgrs/a\127.0.0.1      '$1 < /etc/temp_hosts >
/etc/hosts
10.
11. # Clean up afterwards
12. rm /etc/temp_hosts

```

After the shell script executes successfully, the newly virtualised resource is saved to persistent storage using Gridsphere's persistent storage manager. The interface is updated appropriately to display the newly virtualised resource.

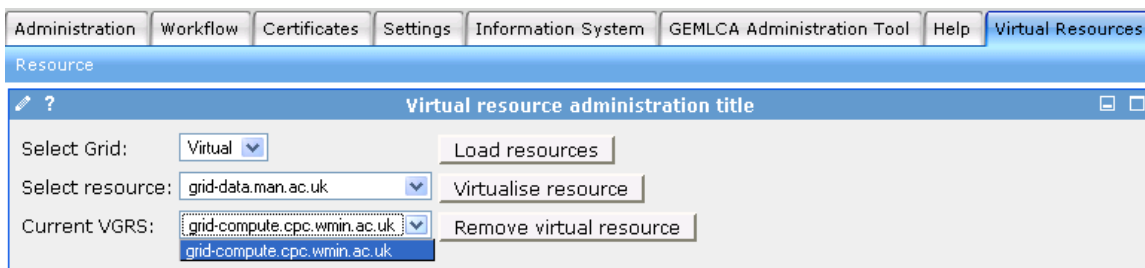


Figure 8.6: Screenshot of Virtual Resources Portlet: Newly created virtual resource.

To recap, the three Grid configurations (NGS, UoW, Virtual) were added to P-Grade. Six Grid resources were attached to the Virtual Grid. Using the Virtual Resources portlet, the Tutor was then able to select a Grid configuration and subsequently view any Grid resources attached to that configuration. A resource may then be selected and virtualised, a process which uses a shell script to insert a mapping between the special loopback IP address of 127.0.0.1 and the resource name. The newly virtualised grid resource is then stored in persistent memory using GridSphere's persistence manager which uses an underlying database implementation of HSQL.

8.5.2 Using virtual resources

A Student may then construct a simple workflow, in this example, four nodes are constructed:

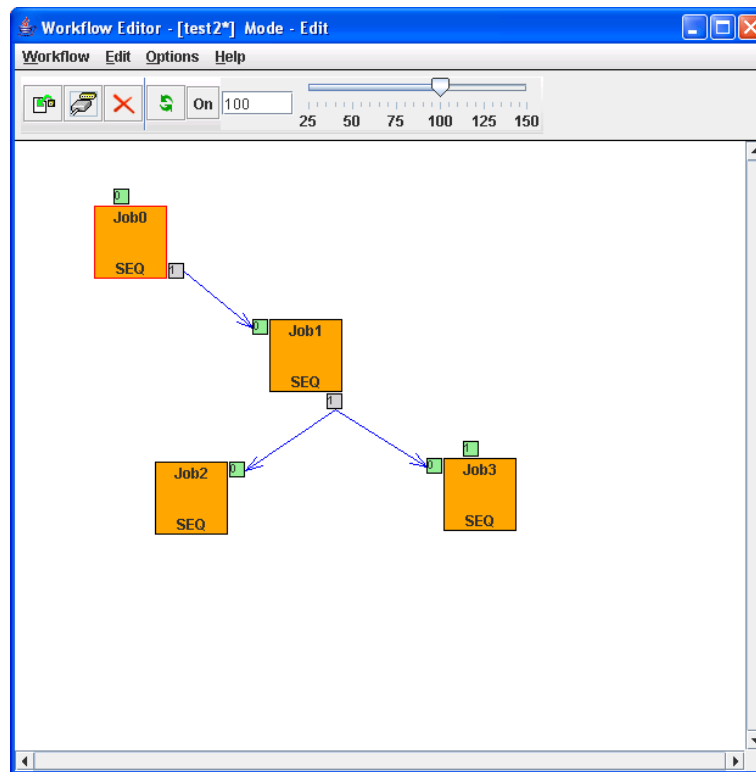


Figure 8.7: P-Grade workflow with four nodes.

Each node requires a resource URL, attached to a Grid configuration. In this example the Student must accept four Grid resources, one for each node. Providing the student selects the 'Virtual' Grid configuration they may choose from six grid resources:

1. grid.lancs.ac.uk
2. grid-data.rl.ac.uk
3. grid-data.man.ac.uk
4. grid-compute.oesc.ox.ac.uk
6. grid-compute.leeds.ac.uk
7. grid-compute.cpc.wmin.ac.uk

Assuming all resources are mapped to loopback IP address 127.0.0.1 in the operating system's `hosts` file (through the use of the Virtual Resources portlet), all nodes will execute successfully and the Grid job will run. However if the Tutor decided to virtualise only half of the six Grid resources, the Student has a 50% chance of selecting a Grid resource that causes the node to fail. In this scenario the Grid job would also fail. It is

this feature that allows the Tutor to model the (sometimes) unpredictable nature of Grid computing.

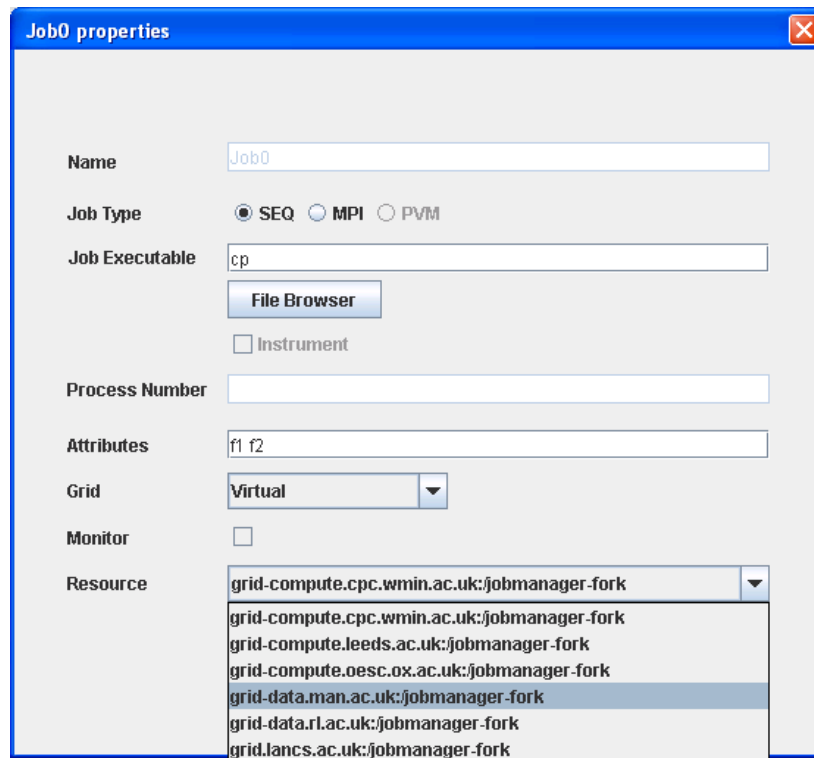


Figure 8.8: Selecting Grid configuration and resource URL for a node.

8.5.3 Remove virtual resource

A virtual resource is removed in much the same way as it was added. However, the shell script `addRes.sh` is replaced by `remRes.sh`.

```

1.  #!/bin/bash
2.
3.  # Create temporary hosts file, because sed overwrites buffer
4.  cat /etc/hosts > /etc/temp_hosts
5.
6.  # Use sed to remove all occurrences of the command
line URL
7.  # Direct temporary hosts file into std input
8.  # redirect std out to hosts
9.
10. sed '/'$1'/d' < /etc/temp_hosts > /etc/hosts
11.
12. # Remove temporary file
13. rm /etc/temp_hosts

```

`remRes.sh` removes the mapping from the operating system's `hosts` file. From this point forward all Grid jobs directed at the previously virtual resource would fail as the resource would not be contactable. The interface is updated to show that there are no virtual grid resources, displayed in figure 8.9.

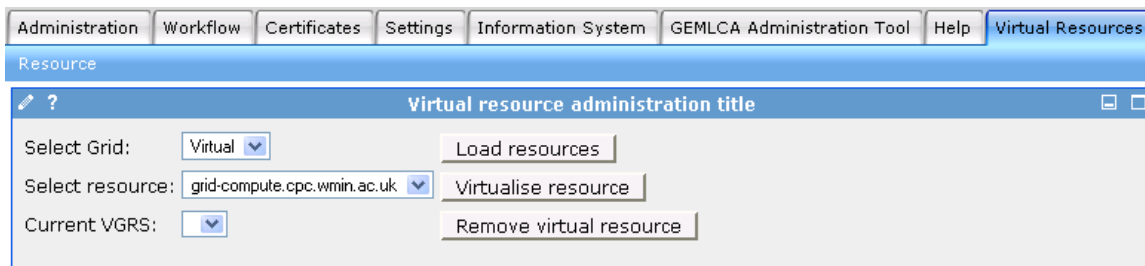


Figure 8.9: Screenshot of portlet view after the removal of a virtual Grid resource.

8.6 Solution configuration

A key element of the Grid teaching solution is a third shell script `initGridResource.sh`, listed here. `initGridResource.sh` is run as `root`. Its main tasks are:

- Create a temporary file to store the contents of the system's `hosts` file.
- Ensure the servlet container and any temporary files required run under the same user.
- Ensure the servlet container and any temporary files required run under the same group.
- Ensure the servlet container can read the system's `hosts` file.
- Ensure the servlet container can read from and write to any temporary files required.
- Modify the system's `hosts` files to ensure grid resources can be added.

```
1.  #!/bin/bash
2.  echo Please enter the group the web server runs under:
3.  read server_group
4.
5.  echo Please enter the user the web server runs under:
6.  read server_user
7.
8.  echo "Initializing...."
9.
10. # Allow server user to remove temporary files used
11. echo "Creating temporary files"
12. echo "/etc/temp_hosts"
13. touch /etc/temp_hosts
14. chown $server_user /etc/temp_hosts
15.
16. # Allow server group to read and write hosts file
17. echo "Modifying /etc/hosts access"
18. chgrp $server_group /etc/hosts
19. chmod ug+rw /etc/hosts
20.
21. # Allow server group to read and execute remove/add shell
    scripts
22. echo "Modifying access to /etc/addRes.sh"
23. chgrp $server_group /etc/addRes.sh
24. echo "Modifying access to /etc/remRes.sh"
25. chgrp $server_group /etc/remRes.sh
26.
27. # Allows grid resource urls to be placed correctly
28. echo "Preparing /etc/hosts"
29. echo '# vgrs' >> /etc/hosts
30.
31. echo "Done"
```

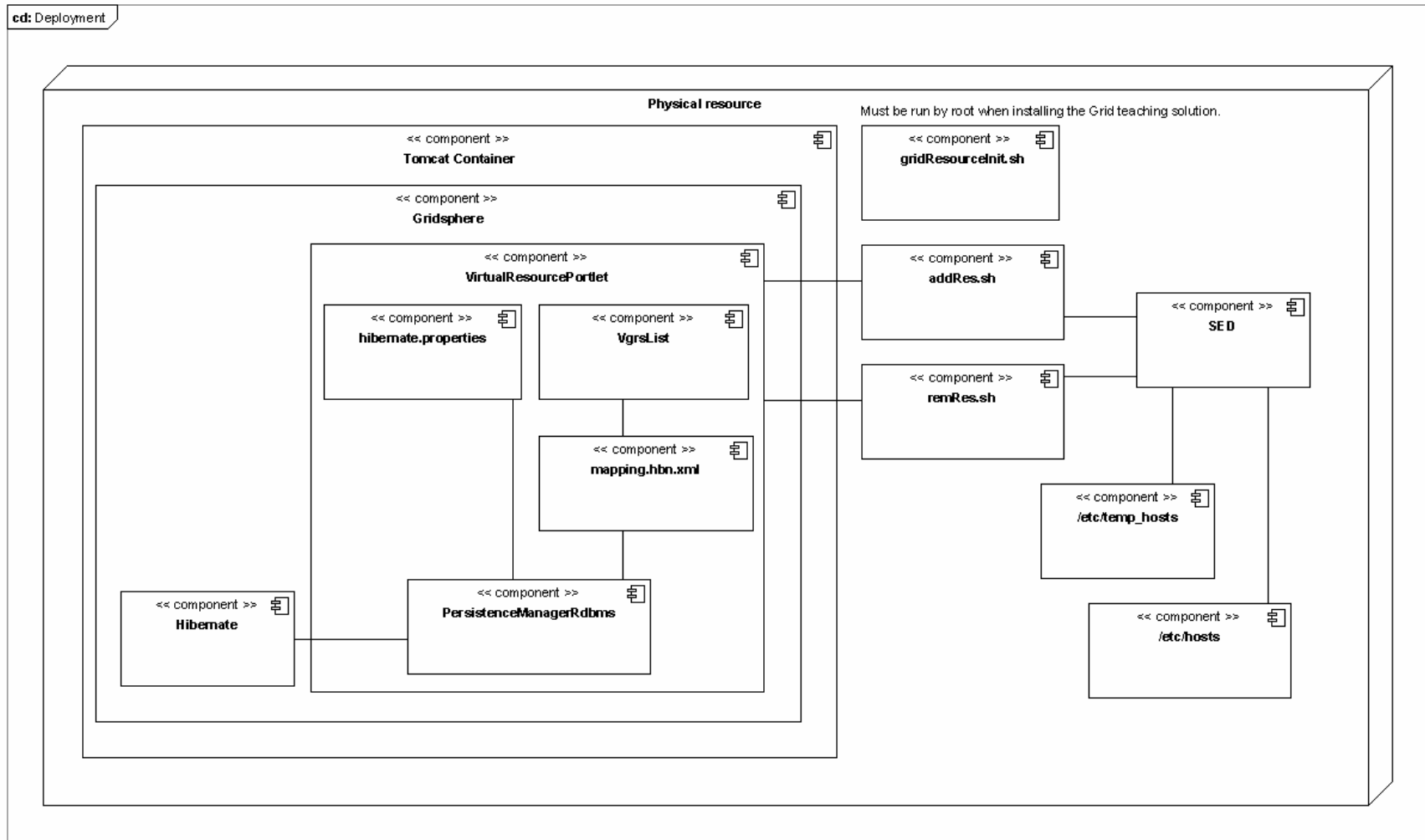


Figure 8.10: Grid teaching solution deployment diagram.

9.0 Testing

This chapter assesses how effectively the requirements (section 3.6) have been met. It does not assess the quality of the solution as a whole. The requirements responsible for satisfying use cases were tested ‘Add virtual resource’ and ‘Remove virtual resource’ were tested. The tests use Linux commands to ascertain the success of the test.

Identifier

TC9.1

Test case owner/creator

Jack Wootton

Name

Add virtual resource.

Requirement identifier

R2

Purpose

- Checks a Grid resource URL can be retrieved from a portlet view and appended, in the correct format, to the operating system’s `hosts` file.

Testing environment/configuration

- Operating system - Linux Suse
- Browser - Mozilla
- Portal Framework - Gridsphere

Initialization

- Functioning servlet container.
- `gridResourceInit.sh` has previously been run by root.
- At least one Grid configuration has been added.
- At least one Grid resource is attached to a Grid configuration.

Finalization

- Restart Tomcat container – ensures persistent storage is functioning correctly.

Actions

- Start portlet container
- Select Virtual Resources portlet.

- Load Grid resources attached to a Grid configuration.
- Select Grid resource to be virtualised.
- Click 'Virtualise Resource'.
- Login into Linux machine hosting portlet container.
- Run command 'cat /etc/hosts | grep <hostname>'.
- Run command 'ls | grep temp_hosts'

Input data

- Input values are provided by the portlet view, specifically the Java UI Beans.

Expected result

```
127.0.0.1    <hostname>
'' (empty string)
```

Actual results

```
127.0.0.1    grid-compute.oesc.ox.ac.uk
'' (empty string)
```

Completed successfully

Yes

Identifier

TC9.2

Test case owner/creator

Jack Wootton

Name

Remove virtual resource.

Requirement identifier

R3

Purpose

- Checks a Grid resource URL can removed from the operating system's `hosts` file.

Testing environment/configuration

- Operating system - Linux Suse
- Browser - Mozilla
- Portal Framework - Gridsphere

Initialization

- Test case TC9.1 must have completed successfully.

Finalization

- Restart Tomcat container – ensures persistent storage is functioning correctly.

Actions

- Start portlet container
- Select Virtual Resources from ‘current vgrs’ select list.
- Click ‘Remove virtual resource’ button.
- Run command ‘tail /etc/hosts’
- Run command ‘ls | grep temp_hosts’

Input data

- Input values are provided by the portlet view, specifically the Java UI Beans.

Expected result

```
# vgrs
```

Actual results

```
# vgrs
```

Completed successfully

```
Yes
```


10.0 Conclusion

This project has presented, implemented and documented the process of developing a solution to the problem of teaching Grid. A number of categories of solution were considered in designing an appropriate solution that would satisfy the core requirements of this research project, detailed in section 6.3. The procedures involved in the implementation of the chosen solution have been explained throughout this project, specifically in section 8.0. The final solution was created from a combination of existing Grid toolkits, distributed computing software and the Linux operating system. The solution fulfilled the objective of providing a safe and realistic Grid execution environment for teaching. Importantly knowledge acquired during use of the Grid teaching environment remains applicable and relevant to a live Grid environment, achieved through the selection of a Grid enabled web portal to provide an interface to the teaching environment.

The solution to Grid teaching, as this paper presents it, introduces the idea of virtual Grid resources and virtual Grids to Grid computing. Virtual Grids are specified as a collection Grid resource URLs that map to the IP address of a submit machine installed on a non live cluster, meaning the cluster not in use on a live Grid, such as the NGS [29]. A Linux hosts file was used to achieve this mapping. Importantly it is not a requirement that the solution use a cluster since both Condor and Condor G can be installed on a single machine. The terminology used may, perhaps, be confused with the use of virtualisation technologies in Grid computing, and for this reason should be changed so as to clearly distinguish between virtualisation software and the mapping of Grid resources to submit machines on a non live cluster (this project).

It has been shown how the configuration of freely available Grid technologies can provide a safe environment for teaching Grid. The deployment of the solution requires relatively small changes to existing Grid configurations, while achieving great flexibility when creating virtual Grids to which new grid users may safely submit jobs to. An advantage of the lightweight nature of the Grid teaching solution presented in this project is that large collections of virtual Grid resources can be created where there is no requirement that the same number of physical resources must exist.

More work is needed to assess the point at which the number of virtual resources impacts on the performance of the physical resource irrespective of whether the resource is a non live cluster or a single machine with Condor and Condor G installed. However this

project asserts that Grid jobs are likely to remain small within the context of teaching Grid since one of the primary objectives of teaching Grid is to provide means by which users interact with the system, understanding how jobs are submitted and results retrieved. Appreciating how long a Grid job may take to execute is, for the most part, irrelevant to new Grid users.

The success of the selected implementation can be assessed, initially, by the number of requirements that were met. Requirements R1-R8 (Tutor's functional requirements), R9-R12 (Student's functional requirements), R13-R14 (non functional requirements) and R15-R16 (interface requirements) were fulfilled. However in order to fully evaluate the appropriateness of the solution as well as the functioning of the software developed, a trial period would ideally be organised where by Students and Tutors could offer their evaluation of the teaching environment.

During the research, development and implementation stages of the Grid teaching solution, three extensions were identified as being significantly pertinent to the research conducted in this project. They were:

- **Simulation of queuing:** The current solution provides no mechanism by which a tutor may simulate queuing such that Grid jobs are delayed while the queue is processed. The current solution processes jobs as soon preceding jobs complete. In this respect the solution fails to mimic live Grid. Future work should focus on the implementation of a simulated submit queue which is under the Tutor's control.
- **Reliability:** The current solution suffers from a single point of failure. All virtual resources map to a single point of submission (the GRAM gatekeeper). If the point of submission fails, the entire teaching environment is rendered inoperable. It is proposed that multiple gatekeepers should be installed to increase reliability.
- **Globus toolkit 4:** The current solution implements the Globus Toolkit 2 (GT2). GT2 has been deprecated by the Globus Alliance. GT4 replaces it. The teaching environment should be extended to work with service oriented job submission.

11.0 References

- [1] A.Matsunaga et al. 2005. *On the Use of Virtualisation and Service Technologies to Enable Grid-Computing*.
- [2] Apache Ant Project - <http://ant.apache.org>
- [3] Apache Software Foundation - <http://tomcat.apache.org>
- [4] B.Wilkinson, C.Ferner. 2006. Teaching *Grid Computing in North Carolina: Part 1*.
- [5] Condor - <http://www.cs.wisc.edu/condor>
- [6] E.Kourpas. 2006. *Grid Computing: Past, Present and Future An Innovation Perspective*.
- [7] G.Andronico et al. 2003. *The GENIUS web portal: grid computing made easy*.
- [8] GILDA .2006. *GILDA: A point of entry to the EGEE Grid infrastructure*.
- [9] Globus Alliance Web Service Resource Framework Reference - <http://www.globus.org/wsrf/>
- [10] Globus Alliance - <http://www.globus.org>
- [11] Globus Toolkit 4 download - <http://www.globus.org/toolkit/downloads/4.0.3/>
- [12] GridLab Testbed Management - <http://www.gridlab.org/WorkPackages/wp-5/>
- [13] Gridsphere Portal Framework - www.gridsphere.org
- [14] I.Foster et al. 2001. *The Anatomy of the Grid Enabling Scalable Virtual Organisation*.

- [15] I.Foster et al. 2006. *Virtual Clusters for Grid Communities*.
- [16] I.Foster, C.Kesselman. 1999. *The grid, blueprint for a new computing*.
- [17] I.Foster, C.Kesselman. 2004. *The grid 2, blueprint for a new computing*.
- [18] I.Foster, D.Iamnitchi, 2003. *On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing*.
- [19] J.Joseph et al. 2004. *Evolution of grid computing architecture and grid adoption models*.
- [20] J.Novotny et al. 2003. *GridSphere: A Portal Framework for Building Collaborations*.
- [21] JSR 168: Portlet Specification - <http://jcp.org/en/jsr/detail?id=168>
- [22] K.Keahey et al, 2005. *Virtual Workspaces in the Grid*.
- [23] K.Keahey et al. 2003. *Dynamic Creation and Management of Runtime Environments in the Grid*.
- [24] K.Keahey et al. 2005. *Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid*.
- [25] M.P.Thomas et al. 2005. *Grid Portal Architectures for Scientific Application*.
- [26] Manchester University UK - www.man.ac.uk
- [27] M. Russell. 2006. *GridSphere's Grid Portlets*.
- [28] MSN - www.msn.com
- [29] National Grid Service - <http://www.ngs.ac.uk>
- [30] National Grid Service GSI-SSHTerm Application - <http://www.grid-support.ac.uk/content/view/81/61/>
- [31] Parallel Grid Run-time and Application Development Environment - www.lpds.sztaki.hu/pgrade
- [32] R.Buyya. 2006. *GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing*.
- [33] R.J.Walters et al, 2006. *Teaching the Grid: Learning Distributed Computing with the M-Grid Framework*.

[34] X.Zhang et al. 2005, *Virtual Cluster Workspaces for Grid Applications*.

[35] Yahoo! - www.yahoo.com

[36] Xeo - <http://www.exoplatform.com/company/faces/public/company>

[37] The Xen virtual machine monitor - <http://www.cl.cam.ac.uk/research/srg/netos/xen/>

[38] Y.Yutaka. 2003. *Java™ Servlet Specification Version 2.4*.

Appendix A

Source code

Page 66 `gridResourceInit.sh`

Page 67 `addRes.sh`

Page 67 `remRes.sh`

Page 68 `Resource.java`

Page 75 `VgrsList.java`

Page 76 `Resource.jsp`

gridResourceInit.sh

```
#!/bin/bash
echo Please enter the group the web server runs under:
read server_group

echo Please enter the user the web server runs under:
read server_user

echo "Initializing...."

# Allow server user to remove temporary files used
echo "Creating temporary files"
echo "/etc/temp_hosts"
touch /etc/temp_hosts
chown $server_user /etc/temp_hosts

# Allow server group to read and write hosts file
echo "Modifying /etc/hosts access"
chgrp $server_group /etc/hosts
chmod ug+rw /etc/hosts

# Allow server group to read and execute remove/add shell scripts
echo "Modifying access to /etc/addRes.sh"
chgrp $server_group /etc/addRes.sh
echo "Modifying access to /etc/remRes.sh"
chgrp $server_group /etc/remRes.sh

# Allows grid resource urls to be placed correctly
echo "Preparing /etc/hosts"
echo '# vgrs' >> /etc/hosts

echo "Done"
```

remRes.sh

```
#!/bin/bash

# Create temporary hosts file, because sed overwrites buffer
cat /etc/hosts > /etc/temp_hosts

# Use sed to remove all occurrences of the command line URL
# Direct temporary hosts file into std input
# redirect std out to hosts

sed '/'$1'/d' < /etc/temp_hosts > /etc/hosts

# Remove temporary file
rm /etc/temp_hosts
```

addRes.sh

```
#!/bin/bash

# Create temporary hosts file, because sed overwrites buffer
cat /etc/hosts > /etc/temp_hosts

# Insert loopback IP, a space, and command line argument
# Use temporary hosts file as input
# Redirect std out to hosts

sed -e '/vgrs/a\127.0.0.1      '$1 < /etc/temp_hosts > /etc/hosts

# Clean up afterwards
rm /etc/temp_hosts
```

Resource.java

```
package org.gridlab.gridsphere.gsexamples.portlets;

/* Portlet */
import javax.portlet.*;
import org.gridlab.gridsphere.provider.portlet.jsr.ActionPortlet;
import org.gridlab.gridsphere.provider.event.jsr.RenderFormEvent;
import org.gridlab.gridsphere.provider.event.jsr.ActionFormEvent;

/* Storage */
import org.gridlab.gridsphere.gsexamples.services.VgrsList;
import
org.gridlab.gridsphere.core.persistence.PersistenceManagerRdbms;
import
org.gridlab.gridsphere.core.persistence.PersistenceManagerFactory
;

/* Grid configs */
import hu.sztaki.lpds.pgportal.services.pgrade.GridConfigs;
import
hu.sztaki.lpds.pgportal.portlets.pgrade.PGradeSettingsBean;
import
hu.sztaki.lpds.pgportal.services.pgrade.GridJobManagerConfigs;
import
hu.sztaki.lpds.pgportal.services.pgrade.SZGJobManagerConfiguratio
n;

/* UI */
import
org.gridlab.gridsphere.provider.portletui.beans.ListBoxBean;
import
org.gridlab.gridsphere.provider.portletui.beans.ListBoxItemBean;
import org.gridlab.gridsphere.provider.portletui.beans.TextBean;
import
org.gridlab.gridsphere.provider.portletui.beans.TextFieldBean;

import java.util.Map;

public class Resource extends ActionPortlet
{
    /* Display */
    private static final String DISPLAY_PAGE = "resource.jsp";

    /* Shell script execution */
    private Runtime runtime;
    private Process process;
    private String cmd;
```

```

    /* Sotrage */
    private PersistenceManagerRdbms pm;
    private VgrsList vgrsList;

    public void init(PortletConfig config) throws
PortletException
    {
        /* Basic initialisation */
        super.init(config);
        DEFAULT_VIEW_PAGE = "prepare";
        runtime = Runtime.getRuntime();
        process = null;
        cmd = "";

        /* Create storage for Virtual Grid Resources object */
        pm =
PersistenceManagerFactory.createPersistenceManagerRdbms("resource
s");

        vgrsList = new VgrsList();
        vgrsList.setUrl("");

        try
        {
            /* Actually create it */
            pm.create(vgrsList);
        }
        catch (Exception e)
        {
            //
        }
    }

    public void prepare(RenderFormEvent event) throws
PortletException
    {
        /****** Start *****/

        PortletRequest req = event.getRenderRequest();

        /****** Grid resources
        *****/

        String[] availableGrids = GridConfigs.getGridNames();
        ListBoxBean gridsListBox =
event.getListBoxBean("gridsListBox");
        gridsListBox.clear();

        for( int i=0;i< availableGrids.length; i++)

```

```

        {
            ListBoxItemBean item = new ListBoxItemBean();
            item.setValue(availableGrids[i]);
            gridsListBox.addBean(item);
        }

        /***** VGRS resources
        *****/

        ListBoxBean vgrsListBox =
event.getListBoxBean("vgrsListBox");
        vgrsListBox.clear();

        try
        {
            VgrsList vgrsList = (VgrsList) pm.restore("from "
+ VgrsList.class.getName());
        }
        catch (Throwable T)
        {
        }

        String currentUrls = vgrsList.getUrl();
        String[] arrayUrls = currentUrls.split(",");

        for (int i= 0;i<arrayUrls.length ; i++)
        {
            ListBoxItemBean item = new ListBoxItemBean();
            item.setValue(arrayUrls[i]);
            vgrsListBox.addBean(item);
        }

        /***** Done *****/

        setNextState(req, DISPLAY_PAGE);
    }

    public void loadResources(ActionFormEvent event) throws
PortletException
    {
        ActionRequest req = event.getActionRequest();

        ListBoxBean gridsListBox =
event.getListBoxBean("gridsListBox");
        ListBoxBean resourcesListBox =
event.getListBoxBean("resourcesListBox");
        resourcesListBox.clear();

        String gridName = gridsListBox.getSelectedValue();

```

```

        String username = getUsername(req);

        PGradeSettingsBean pgsb = new
PGradeSettingsBean(username, gridName);
        SZGJobManagerConfiguration[] jmConfigs =
pgsb.getJMConfigurations();

        for( int i = 0; i < jmConfigs.length; i++)
        {
            ListBoxItemBean item = new ListBoxItemBean();
            item.setValue(jmConfigs[i].getContactString());
            resourcesListBox.addBean(item);
        }

        setNextState(event.getActionRequest(), DISPLAY_PAGE);
    }

    private String getUsername(PortletRequest request)
    {
        Map userInfo = (Map)
request.getAttribute(PortletRequest.USER_INFO);
        return (String) userInfo.get("user.name");
    }

    public void add(ActionFormEvent event) throws
PortletException
    {
        ActionRequest req = event.getActionRequest();

        /* Retrieve gui elements */
        ListBoxBean resourcesListBox =
event.getListBoxBean("resourcesListBox");
        String resourceName =
resourcesListBox.getSelectedValue();

        /* Check the user actually entered a value */
        if (!(resourceName.equals("")) )
        {
            /* Build the command */
            cmd = ("/etc/addRes.sh " + resourceName);

            try
            {
                /* Actually execute the shell script */
                process = runtime.exec(cmd);

                /* Load existing resource urls */
                vgrsList = (VgrsList) pm.restore("from " +
VgrsList.class.getName());
                String currentUrls = vgrsList.getUrl();
            }
        }
    }

```

```

        /* Add new resource url */
        currentUrls += resourceName + ",";
        vgrsList.setUrl(currentUrls);

        /* Actually update */
        pm.update(vgrsList);
    }
    catch(Throwable t)
    {
        //
    }
}

/***** VGRS resources *****/

    ListBoxBean vgrsListBox =
event.getListBoxBean("vgrsListBox");
    vgrsListBox.clear();

    try
    {
        VgrsList vgrsList = (VgrsList) pm.restore("from "
+ VgrsList.class.getName());
    }
    catch (Throwable T)
    {
    }

    String currentUrls = vgrsList.getUrl();
    String[] arrayUrls = currentUrls.split(",");

    for (int i= 0;i<arrayUrls.length ; i++)
    {
        ListBoxItemBean item = new ListBoxItemBean();
        item.setValue(arrayUrls[i]);
        vgrsListBox.addBean(item);
    }

    /* Display the page */
    setNextState(req, DISPLAY_PAGE);
}

    public void remove(ActionFormEvent event) throws
PortletException
    {
        ActionRequest req = event.getActionRequest();

        /* Retrieve gui elements */

```

```

        ListBoxBean vgrsListBox =
event.getListBoxBean("vgrsListBox");
        String resourceName = vgrsListBox.getSelectedValue();

        /* Check the user actually entered a value */
        if (!(resourceName.equals(""))))
        {
            /* Build the command */
            cmd = ("/etc/remRes.sh " + resourceName);
            try
            {
                /* Actually execute the shell script */
                process = runtime.exec(cmd);

                /* Load existing resource urls */
                VgrsList vgrsList = (VgrsList)
pm.restore("from " + VgrsList.class.getName());
                String currentUrls = vgrsList.getUrl();

                /* To store new url list */
                String updatedList = "";
                String[] arrayUrls =
currentUrls.split(",");

                /* Create new resource url list, excluding
value entered by user */
                for (int i= 0;i<arrayUrls.length ; i++)
                {
                    if (!
(resourceName.equals(arrayUrls[i])))
                    {
                        updatedList += arrayUrls[i] +
", ";
                    }
                }

                vgrsList.setUrl(updatedList);

                /* Actually update */
                pm.update(vgrsList);
            }
            catch(Throwable t)
            {
                //
            }
        }

        /***** VGRS resources
        *****/

        vgrsListBox.clear();

```

```
        try
        {
            VgrsList vgrsList = (VgrsList) pm.restore("from "
+ VgrsList.class.getName());
        }
        catch (Throwable T)
        {
        }

String currentUrls = vgrsList.getUrl();
String[] arrayUrls = currentUrls.split(",");

for (int i= 0;i<arrayUrls.length ; i++)
    {
        ListBoxItemBean item = new ListBoxItemBean();
        item.setValue(arrayUrls[i]);
        vgrsListBox.addBean(item);
    }

    /* Display the page */
    setNextState(req, DISPLAY_PAGE);
}
}
```

VgrsList.java

```
package org.gridsphere.gsexamples.services;

public class VgrsList
{
    private String oid = null;
    private String url = new String();

    public String getOid()
    {
        return oid;
    }

    public void setOid(String oid)
    {
        this.oid = oid;
    }

    public String getUrl()
    {
        return url;
    }

    public void setUrl(String url)
    {
        this.url = url;
    }
}
```

resource.jsp

```

<%@ taglib uri="/portletUI" prefix ="ui" %>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
<portlet:defineObjects />
<ui:form>
  <ui:table>
    <ui:tablerow>
      <ui:tablecell>Select Grid:</ui:tablecell>
      <ui:tablecell><ui:listbox
beanId="gridsListBox"></ui:listbox></ui:tablecell>
      <ui:tablecell><ui:actionssubmit
action="loadResources" value="Load resources"/></ui:tablecell>
    </ui:tablerow>
    <ui:tablerow>
      <ui:tablecell>Select resource:</ui:tablecell>
      <ui:tablecell><ui:listbox
beanId="resourcesListBox"></ui:listbox></ui:tablecell>
      <ui:tablecell><ui:actionssubmit action="add"
value="Virtualise resource"/></ui:tablecell>
    </ui:tablerow>
    <ui:tablerow>
      <ui:tablecell>Current VGRS:</ui:tablecell>
      <ui:tablecell><ui:listbox
beanId="vgrsListBox"></ui:listbox></ui:tablecell>
      <ui:tablecell><ui:actionssubmit action="remove"
value ="Remove virtual resource"/></ui:tablecell>
    </ui:tablerow>
  </ui:table>
</ui:form>

```