# Building Computational Communities from Federated Resources*

Nathalie Furmento, Steven Newhouse, and John Darlington

Parallel Software Group, Department of Computing, Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
icpc-sw@doc.ic.ac.uk
http://www-icpc.doc.ic.ac.uk/components/

**Abstract.** We describe the design and the implementation in Java and Jini of a *Computational Community*, which supports the federation of resources from different organisations. Resources from the local *Administrative Domain* are published in a Jini space to form a *Computational Community*. Different access control policies can be applied to the same resource in different *Computational Communities*. We show how this architecture can be extended through the addition of an *Application Mapper* and *Resource Broker* to build a computational economy.

**Keywords:** Computational Community, Computational Economy, Grid Computing, Distributed Systems.

## 1 Introduction

The accelerating proliferation of computing resources together with the rapid expansion of high speed connection networks has increased the interest of users in computational grids. A computational grid is defined as a combination of geographically distributed heterogeneous hardware and software resources that provide a ubiquitous computation environment [1]. The motivation behind computational grids is to deliver computational power to a user's application in a similar way the electrical power is delivered to an electrical appliance. This can only be achieved through integration between the resources and the application.

As no single organisation is able to provide all the resources in such an infrastructure, it is inevitable that the grid will be composed of federated resources. However, organisations will only be willing to contribute their resources to the grid if they retain ultimate control of how and when they are used. Therefore, any grid middleware must not only provide strong authentication mechanisms suitable for a distributed computing environment, but must also support sophisticated access control policies capable of differing between individuals, groups and

* Research supported by the EPSRC grant GR/N13371/01 on equipment provided by the HEFCE/JREI grants GR/L26100 and GR/M92455

R. Sakellariou et al. (Eds.): Euro-Par 2001, LNCS 2150, pp. 855–863, 2001.

organisations. Resource federation is more likely to take place between organisations if there is mutual self interest or a natural inter-organisational grouping, e.g. to build a *Computational Community* of particle physics users.

Ensuring effective utilisation of these distributed federated resources is a challenge to both users and resource providers. It is essential that any *Computational Community* is able to balance the computational demand and supply of its users and resource providers. We will show how our *Computational Community*, built from federated resources, can be extended through market mechanisms to a computational economy where resources are traded between providers and consumers to ensure effective resource utilisation.

*Roadmap.* This paper will present our vision of a *Computational Community* and how it can be extended to form a computational economy. Section 2 gives a global overview of the *Computational Community* middleware as well as related work. Sections 3 and 4 give further details of the abstractions in the middleware. Before concluding, we describe in Section 5 how the *Computational Community* can be extended through higher level services.

## 2 Building a Computational Community

### 2.1 Concepts

Our middleware (see Fig. 1) consists of: (1) a private *Administrative Domain* that allows the local administrator to manage the resources of their organisation (see §3), (2) a *Domain Manager* that acts as a conduit between the private and public areas of the infrastructure. It annotates publishable resources in the private *Administrative Domain* with access control information, and authenticates and authorises incoming requests to use these local resources (see §4), and (3) a public *Computational Community* where the information about the resources and how they can be used are made available to the users (see §5).

The *Computational Community* is built from potentially diverse computational, storage and software resources that have both static and dynamic attributes. These resources are managed through the local *Administrative Domain*. The *Domain Manager* is used by the local administrator to publish the resources from the *Administrative Domain* in any number of public *Computational Communities*. The *Identity Manager* is trusted by the *Domain Manager* to authenticate the identity of the users wishing to access the local resources and is administered by the local administrator.

Three management tools – the *Resource Manager*, the *Policy Manager* and the *Resource Browser* – allow both users and administrators to interact with the framework through a graphical interface.

### 2.2 Prototype

A prototype of this architecture has been completed using a Java and Jini environment. We exploit the cross-platform portability of Java [2] with its diverse

class libraries to simplify many of the development tasks. Jini [3] is used as the primary service infrastructure of the architecture as it has several desirable features for a wide-area grid environment. It supports dynamic registration, lookup and connection between the Java objects which represent our grid resources. We use a Jini lookup server to implement the public *Computational Communities* and the private *Administrative Domains* defined in our architecture.

Our middleware could be implemented using technologies other than Java and Jini. For example, the private domain could use a LDAP server to maintain a register of the available resources [4].

### 2.3 Related Work

Our approach is a combination and a logical extension of two major grid infrastructure projects: (1) Globus which provides a toolkit of services (information management, security, communication etc.) to integrate heterogeneous computational resources into a single infrastructure [5], and (2) Legion which uses a uniform object model for both applications and resources allowing users and administrators to subclass generic interfaces to their specific local needs [6].

Our initial implementation indicates that the Java/Jini combination is capable of providing an extensible fault tolerant distributed infrastructure for grid computing. Other projects have also demonstrated the effectiveness of Jini in providing a grid middleware [7] and the use of Java to provide a homogeneous distributed computing environment across heterogeneous resources [8].

## 3 Local Resources

### 3.1 Overview

Resources within an organisation are managed by a local administrator. Once started, the *Administrative Domain* (a Jini space) maintains a list of the cur-

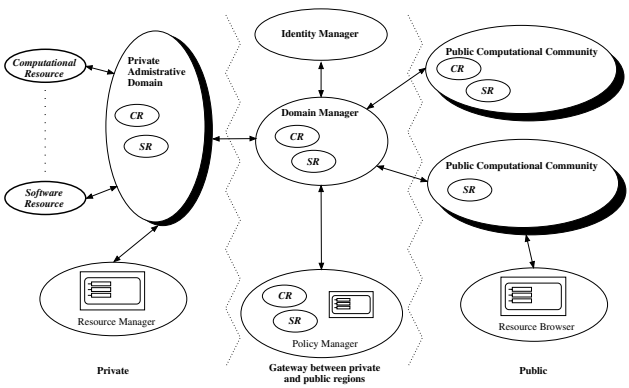


**Fig. 1.** Building a *Computational Community* through Federated Resources

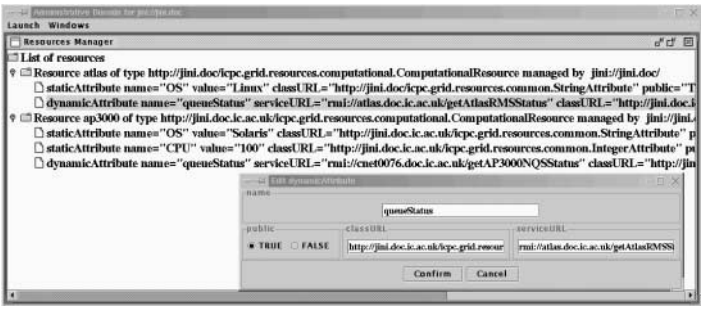


**Fig. 2.** The *Resource Manager*

rently available resources. These resources are monitored by the *Resource Manager* (see Fig. 2), which is notified when a resource enters or leaves the Jini space. The *Resource Manager* allows the administrator to alter the resources' configuration by adding, modifying or removing attributes. An XML scheme and configuration file (maintained on disc to ensure persistence) describes the resources (see §3.2).

We are currently concentrating on three different resources types:

- **Computational Resources.** We access our own local computational hardware through a batch scheduler abstraction with implementations for NQS, PBS [9] and Condor [10]. Each computational resource executes its own segment of an XML defined execution plan passed to it by the *Domain Manager*.
- **Storage Resources.** The user must be able to access their storage space from any resource. This allows input and output files to be transfered to the execution location. Read and write access policies are defined to allow authenticated individuals, groups or organisations to use their file space.
- **Software Resources.** Our current implementation only represents unlimited-use software libraries but the execution of a licensed library or application has to be scheduled in the same manner as a computational resource to ensure that a licence is available.

### 3.2 Implementation

A resource is defined by its *name*, a Jini space into which it *publishes* its availability (the *Administrative Domain*) and a Java class (its *type*) that extends the appropriate `Resource` class. The `execute` method in the `Resource` class accepts an XML file describing the task that is to be performed on the resource.

The resource can have any number of static and dynamic attributes which are encapsulated in the published service class. An attribute can be static (i.e. initialised once at the beginning of the execution) or dynamic (i.e. periodically updated during the execution). A static attribute is defined by its name, a value, and the location of the defining Java class. A dynamic attribute is defined by its

name, the URL of the service used to update the attribute (e.g. Remote Method Invocation), and the location of the defining Java class. A *public* attribute is visible to the whole *Computational Community*, while a *non-public* (i.e. private) attribute is only visible to the *Administrative Domain* and the *Domain Manager*.

*Example.* The XML configuration file defining a computational resource (the AP3000) is shown below and in Fig. 2. It has two static attributes (only one of which is public) and a dynamic attribute updated through a RMI service.

```
<resource name="ap3000" publish="jini://jini.doc/"
    type="http://jini.doc/icpc.grid. ... .ComputationalResource">
  <staticAttribute name="OS" value="Solaris" public="TRUE" classURL="http://jini.doc..."/>
  <staticAttribute name="CPU" value="100" public="FALSE" classURL="http://jini.doc..."/>
  <dynamicAttribute name="queueStatus" public="TRUE" classURL="http://jini.doc..."
    serviceURL="rmi://cnet0076.doc/getAP3000NQSStatus"/>
</resource>
```

## 4 Resource Federation

### 4.1 Overview

The *Domain Manager* is the sole route between the private *Administrative Domain* containing the local resources of an organisation and the *Computational Community* in which the resource is published. Its role is to enforce the access control policies defined by the administrator for users in each of the *Computational Communities*. It is also able to restrict the published information in order to hide a specific resource, an attribute of a specific resource (the resource itself can decide which attributes are publicly visible), or the details of the access policy from a particular *Computational Community*.

When a new resource becomes available in the *Administrative Domain*, it is automatically published in the appropriate *Computational Community* if the *Domain Manager* already has an entry for it in its configuration file (see below). If there is no record in the configuration file of the resource for a specific *Computational Community*, then the administrator can use the *Policy Manager* to define where the resource should be published.

```
<domainManager manage="jini://jini.doc/" name="icpc">
 <promote domain="jini://trident.doc">
  <!-- the list of resources                                                    -->
   <resource name="condor"/>
   <!-- this resource ap3000 got some specific access permission               -->
   <resource name="ap3000">
    <deny stopDay="monday"> <entity type="person" name="nfurSigned"/> </deny>
   </resource>
   <!-- default access control policies for the domain                         -->
   <deny startDay="saturday" stopDay="sunday"> <entity type="group" name="ra"/> </deny>
   <allow startDay="tuesday"> <entity type="organization" name="ICPC"/> </allow>
 </promote>
 <promote domain="jini://ariane.doc"> ... </promote>
</domainManager>
```

### 4.2 Access Control

Access to individual resources is controlled through conventional access control lists based on the following entities: individuals, groups and organisations. An organisation is composed of any number of groups, each containing any number of individuals. The *Domain Manager* is able to implement fine-grained access control policies relating to the resources in the local *Administrative Domain*.

The access control policy is qualified through a time interval, defined by a start time (by default, Monday at 00:00) and a stop time (by default, Sunday at 23:59). The access policies of an organisation are also applied to all the groups inside the organisation. Similarly, the access policies of a group are applied to all the individuals inside the group.

The *Domain Manager* configuration file allows an administrator to define the default access control policies for a *Computational Community*, which will be used for all the resources published within this community. Additional access policies can also be defined for specific resources as they are required.

### 4.3 The Management Tools

The *Policy Manager* allows an organisation's administrator to define the behaviour of the *Domain Manager* and therefore how the organisation's resources are contributed into the *Computational Communities* (see Fig. 3). It shows the available resources and the communities in which they are published. For each resource promoted in a specific community, it is possible to modify the access control policies, every modification will be propagated from the *Domain Manager* to the corresponding *Computational Community*.

*Example.* In Fig. 3, the AP3000 is published in the *Computational Community* hosted by `jini://trident.doc`. The default access policies of the community

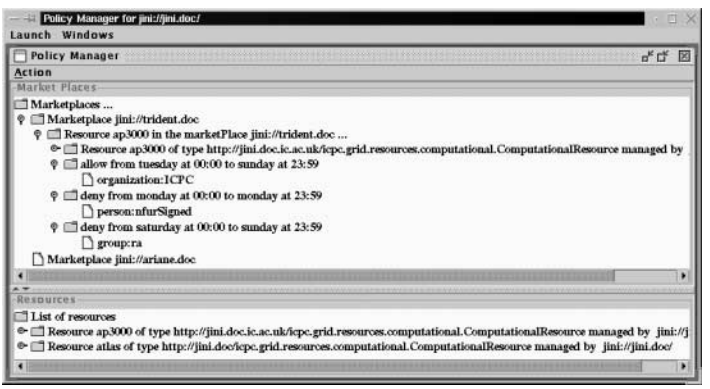


**Fig. 3.** The *Policy Manager*

are extended by a resource specific access policy. Note, that the non-public resource attribute, the CPU, is no longer visible. The *Domain Manager* retains knowledge of this attribute but it is not available outside the private domain.

The *Identity Manager* contains the X.509 certificates [11] (used by our public key authentication infrastructure) of the trusted organisations and their associated groups. An organisation acts as a certification authority for the individuals that are members of that organisation. All individuals using a resource within a domain have to belong to one of the trusted organisations. Global access control policies can also be defined within the *Identity Manager*. For instance, a particular user or group of users could be barred from all of the resources in an *Administrative Domain*.

## 5 The Computational Community

The *Resource Browser* (see Fig. 4) allows the users to examine the usage policies and the attributes of the accessible resources in a particular *Computational Community*. Dynamic attributes are updated periodically or on request.

*Example.* The *Computational Community* shown in Fig. 4 is hosted by `jini://trident.doc` and currently contains a computational resource, the AP-3000. The access control policies and attributes match those defined by the *Policy Manager* and *Resource Manager* respectively. Fig. 4 also shows the current status of the batch queuing system on the AP3000.

The *Computational Community* can be used to provide information to high level services, e.g. to improve job turnaround and increase resource utilisation. We will enhance the usability of the *Computational Community* to provide automatic or semi-automatic resource selection by using meta-data relating to both the performance of the application on different resources and the user's requirements.

A related project within our group is to define an application as a composition of software components annotated with implementation and performance information [12,13]. This meta-data, together with the user's requirement, will

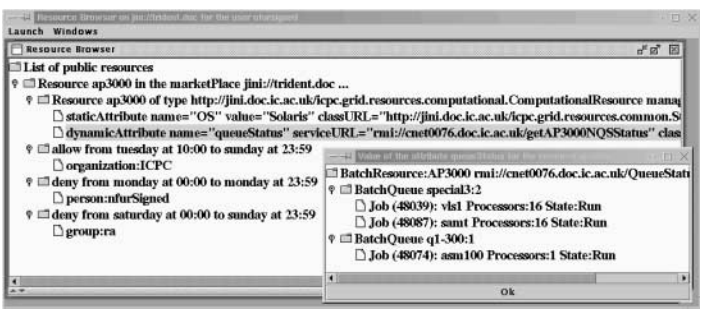


**Fig. 4.** The *Resource Browser*

be used by: (1) an *Application Mapper* to select the most effective implementations by utilising the application's knowledge, and (2) a *Resource Broker* to optimise the global job mix through computational economics [14,15].

## 6 Conclusion

Computational communities will eventually, like the Internet, change the way we work. However, to effectively exploit the computational potential of the grid, we need to balance the needs of the users, their applications and the resource providers, in order to deploy an application to a resource that will satisfy the stated requirements of both the user and the resource provider.

As we have shown, our *Computational Community* can make an efficient use of this information to enable us to build application mappers to effectively match applications to resources and brokers to make the best economic use of the available resources. We are therefore able to address some of the weaknesses in current grid infrastructures.

## Acknowledgements

We would like to thank Keith Sephton, the Imperial College Parallel Computing Centre's Systems Manager, for his help with this work.

## References

1. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998. 855
2. K. Arnold, J. Gosling, and D. Holmes. *The Java Programming Language*. Addison-Wesley, 3rd edition, 2000. 856
3. Sun Microsystems. Jini(tm) Network Technology. `http://www.sun.com/jini/`. 857
4. S. Fitzgerald and I. Foster *et al.* A Directory Service for Configuring High-Performance Distributed Computations. In *6th IEEE Symp. on High-Performance Distributed Computing*, pages 365–375, 1997. 857
5. I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997. 857
6. A. S. Grimshaw and W. A. Wulf *et al.* The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1):39–45, 1997. 857
7. Z. Juhasz and L. Kesmarki. A Jini-Based Prototype Metacomputing Framework. In *Euro-Par 2000*, volume 1900 of *LNCS*, pages 1171–1174, 2000. 857
8. M. O. Neary, B. O. Christiansen, P. Cappello, and K. E. Schauser. Javelin: Parallel Computing on the Internet. In *Future Generation Computer Systems*, volume 15, pages 659–674. Elsevier Science, 1999. 857
9. Veridian Systems. Portable Batch Systems. `http://www.openpbs.org`. 858
10. Condor Team. Condor Project Homepage. `http://www.cs.wisc.edu/condor`. 858

11. Sun Microsystems. X.509 certificates. http://java.sun.com/products/jdk/1.2/docs/guide/security/cert3.html, 1998. 861
12. S. Newhouse, A. Mayer, and J. Darlington. A Software Architecture for HPC Grid Applications. In *Euro-Par 2000*, volume 1900 of *LNCS*, pages 686–689, 2000. 861
13. N. Furmento, A. Mayer, S. McGough, S. Newhouse, and J. Darlington. A Component Framework for HPC Applications. Accepted for Euro-Par 2001. 861
14. C. A. Walspurger and T. Hogg *et al.* Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992. 862
15. R. Buyya and S. Chapin. Architectural Models for Resource Management in the Grid. In *Grid 2000*, volume 1971 of *LNCS*, 2000. 862