

Resource Brokering: The EUROGRID/GRIP Approach

John Brooke, Donal Fellows, Jon MacLaren

University of Manchester, Oxford Road, Manchester

{john.brooke,donal.fellows,jon.maclaren}@man.ac.uk

Abstract

This paper describes the Resource Discovery and Brokering system used within the UNICORE Grid System and describes how it may be used for selection of resources within a general computational grid. By having a focus on describing systems in terms of a simple standardized information model, it is possible to quickly deploy a UNICORE grid engine on a resource provider while allowing external agents to understand what software is available and what system resources are available to run that software. Through work done at the University of Manchester, an application-specific brokering system may be easily set up on top of an existing computational grid, with users (or their front-end application interfaces) describing their application requirements in terms that make sense within the context of the application and which are translated into the resource limits understood by underlying batch queuing subsystems like LSF or PBS.

Introduction

As Grids begin to grow beyond single-organisation or (single-VO) boundaries, new components and techniques will be required to keep the Grid usable. Resource Brokering, which enables the efficient discovery of appropriate resources, is an essential part of any scalable Grid. We describe a Resource Broker component capable of finding resources on both UNICORE[UNI] and Globus Toolkit[Glo] based Grids, and which is currently being proposed as a component for inclusion in the UK National eScience Grid.

In general, resource brokering has several significant aspects:

- The task must be described.
- The resources available must be described.
- The task and resources must be matched up with each other, ideally with some guarantee of a level of Quality-of-Service (QoS).
- The “most optimal” matching of the task to its implementing resources must be discovered.

The EUROGRID[EUR]/GRIP[GRI] Resource Broker, developed at the University of Manchester, provides a solution to the first three parts of this problem within the context of the UNICORE Grid Architecture, and work is ongoing on development of a scheme for

automatically selecting offers based on the QoS information provided.

Describing the User Task

In the UNICORE Resource Model, users describe their task in terms of capabilities and capacities. A capability states that something must be present (for example an application, a library, a software license or the ability to run things under an environment such as MPI), and a capacity describes an amount of some resource that is required (such as how many processors, how much memory, the amount of temporary disk space, some level of network bandwidth).

However, the UNICORE Resource Model allows for even richer descriptions than that, since application resources may include metadata that describes the user’s configuration of the application further. This gives a resource broker access to user-oriented information about the task, such as the number of grid points desired in a weather model or the angular separation between views in a 3D visualisation, and allows better estimation of the cost of performing the task on various machine models. By getting better estimates of problem size and the specifics of what the problems are, better turnaround-time estimates and pricing offers can be made.

This also encourages users to provide better descriptions of their problems to the job management subsystems. With traditional batch queues, users tend to just request resources up to just under the limit for a particular queue. This

habit of users makes it more difficult to do advanced resource management; it is difficult to tell whether a small job can be fitted in before some pre-booked time-slot because although there is a high chance that the actual job will fit into the space available, the amount of time actually requested may well overrun significantly. By using the applications' own scaling estimates (and hence better estimates of the actual size of jobs) it becomes much easier to do efficient resource allocation with an aim to maximizing utilization.

Describing the Available Resources

In the UNICORE Resource Model, it is not the size of systems that are generally described, but instead the acceptable range of values that a particular resource on a system may take (together with a default for when the user does not specify a value.) This is a very deep difference as it means that UNICORE describes the policies associated with a site instead of the physical configuration of a site; these may differ substantially on production supercomputers such as CSAR or HPCX.

Thus, a particular batch queue might be described (using a Capacity Resource) as

permitting jobs of between 10MB and 1GB in size, with the default amount of space available being 100MB. Where a system is describing capabilities (such as the available software packages) it merely states those capabilities which it knows it has. Although there are conceivably infinitely many capacities and capabilities available, a system will only ever describe those that it understands and supports.

Matching User Tasks and Resources

The process of Resource Brokering is the matching of abstracted user tasks to concrete resources capable of executing these tasks. The EUROGRID broker (see Figure 1) works by taking the set of resources requested and testing whether those resources are available at a particular site. Once availability has been established, the broker then contacts the site to obtain a QoS offer for that particular configuration of task.

However, a broker may also be configured to delegate the checking of resource availability and obtaining of QoS offers to other brokering agents. This delegation mechanism allows whole networks of brokering agents to be quickly built (see Figure 2), allowing a request

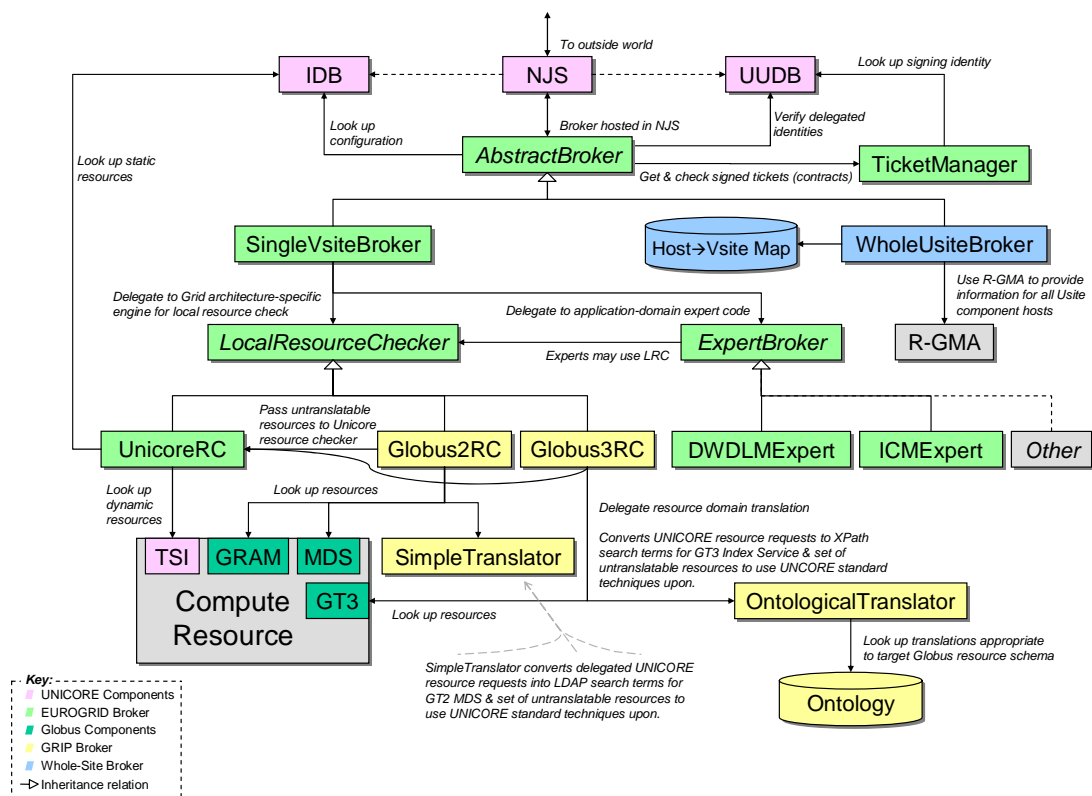


Figure 1: Resource Broker Internal Architecture

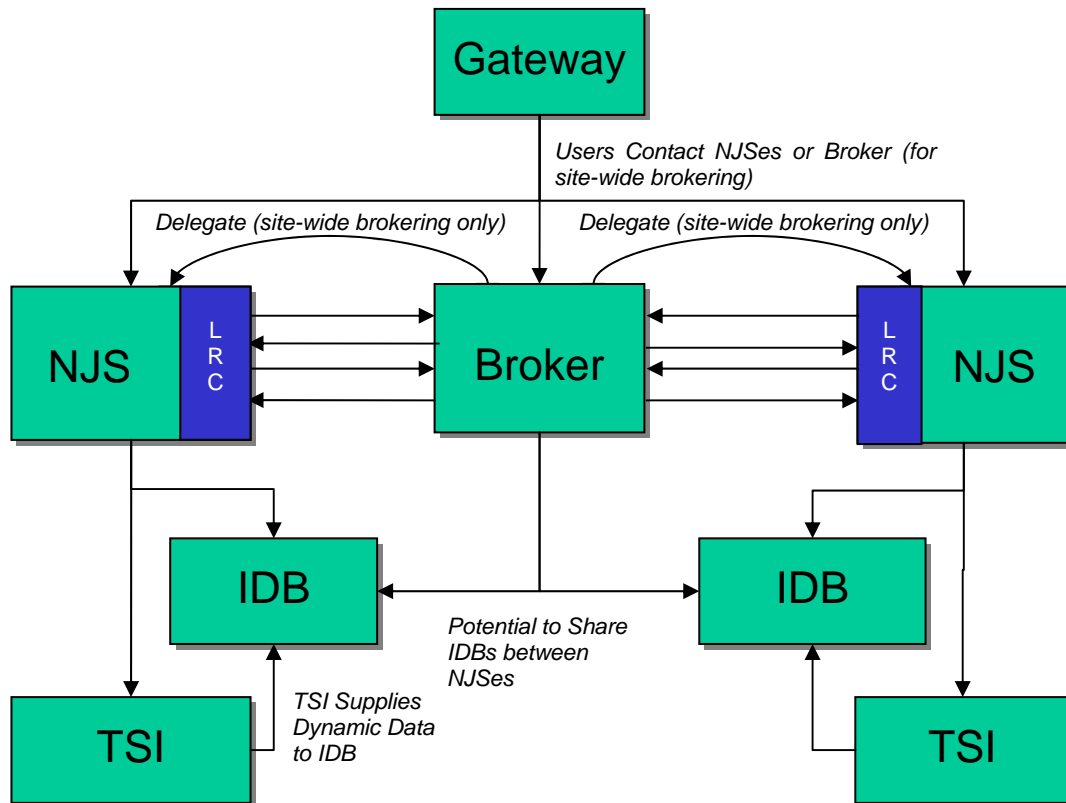


Figure 2: UNICORE Brokered Site Architecture

to a single broker to be converted into a search of all resources within a virtual organization. We also allow the issuing of invalid QoS offers; these act as advertisements, e.g. allowing resources at sites with complex configurations to inform users about how to obtain accounts on specific machines given their existing access to the administrative domain.

The EUROGRID Resource Broker consists of two major components; a master delegation and user identity manager (the Single Vsite Broker), and a slave Local Resource Checker (LRC) which actually checks whether the requested resources are available on a particular system and which generates the QoS offers if those resources are available. In the Grid Interoperability project, the LRC was extended with resource translation modules so that UNICORE requests could be converted into searches of Globus Toolkit metadata. When translating to Globus Toolkit 2, the requests are converted into a set of LDAP searches that can be fed into an MDS server (either a GRIS or a GIIS, depending on local preferences), and when translating to Globus Toolkit 3 (see Figure 3), the requests are mapped[BF01] (via an ontology described in XML) into an XPath

search term that can be understood by a GT3 Index Service containing data structured using the Glue information schema[Glu] and which would produce a non-empty result if the original resource request is satisfiable. In both cases, there is an additional fall-back strategy for untranslatable resources; it turns out that many UNICORE concepts have no mapping within the default configuration of the various Globus Toolkit metadata providers (for example, descriptions of software available or resource limits on batch queues) so these are handed back to the standard UNICORE local resource checking mechanism. Although this means that the amount of UNICORE information that has to be provided may be quite high even where the underlying Grid system is not UNICORE, it does allow the system administrator to get the broker working with relatively little effort as they have direct control of the database that the UNICORE resource checker examines.

The final part of the EUROGRID Resource Broker is a plug-in interface for application-domain expert brokering modules. These are designed to provide one-to-many mappings from application-specific descriptions of problem size (described as an XML document

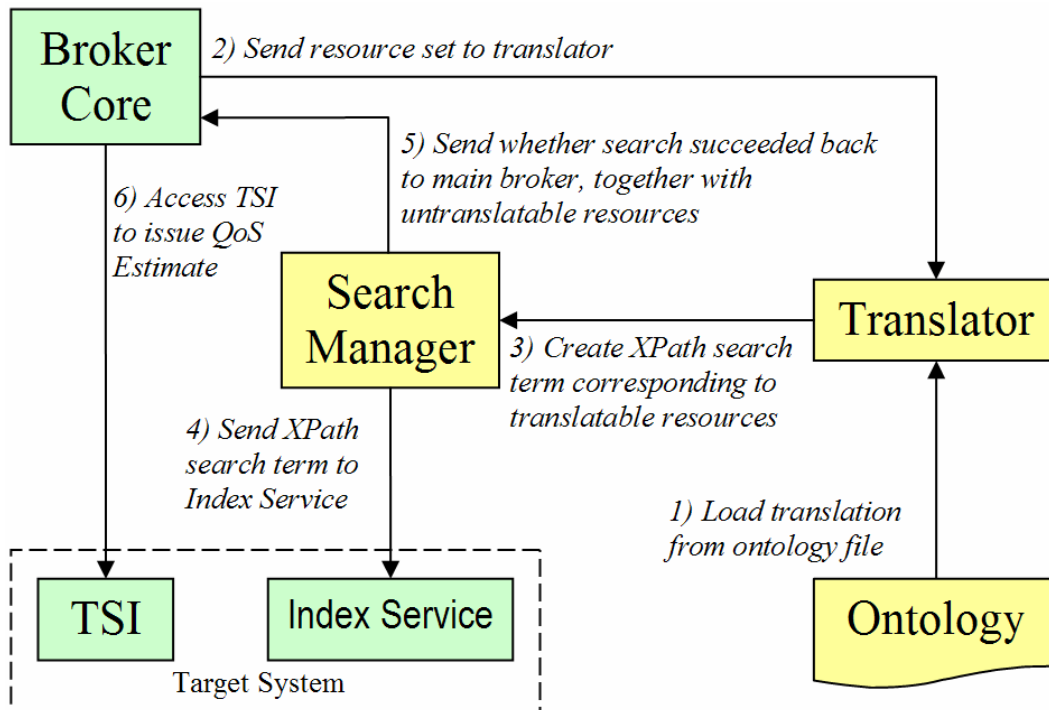


Figure 3: Broker Translating from UNICORE to GT3

passed in the resource request from the client, where UNICORE has a long history of application-specific plug-ins for presenting user interfaces) to possible concrete resource estimates of what is required to execute the application. This allows a single request to get (when combined with the built-in delegation) offers back from a wide range of machines, and indeed for that request to get back many offers from a single machine (which may vary in, say, the number of processors used and consequently the overall cost and turnaround-time.) This process allows for both discovery of systems and picking an execution model based on expert knowledge of the internal configuration of those systems and the applications running on them.

Every QoS offer is stated in terms of a modified resource set; this allows the resource brokers to make offers based on not just what the user asks for, but also on what the broker believes to be an acceptable alternative (for example, it might offer fewer processors for a longer time-slot). Furthermore, the QoS offers generated are bound to abstract tokens (Tickets) issued by the resource broker that are valid only when associated with the full workflow tasks that the brokering was originally performed for; all the user has to do is select which ticket (within the validity period of that ticket, of course) to use with their workflow in order to get the promised

QoS on the stated resources. Because the tickets are issued by trusted agents in the system, the user may have a reasonable degree of trust in their interpretation and in the fact that tickets may not be repudiated (except under the explicit circumstance of lifespan expiry), and the workflow engine may check that a ticket is valid and only used once, allowing resources to make one-off offers without having to worry about whether those offers would subsequently be abused.

Choosing Between Ticketed Offers

The next stage of resource brokering is the process of matching resources (on the basis of the QoS offer tickets issued, which encode snapshots of resource policy) to the user's preferences for what a resource consists of. Currently, this process is done via a GUI client which presents a table of offers received (see Figure 4, where valid offers are in white and invalid offers are in red) and which sorts the offers by factors like whether the offers are valid (a broker may make an invalid offer containing an advertisement describing how, for example, to get an account on the system so that valid offers might be made in future) and how much those offers cost. The prices may be

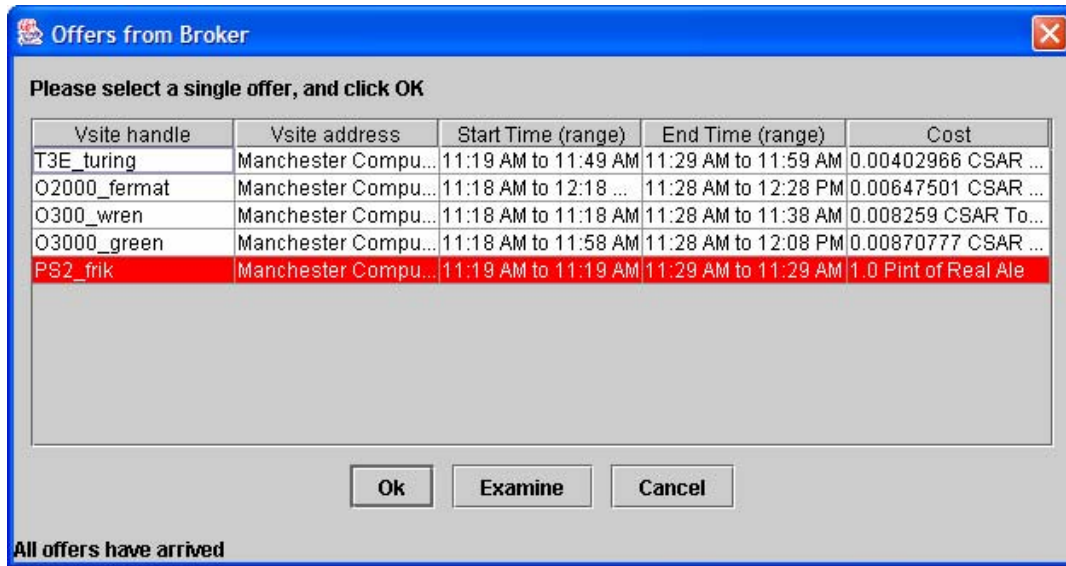


Figure 4: Resource Broker Offer Selection Dialog

displayed in either in the base “currency” of the resources being brokered (often abstract computation tokens of some kind), or converted into the user’s own preferred real currency.

In the future, such selection will need to be done automatically according to some user-specified policy. We are currently looking into using a scheme like Condor’s ClassAds[RLS] to filter and rank offers. Initially, this would be used just to offer a more sophisticated sorting mechanism in the GUI client, but full automation is an obvious stage to advance to after that, just picking the most highly rated offer that satisfies the filtering conditions. Indeed, with full automation it becomes possible to augment a workflow engine with the ability to run a section of a workflow at a site selected by brokering, which would in turn allow for long-running sets of tasks that pick optimal resources for each stage in the overall process.

Mapping Job Submissions to Concrete Execution Plans

The final stage of job submission within the UNICORE model is the mapping of a submitted job to a concrete execution plan. This is integrated within the UNICORE job manager component, the Network Job Supervisor (NJS), which contains a description of how to invoke each of the applications it supports.

When a request to run a particular application arrives at an NJS, the name of the application is

looked up and a shell script that describes exactly how to invoke the application is extracted (a procedure known as Incarnation). This is written to a location where the underlying batch subsystem can read it and the batch subsystem is then instructed to execute that script as its job. This allows the same abstract job request to be sent to any number of different batch subsystems without modification. Only the NJS needs to understand how to actually run a particular program, so the amount of system-specific information that needs to be passed around is greatly reduced.

The resources consumed are controlled by the ticket attached; if a ticket is present at all (direct job submissions do not include tickets) then that ticket is looked up in a local database to check to see if the ticket is being used for the task for which it was issued, to see if the ticket was actually issued before (and not used, so preventing replay attacks on the QoS system) and to see if the resources actually requested match those that the ticket was issued against. These integrity checks allow a greater degree of trust by system administrators in the brokering engine while providing users with guarantees against potentially malicious intermediate agents; even if a particular ticket could be captured in transit, it would not give the attacker the ability to use resources by posing as a legitimate user.

Future Work

It is planned in the future (as part of the UniGrids project[UGr]) to extend the resource broker by adding an automated mechanism for monitoring what resources were actually consumed by a particular job and integrating this with the application-specific brokering engines so that they can correlate the abstract job sizes (e.g. number of grid points to use in the weather simulation) with the observed cost of satisfying those requests. We will be looking to apply machine-learning techniques to enable the automated discovery of models for applications, allowing the use of the full power of domain-specific brokering to be applied to systems without the application developer having to provide a specific performance model and method of determining local performance characteristics.

We shall also be working on extending the brokering engine so that it has a more advanced economic model of the world. A key aim is to gain a mechanism for performing negotiation with sites making QoS offers, with an aim to promoting auctions between the offering sites for the tender being offered by the user when they submit their job. This will better enable the satisfaction of user requirements in job submission; currently optimization focuses on getting the best deal possible for resource providers, often resulting in users getting a less-than-satisfactory service. By allowing users much greater control over what it means to run a job, it becomes possible for a real economy in computational jobs to be formed.

References

- [UNI] UNICORE Forum Web Site,
<http://www.unicore.org/> with
software available from
<http://unicore.sourceforge.net/>
- [Glo] Globus Toolkit Web Site,
<http://www.globus.org/>
- [EUR] EUROGRID Project Web Site,
<http://www.eurogrid.org/>
- [GRI] GRIP Project Web Site,
<http://www.grid-interoperability.org/>
- [UGr] UniGrids Project Web Site,
<http://www.unigrids.org/>
- [BF01] J. Brooke & D. Fellows, *Abstraction of functions for resource brokers*, 2001. Working Draft in GPA-WG, Global Grid Forum.
- [Glu] *Glue Computing Element Schema*, available online at
http://www.cnaf.infn.it/~sergio/datata/g/glue/v11/CE/GlueCE_DOC_V_1_1.htm
- [RLS98] R. Raman, M. Livny & M. Solomon, *Matchmaking: Distributed Resource Management for High Throughput Computing*, in Proceedings of HPDC7, Chicago, 1998.