A Framework for Sharing Heterogeneous Grid Resources in a Campus Environment

Tiejun Ma[1], Xin Xiong[2], and David Wallom[2]
[1]Imperial College, London, UK
Email: tma@doc.ic.ac.uk
[2]Oxford e-Research Centre, University of Oxford, Oxford, UK
Email: {xin.xiong,david.wallom}@oerc.ox.ac.uk

***Abstract*—Grid computing enables the access of heterogeneous computational resources across a dynamic set of physical organisations seamlessly. A campus grid is an example of the grid solution which is located within a single administrative domain. In the majority of installations these enable connection of all institutionally owned computational resources, making them transparently available to appropriately registered members of staff.**

In this paper we introduce the design and implementation of a lightweight campus grid framework. It manages a set of heterogeneous computational resources, using virtual organisation based policies for each individual grid resource and enables user's to transparently access resources through a unified user-friendly interface.

The proposed framework can discover, aggregate and broker all of these heterogeneous resources with low management costs that matches users requirements. With this framework, a campus grid enables university-wide, regional-wide, national and international grid resources become transparently available to university members through their university account without knowing how to access each individual resource. Such a framework allows scalable grid resource sharing, virtual organisation based resource management, dynamical and autonomous resource brokering with improved usability and manageability in a campus grid environment.

Keywords: Grid Computing, Campus Grid, Resource Broker, Condor, GLUE Schema, BDII, Virtual Organisations, Windows HPCS.

I. Introduction

Grid computing has been widely adopted within the research community to support distributed computational tasks and conduct scientific experiments. Generally, grid computing is a method by which access is given seamlessly to a set of heterogeneous computational resources across a dynamic set of physical organizations, supplying massive computing and storage capability.

In a university-wide environment, a campus grid aims to aggregate available computational resources with easy access to both staffs and students. Within a university there are a large number of desktop systems available in addition to a number of university or departmentally owned dedicated clusters for high performance computing. All of these resources are unlikely to be fully utilised and as such methods to improve return on investment need to be pursued. In addition, there are also national and international grid networks being constructed, such as the UK National Grid Service (NGS) [1] and the European Grid Initiative (EGI) [**?**], which aggregate resources from many different academic institutions using defined open standard grid interfaces.

Many universities wish to aggregate and fully utilise university-wide resources as well as ensure that researchers within their organisation are able to optimally access national resources. This will allow the institution to improve the resource utilisation as well as reduce the perceived overhead of buying expensive high performance computers (HPCs) for limited numbers of researchers. Many university researchers without dedicated computing budget also need such low-cost or even free computing facilities for their research. Thus a number of research and development efforts have been carried out for services grid, desktop grid and volunteer computing systems, such as [2]–[5]. However, there are still several challenges for building such a campus-wide grid system from the perspective of computational resource management, discovery, brokering and user-friendly resource access with trustful security.

First, unlike volunteer computing paradigms, in which computers can be fully available when they are idle. Within many campuses departments are responsible for their own energy costs. Therefore many computational resource providers might only want to share a proportion of their computing resources with others (e.g. between two departments). Also, these providers are unlikely to provide significant systems management effort to external users of such services.

Second, the majority of researchers are not experts in distributed computing and hence will not be familiar with grid technology, including not only accessing remote systems but also knowing how to find out what researchers are available to them. The process of applying and using the X.509 [6] based certificate, and the complexity of using digital certificates to access grid resources also raise possible barriers to usage.the learning curve for users.

Third, many of these computational resources are heterogeneous, and need to be discovered and managed in a campus environment. Thus a resource discovery mechanism

is needed for harvesting these computational resources. The user community also does not present a homogeneous set of requirements for their computational and the status and availability of most grid resources are changing from time to time. Thus an effective and scalable campus grid resource broker (RB) is needed to match and dispatch jobs according to user's requirements.

In this paper, we address these issues, and provide a lightweight campus grid framework. The framework provides resource discovery and harvesting mechanisms for these heterogeneous resources. To improve the usability as well as balance the security concerns, a user-friendly unified access interface is adopted to based on the university user account system. Overall, the proposed framework improves the utilisation, manageability and usability of these computing resources and lower the overall computational cost for universities. It also accelerates the research process for researchers.

The rest of the paper is organized as follows: Section II introduces previous and related work; Section III introduces the architecture overview of the frame work; Section IV introduces technical details of implementations; Section V introduces the evaluation results based on initial implementations; Finally, in Section VI, we draw some conclusions and discuss possible future work.

II. Related Work

The Campus Grid is a popular research and development area for universities, since most universities own a large number of different types of computer and have significant demands for provision of computational resources to support different types of research. Thus many universities have developed infrastructures to achieve such computational resource management goals, such as the Imperial College e-Science Networked Infrastructure (ICENI) [7], the Cardiff Condor Pool [8], University of Bristol campus grid [**?**], and Clemson University Condor system [9] etc. However, compared to systems such as [7], [8], our framework aims to support VO-based individual resource management with policies, BDII and UDDI resource index services based resource discovery and harvesting, especially for Microsoft HPC Server. Furthermore, our framework also aims to take additional resource information into consideration, such as application software availability, geographical location of the resource, the current resource usage and the network performance etc. Work done in [**?**] went some way to support this though had not constructed all the appropriate services.

At the resource brokering level, GridWay [10] is designed as a grid-level meta-scheduler, which relies on the Globus Toolkit [**?**] to dispatch jobs to other cluster-level resources management systems, such as PBS, gLite, Condor etc. However, adopting GridWay was not an ideal solution for us with the following incompatibilities. First, GridWay does not support all existing resource index services directly (e.g., MDS, BDII, UDDI) and relies on the underlying resource management systems to provide resource discovery capability.

Second, we aim to have one unified resource broker, job scheduling and matchmaking capability building on the functionaility provided by the Condor system. Such a solution on its own is sufficient for many campus grid systems and is fully compatible with many existing departmental Condor pools. However, within the Condor system there are no automated methods by which resource information from larger clustered systems may be added into the system. As such this is essential for those institutions that are wishing to aggregate existing production HPC systems into their grid. Within the University of Oxford campus network system there is also considerable network complexity that makes a single Condor solution unfeasible.

Third, GridWay must rely on the Globus Toolkit to submit jobs and stage the results back. However, our future aim is to build the whole campus grid framework on open standards, with which Globus is curently incompatible. We aim to use the GridSAM software as the front-end of our RB and its distributed resource manager (DRM) connectors to provide the job submitting and data staging work, which also provide the capability of accepting Job Submission Description Language (JSDL) described [11] job submission, and Open Grid Services Architecture - Basic Execution Service (OGSA-BES) [12] based job execution [13].

For these reasons, we argue that our framework is an alternative choice for campus grid systems, which can integrate heterogeneous grid resources and natively support most existing grid resource index services using Condor matchmaking capability directly, without deploying another level of grid meta-scheduler.

III. Framework Design

For a campus grid framework, several key requirements are identified, which can be briefly summarised as follows:

First, the framework should be scalable to a large number of connected resources, able to automatically limit the resources consumed to an acceptable level and control their intrusiveness on the server they are running on including the performance of application services, servers or networks [14].

Second, the framework should be able to adapt to the diversity of published resource information. As each site may adopt different mechanisms of advertising its resources and publish them to a particular type of information hosting/indexing server. All of these diversities should be tolerated and not affect the success of resource information discovery.

Third, the framework should be reliable and tolerate the instability of connected grid resources. Since the framework will gather resource information from multiple information

sources that are published in a distributed manner, it is likely that querying information for an individual site/cluster might fail due to the network delay or host failure etc. In such circumstances, the framework should be reliable and tolerate these failures, minimize the dependencies between the information retrieval and translation for each site/cluster.

Based on these requirements, we introduce the design of a scalable and policy-aware distributed grid resource management framework as shown in Figure 1. From this we can see that the framework contains an RB, a virtual organisation management server (VOMS), a resource information harvesting and integration engine, a number of resource information index servers for Linux, Unix and Windows based resources, and finally a single sign-on authentication service.

Together with these components, the framework provides brokered access for users, attribute based resource and user management, heterogeneous resource publishing, discovery, aggregation and user's jobs matchmaking with the RB. In the following sections, we will introduce the design of each of these components in detail.

A. Virtual Organisation-Based Resources and Users Management

For improving the usability of the grid security system we have integrated university single sign-on (SSO) to provide the users with credentials for accessing grid resources. The framework automatically generates a short lived X.509-based credential for the user. Thus all of the currently developed grid access mechanisms are transparently available to users, such as Web-portals, workflow engines and SSH-based access methods. This methodology allows all users to gain access to the system using their university account and not forcing them to go through complex externally managed procedures.

We have locally built a VOMS system, which stores resource information, resource usage polices and user information. The resource information includes the grid resource's address and the name of its underlying DRM system. The user's information includes user's name, role and department. This information is from the university SSO service and is used for controlling access to grid resources. The resource usage policies include whether a grid resource is enabled within the campus grid and what fraction of its resources are accessible to the grid. Valid users, verified by the VOMS, can submit their jobs to the RB, which then matches the job specification with the most appropriate resource and dispatches the job to the allocated resource.

To increase fault tolerance, the VOMS also stores the correct DRM system installed on the grid resource. This is due to experience of a number of systems incorrectly configured and publishing false grid resource information and hence causing job submission failure. Harvesting such inaccurate resource information will result in inconsistent resource states and job submission failures. However, within large-scale distributed systems, manually checking the correctness of the published resource information is not feasible. Thus, before the information of a grid resource is advertised as available, the critical resource information will be automatically validated and verified against the pre-stored information in the VOMS server according the previous success history. In the case of a failure of verification, the resource will not be advertised. In such a way, we ensure that all of the advertised resource descriptions are valid hence improving the success rate of job execution.

B. Grid Resource Integration and Brokering

For most heterogeneous grid resources (such as Desktops, Windows HPCs and Linux HPCs), resource information, such as the number of CPUs, the size of memory, operating systems and installed software stacks etc. will be published into various resource indexing services, such as Berkley Database Information Index (BDII) [15] and Universal Description Discovery and Integration (UDDI) [16] using the widely adopted Grid Laboratory for a Uniform Environment (GLUE) schema format [17]. The resource integration component can discover and harvest the up-to-date resource information published into these index services. The collected resource information is then translated and advertised into the RB, which enables matchmaking for submitted jobs. Due to the diverse methods of publishing the resource information, it is not easy to gather these various resource information published at different locations in different formats and on different types of servers.

Within the proposed framework, we design a key-value pair-based mapping rule pre-stored in XML-format, which specifies the translation rule for both BDII-based GLUE format as well as UDDI-based GLUE format whilst also supporting various different GLUE versions. The key defines the needed resource information for the RB and the value is the actual format of such information stored in either BDII-based or UDDI-based registries. In such a way, the grid resource information published into either a BDII-based directory service or UDDI-based registry, e.g. number of CPU and memory size, can then be retrieved and advertised into the RB according to the predefined rules.

The RB must be aware of the load on each remote resource. To gather this information the RB will periodically query the current job queue to retrieve the number of queued jobs that have been submitted to a specified grid resource. The RB then retrieves the policy information from the VOMS, such as the maximum allowed jobs to be submitted to a particular site/cluster and adds this as a requirement in the resource description. This information is used to provide load balancing and control between the available grid resources as well as perform policy-aware job submission (e.g. restricted CPUs usage).

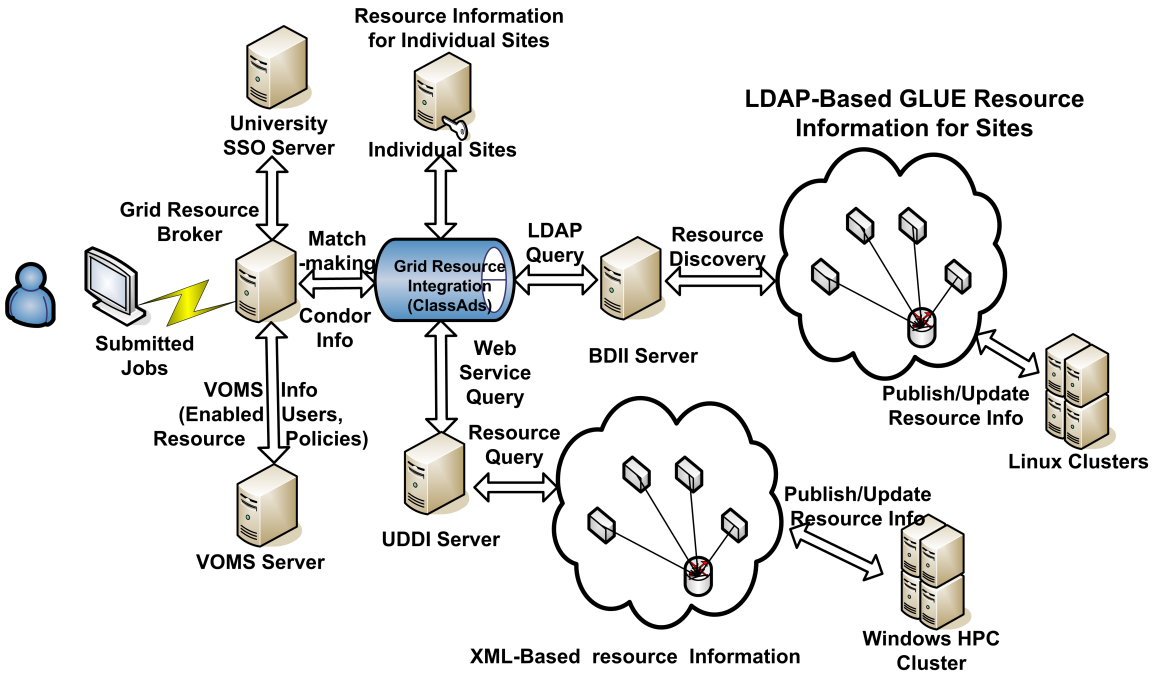

Figure 1. A Campus Grid Resource Management Framework

With our scheme for job allocation and resource selection, the RB matches user job requirements and satisfies resource providers' policies on how they are willing to share their resources. With such a schema, it is also easier to convince resource owners not currently participating, to share their computing resources without breaking the agreed resource usage policies and hence generating extra management costs. From users' perspective, we also allow users to select the appropriate resources for their jobs based on specific requirements.

IV. Implementations

For implementing the proposed campus-wide grid resource management framework, we hope to reuse components from available toolkits as much as possible and aim to build a relatively lightweight software stack to deploy the framework as easily as possible.

We have adopted Condor-G [18], a compute-intensive workload management system, as the meta-schedular of our campus grid broker. It performs Grid-like computing and has the capability of aggregating a large number of heterogeneous systems as one computing pool. The Condor system also provides sophisticated job-resource matchmaking mechanisms, allowing dynamic requirement based job allocations to be performed.

However, Condor does not support VO-based policies. It cannot recognise and submit jobs to Windows HPCs and harvest grid resource information from BDII-based or UDDI-based resource information publishing mechanisms. In addition, we have found that high frequency concurrent access of the job queue will result in a performance degrade of Condor daemons. Thus, we also design a simple algorithm within our framework for solving this problem.

Therefore on top of Condor-G we build an additional software stack to fully enable the features needed for our campus grid, such as supporting VO-based resource and user management, grid resource harvesting from various registries and providing customised submission scripts to improve the usability for users.

A. Grid Resource Information Harvesting and Aggregation into the Resource Broker

For resource discovery purposes, resource description information must be made available in such a way that the meta-scheduler can access the information. Currently, GLUE is a widely adopted format for describing resource information and such information is hosted by BDII systems or Monitoring and Discovery Service 2 (MDS2) [19]. Resource information can also be published as metadata into an XML-based UDDI registry, such as Grimoires [20] or MDS4 [21], both of which are service registries designed to be the central contact point where service/resource endpoints are published. To provide an XML-based registry, we have adopted Grimoires, which is able to expose registry entities, such as businesses and services, together with their attached metadata. By doing so, standard operations defined in the Web Service Resource Framework (WSRF) [21] specifications can be used to operate on registry entities. Furthermore, Grimoires 1.7 and above support the GLUE schema natively.

In order to let the RB adapt to these heterogeneous resource information publishing mechanisms, we implement the XML-based mapping schema, which contains details of the mapping relationship, described in Section 3. It specifies the requested attributes and its related GLUE schema for BDII-based GLUE publishing or the XPath [22] query expression for UDDI-based GLUE publishing. When the RB starts, such mappings will be read into a hierarchical hash table. When the resource information for a site is retrieved, the RB will first try to discover its type (e.g., GLUE version), then start matching the required information. In this way, we improve the flexibility and adaptability of the resource information harvesting and allow the GRB to work with multiple types of index services. When additional resource information is needed, we only need to update the mapping rule file rather than redevelop and recompile the system packages, which simplifies the implementation development. Currently, we adopt the scheme of creating one resource information query instance for each grid site/cluster to avoid dependencies and reduce the query delay. In this way, a failure or delay of querying a single grid resource will not affect others and improve the reliability of the RB.

We design and implement the VOMS using the PostgreSQL database server with predesigned tables to store all necessary information as introduced in Section III-A. Thus together with the RB, the implemented system can harvest heterogeneous published resource information, such as free CPUs, CPU types, operation systems, memory size and job manager etc. In addition, a function of estimating the round trip network delay is implemented, which records the network overhead for resource harvesting. Such information can be used for network performance related measurements. If the geographical coordinates of a site is published, such information is used to calculate the distance between the RB and the grid site. This information is used for location-aware job submission. For example, restricting the submission of jobs within a given specified distance. Furthermore, we implement a function of monitoring the current RB's queue, which analyses how many jobs have been submitted to a specific resource. Together with the retrieved resource usage polices from the VOMS, this information allows the the RB performing load balancing according to current resource load and usage restrictions.

Figure 2 shows a sample resource description produced by the RB. It adopts the ClassAd format used in the Condor systems, with new additions/differencs between this and standard ClassAds highlighted. This includes, the current resource usage for estimating the current resource workload, the network query delay, the geographical distance between the resource and the RB, the maximum number of jobs allowable on a resource and an exemplar of an installed piece of software. These have been added to enable more accurate resource description and hence adaptive job requirements matchmaking.

```
Rank="0.000000"
Location="OXFORD, UK"
OpSys="LINUX
WantAdRevaluate=True
TargetType="Job"
Usage="0.031914893617021274"
NetworkDelay="503.0ms"
MajorVersionTag="1"
Longitude="-1.259823"
Arch="INTEL"
Name="condor-zeus.oerc.ox.ac.uk"
MyType="Machine"
DistanceToRB="0.0km"
Memory=2026
State="Unclaimed"
TotalCPUs=94
Latitude="51.759792"
JobManager="condor"
CurMatches=3
Requirements=(CurMatches<96)
&&(TARGET.JobUniverse==9)
resource_name="condor-zeus.oerc.ox.ac.uk/jobmanager-condor"
UpdateSequenceNumber=100000218
CurrentRank="0.000000"
FreeCPUs=91
DLPOLY=TRUE
```

Figure 2. A Sample Customised Resource Description ClassAd

We also implement a concurrent RB query avoidance algorithm (see Algorithm 1) to avoid CPU and system I/O overhead, which affects the performance of the RB. We implement a daemon, which takes the snapshot of the RB queue information at a fixed interval (1 minutes in default), this information is stored in a global buffer and valid for a specified period. The additional RB queue query will only be executed when it reads the global buffer and finds that the information in the buffer is empty or invalid. In addition, in order to avoid concurrent global buffer access before each query instance starts, a random delay will be generated. This will avoid triggering the concurrent RB queue query when the global buffer is empty. In such a way, it ensures that concurrency will be avoided.

Overall, the whole framework is implemented in a modular manner and has been made operating system independent way using JAVA, so that additional functionality may be easily added in the future and the implemented system can be easily reused in different environments or be extended to adapt to specific requirements.

B. Microsoft Windows HPCS08 Information Advertising

With the introduction of Windows HPC Server 2008 (HPCS08) a number of institutions have installed these systems. Oxford is one such institution and as such our campus grid must also provide access to these facilities. However, the Windows HPC platform does not provide native GLUE based resource information publishing. To enable this we provide a solution to retrieve resource information from the HPCS08 scheduler, generate GLUE format resource descriptions and publish such resource information into the XML-based Grimoires registry.

Algorithm 1 A simple concurrent GRB queue query avoidance algorithm

Let p_i be the process for retrieving resource information for resource i;
Let t_i be the resource information update interval for resource i;
INIT:
Start query GRB queue Info
Create an empty cache space (C) in the memory.
C will be emptied for every t_i.
for all p_i **do**
  Inject a random delay $x \in (0,\ 10)$ seconds for p_i.
  if x not reached **then**
    wait;
  else
    read C
    if $C == \varnothing$ **then**
      copy C to the local process
    else
      take a snapshot of the current GRB queue and store in C
    end if
    Parsing the interested GRB queue info
  end if
end for

In addition, in order to enable the JSDL job to be executed in HPCS08, GridSAM [23] has been revised to consume an OGF HPC Basic Profile [**?**] compliant web service.

The GLUE information publishing system developed for HPCS08 is implemented in three packages: GLUE-Generation (GLUE-G), GLUE-Translation (GLUE-T) and GLUE-Publish (GLUE-P). The GLUE-G provides access to HPCS08 and retrieves the resource information such as the number of CPU, memory, running job numbers etc. All of this information has been constructed in key-value form, which is similar to the GLUE schema layout. The GLUE-T package then reads the input stream of GLUE-G and translates the retrieved information to XML according to GLUE Schema 1.2 (which can be extended to 2.x). As the information retrieval and translation has been separated, GLUE-G is developed in C# while GLUE-T is in JAVA, information translation is generic and GLUE-T can be applied to other computing platforms, exposing a common programming style to the extensible API. The GLUE-P is a JAVA Web service that addresses the aspects of publishing HPCS08 GLUE information to an XML-based UDDI registry using Grimoires.

V. Evaluation

Following the implementation introduced in Section IV, we are now able to test the performance of the framework.

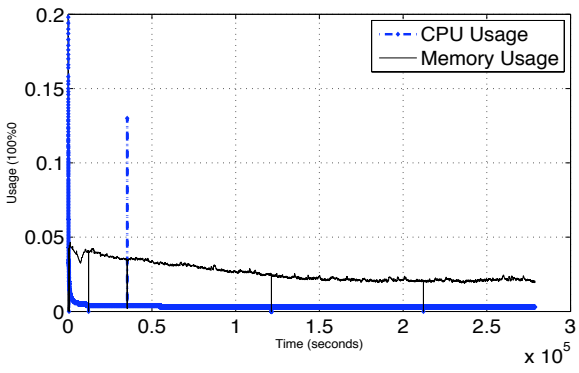

Figure 3. CPU and Memory Usage

We deploy specific components including; the VOMS and its associated PostgreSQL database, the implemented GLUE-G, GLUE-T and GLUE-P components on a Windows HPC Server 2008 system and a UDDI registry using Grimoires 1.7. We also harvest resource information from BDII index servers from the UK National Grid Service, the Oxford Supercomputing Centre (OSC) and departmental clusters and condor pools. We deployed the RB on a Dell Poweredge 2950 machine with dual Intel Xeon dual core 3GHz processors, 4GB RAM and 584GB data capacity. The kerberos-based SSO service, Condor-G 6.8, and most of our implementations are deployed on this server for operating the VO-policy aware grid resource harvesting, job matchmaking and dispatching for resource brokering.

Overall, the RB pool aggregates between 40 and 50 clusters depending on availability, including departmental resources, UK National Grid Service clusters, the OSC HPC clusters and OeRC Microsoft HPC. These resources show a reasonable diversity and enable us to show that our framework enables easily configurable management and sharing of these heterogeneous grid resources with satisfactory services according users' requirements. Within a month, the RB has successfully dispatched around 13500 real users individual jobs onto the dynamically discovered and connected heterogeneous grid resources, which contains more than 1500 CPUs[1] without breaking each resources usage policies as well as involving significant manual management effort.

Another requirement we had was about the system overhead of the RB software developed. Thus during the RB test running period, we also monitored the CPU, memory and network overhead generated by the implemented components on the overall RB system hardware.

Figure 3 shows the CPU and memory usage of the RB daemon. From Figure 3, we can see that the CPU usage of our implementation is stable to less than 1%, which satisfies the overall requirement of low system overhead.

[1]The number of resources and CPUs varies and depends on the availability of each individual resource.

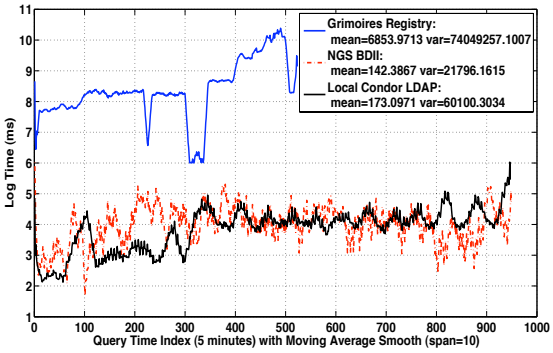

Figure 4. Network Query Delays

We can also see that the CPU and memory usage do not increase over time, thus the daemon can run stably within the framework for significant periods of time without operator intervention. Figure 4 shows the logged query delay of discovering resources from various resource information indexing services. It shows the query delay of BDII-based departmental Condor resources and BDII-based National Grid resources are around 170ms on average. The query delay of the Grimoires registry for HPCS08 is much longer and less stable. These results indicate that the average query delay of the BDII-based queries is much lower than for the XML-based Grimories queries. Although, the Grimoires-based service registry (UDDI) provides powerful capabilities, such as Xpath-based queries and semantic-based resource discovery, in terms of performance, the BDII-based resource registries are more efficient than the XML-based resource information discovery at the current stage.

Overall, the evaluation results show that the designed framework can broker these heterogeneous grid resources smoothly with a reasonable system overhead. The framework can also manage these resource autonomously according to the given policy restrictions and dispatch user submitted jobs according to their job specifications as well as aware resource location, network condition and balance the resource usage. All of these results match our design goals.

VI. Future Work and Conclusion

At the moment, we are actively continuing to develop a more stable and robust system and enrich system functionality. For example, more sophisticated matchmaking and economic models can be designed and adopted for the resource selection according to the job requirements. This will improve the job success rate and lower the users costs. In addition, as the cloud-computing paradigm becomes more and more widely used, the concept of our framework can be transformed into a resource-brokering framework for cloud-based resources. One of the major functionalities provided by a cloud service is the ability to find suitable resources to create or migrate virtual machine images to satisfy user specified requirements. This will include future 'cloud bursting from private cloud resources within a single institution to those within public clouds when the resource utilisation is very high. Such functionality is similar to the job submission matchmaking within our framework but such a paradigm will have a different virtualised resource management infrastructure. The design of this framework would possible adapt to such cloud-based services with a customised cloud resource broker and virtual machine pool manager. We will look into such development for campus-based cloud infrastructures in the future.

In this paper, we introduce a framework for enabling heterogeneous grid resources transparently shared within a campus grid environment with supporting VO-based resources and users management. With the proposed framework, heterogeneous grid resources, such as departmental desktops, Linux-based and Windows-based HPCs can be aggregated into a RB dynamically and autonomously. Such a campus grid framework reduces the complexity of using and managing grid resources. It allows those researchers who do not familiar with grid technologies to avoid using X.509-based digital certificate to access grid resources transparently with their university SSO accounts. It also allows the grid resource providers restrict the proportion of resources they want to share.

We have implemented the proposed framework and the RB, which is capable of discovering, harvesting and aggregate a large number of computational resources into the campus grid. The RB can discover the resource information published in GLUE format from either BDII or UDDI registries regardless the format of the published resource information and the type of resource index server, then match suitable resources according to users' job specifications. At the moment, the RB's matchmaking capblity relies on customised Condor's ClassAd mechanism. Within in the framework we propose a solution of dynamical ClassAd generation, customisation and advertising according to the current job queue' status, VOMS-based resource usage restrictions, resource locations and the network conditions between the RB and each individual resources. Then these published resource information can be discovered, translated, verified and advertised into the GRB for more sophisticated matchmaking.

We also implement a mechanism to publish HPCS08 resources into the Grimoires registry using the widely adopted GLUE format in a UDDI compatible way and allow jobs to be submitted to HPCS08 resources from the RB. This experimental solution allows Windows resources to be aggregated within a grid environment.

The design of the framework improves the scalability, usability and manageability of a campus grid system with large-scale heterogeneous local, regional and national computational resources. Users can then submit their computa-

tional jobs to the campus grid and allow the RB to dispatch these jobs to each individual resources according to the job description, resources usage policies specified within the VOMS and current resource conditions without knowing where are these resources.

In order to improve the performance of the RB, we also implement a simple algorithm to avoid concurrent job queue access, which will generate an unexpected CPU and system I/O overhead and affects the performance of the RB. Our evaluation results show that such a solution is effective and the overall system overhead of the RB is reasonable.

The initial implementation was evaluated within a campus environment to dispatch real-users jobs. It brokers jobs to more than 40 heterogeneous national, institutional and departmental computational resources, which include the UK NGS partner and affiliate resources, OSC HPC systems, university departmental resources and an instititional HPSC08 resource. It processes in excess of 13500 individual jobs successfully per month in accordance of users job specifications without violating resource usage polices.

Overall, the proposed framework allows scalable grid resource sharing, dynamically and autonomously resource brokering with a reasonable usability and manageability in a campus grid environment, which achieves our design goals and can be reused easily for many other grid resource sharing environments.

VII. Acknowledgements

We would like to acknowledge Steven McGough, Vesso Novov and David Colling our partners in the GridBS project. We would also like to thank the technical support from the Open Middleware Infrastructure Institute (OMII) UK. This work is funded by the OMII UK, in conjunction with the GridBS project.

References

[1] "National Grid Service (NGS)," http://www.ngs.ac.uk.

[2] D. Wallom and A. Trefethen, "Oxgrid, a campus grid for the university of oxford," in *Proceedings of the UK e-Science All Hands Meeting*, 2006.

[3] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, "From open grid services infrastructure to ws-resource framework: Refactoring & evolution," 2004.

[4] P. Kacsuk, N. Podhorszki, and T. Kiss, "Scalable Desktop Grid System," *Lecture Notes in Compt. Sci.*, p. 27, 2007.

[5] D. Anderson, E. Korpela, and R. Walton, "High-performance task distribution for volunteer computing," in *First IEEE International Conference on e-Science and Grid Technologies*, vol. 147. Citeseer, 2005.

[6] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2002.

[7] A. Mayer, S. McGough, N. Furmento, J. Cohen, M. Gulamali, L. Young, A. Afzal, S. Newhouse, and J. Darlington, "ICENI: An integrated grid middleware to support e-Science," *Component Models and Systems for Grid Applications*, vol. 1, pp. 109–124, 2005.

[8] J. Osborne and A. Hardisty, "Cardiff Universitys Condor Pool: Background, Case Studies, and fEC," in *Proc. of the UK e-Science All Hands Conf.*, 2006, pp. 361–364.

[9] "Clemson University Condor system," http://citi.clemson.edu/htc.

[10] E. Huedo, R. Montero, I. Llorente, D. Thain, M. Livny, R. van Nieuwpoort, J. Maassen, T. Kielmann, H. Bal, G. Kola *et al.*, "The GridWay Framework for Adaptive Scheduling and Execution on Grids," *SCPE*, vol. 6, no. 8, 2005.

[11] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, "Job Submission Description Language (JSDL) Specification, Version 1.0," in *Open Grid Forum, GFD*, vol. 56, 2005.

[12] A. Grimshaw, S. Newhouse, D. Pulsipher, and M. Morgan, "OGSA Basic Execution Service Version 1.0," in *Open Grid Forum*, 2006.

[13] D. Colling, A. S. McGough, J. M. Smith, V. Novov, T. Ma, D. Wallom, and X. Xiong, "Adding standards based job submission to a commodity grid broker," *eScience, IEEE International Conference on*, vol. 0, pp. 380–381, 2008.

[14] P. Stelling, C. DeMatteis, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski, "A Fault Detection Service for Wide Area Distributed Computations," *Cluster Computing*, vol. 2, no. 2, pp. 117–128, 1999.

[15] "Berkeley Database Information Index," https://twiki.cern.ch/twiki//bin/view/EGEE/BDII.

[16] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter *et al.*, "The Universal Description, Discovery and Integration (UDDI) Specification," *Version*, 2003.

[17] S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Konya, M. Mambelli, J. Schopf, M. Viljoen, and A. Wilson, "GLUE Schema Specification-Version 1.2," *OGF GLUE-WG*, 2007.

[18] M. Litzkow, M. Livny, and M. Mutka, "Condor-a hunter of idle workstations," in *Proc. of the 8th Inter'l Conf. of Distributed Computing Sys.*, vol. 43, 1988.

[19] X. Zhang and J. Schopf, "Performance analysis of the globus toolkit monitoring and discovery service, mds2," in *Proceedings of the international workshop on middleware performance (MP 2004)*, vol. 4. Citeseer, 2004.

[20] S. Wong, V. Tan, W. Fang, S. Miles, and L. Moreau, "Grimoires: grid registry with metadata oriented interface: robustness, efficiency, securitywork-in-progress," *In Proceedings of Work in Progress Session in Cluster Computing and Grid (CCGrid), Cardiff, UK*, 2005.

[21] J. Schopf, M. Darcy, N. Miller, L. Pearlman, I. Foster, and C. Kesselman, "Monitoring and discovery in a web services framework: Functionality and performance of the globus toolkit's mds4," *Preprint ANL/MCS-P1248-0405, Argonne National Laboratory, Argonne, IL*, 2005.

[22] J. Clark, S. DeRose *et al.*, "XML path language (XPath) version 1.0," *W3C recommendation*, vol. 16, p. 1999, 1999.

[23] W. Lee, A. McGough, and V. Novov, "GridSAM-Grid Job Submission and Monitoring Web Service."