Adding Standards Based Job Submission to a Commodity Grid Broker

David Colling[1], A. Stephen McGough[2], Tiejun Ma[3],
Vesso Novov[2], Jazz Mack Smith[2], David Wallom*[3] and Xin Xiong[3]

[1] Department of Physics, Imperial College London, London SW7 2AZ, UK
d.colling@imperial.ac.uk
[2] Department of Computing, Imperial College London, London SW7 2AZ, UK
{asm, jms3,vesso}@doc.ic.ac.uk
[3] Oxford e-Research Centre, University of Oxford, 7 Keble Road, Oxford, OX1 3QG, UK
{tiejun.ma, david.wallom, xin.xiong}@oerc.ox.ac.uk

***Abstract*—The Condor matchmaker provides a powerful mechanism for optimally matching between user task and resource provider requirements, making the Condor system a good choice for use as a meta-scheduler within the Grid. Integrating Condor within a wider Grid context is possible but only through modification to the Condor source code as each new mechanism for connection to remote resources is defined. In this paper we describe how the emerging standards for job submission and resource description can be integrated into the Condor system, thereby allowing arbitrary remote Grid resources which support these standards to be brokered using Condor.**

***Keywords*-Grid Computing, GridSAM, Condor, Distributed Computing, Windows HPC**

I. Introduction

The Grid [1] provides a platform in which owners of computational power – referred to as *resources* – can expose these for consumption by *users* potentially for financial (or other) reward. In order for users to discover suitable resources for their *jobs* and resource owners to obtain high utilisation of their resource, a brokering or meta-scheduling service is required. Conventionally a resource owner, who manages their resources through a Distributed Resource Manager (DRM), will *advertise* their resources to a *Grid brokering service*. Likewise users will *submit* usage requests for resources to the broker. The broker is then responsible to *select* resources which meet both the users and the resource owners requirements. The broker may also provide mechanisms to *monitor* the progress of the users computation on the resource and *transfer files* both to and from the resource on behalf of the user. This generic Grid architecture is illustrated in Figure 1.

As the number of both sites exposing their resources and the number of resources within these sites increase the problem of selecting the *'best'* resource to use becomes more complex. Where '*best*' is defined as an optimised choice based on a combination of both the users and the resource owners requirements. This complexity comes not only from the scale of the problem but the heterogeneity of the technologies used

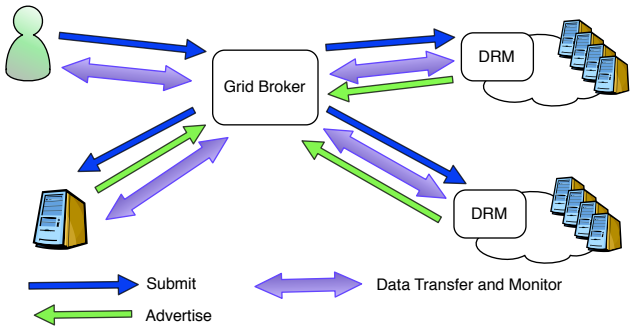

Fig. 1. A generic Grid architecture

to advertise, submit, select, monitor and transfer files. Vendors exist such as (UNICORE [2], Globus [14], Condor [3], gLite [4]) each providing a customised solution to these problems. However, in general these solutions present a tie-in to their particular interfaces. Users therefore cannot arbitrarily discover and use any resource available on the internet for which they have security clearance. Instead only the subset of resources with which their software solutions are able to communicate with its supported interface. For the Grid to truly become a success it is necessary to break down these incompatibilities through the utilisation of open standards for these interfaces to prevent 'vendor lock-in'.

The work within the Open Grid Forum (OGF) [5] has led to a number of standards for resource discovery e.g. Grid Laboratory Uniform Environment (GLUE) [6]), job description e.g. Job Specification Description Language (JSDL) [7] and submission interface, e.g. Open Grid Services Architecture Basic Execution Service (BES) [8]. These are all recommended standards and as such have a number of interoperable server side implementations developed including such projects as OMII-UK GridSAM [9] and Microsoft HPCServer 2008 [10].

Of the implementations of the OGF standards that have been produced there are though no high functionality clients and as such in this paper we propose integrating a set of standards based interfaces into Condor. Through the use of these

*The correspondence author.

standards Condor can be used as a meta-scheduler capable of interfacing directly with any standards compliant DRM. Thus any new system implementing these standards may be quickly and easily added to the grid, providing a clean and clear interface between the DRM and Condor. We exemplify this through the ability to gather resource information from both the Microsoft HPCServer 2008 cluster and resources made available through GridSAM – a veneer service for exposing DRM systems through a BES interface – along with the submission of jobs to these systems. Both sets of resources are advertised through GLUE schema documents which are translated into ClassAds for submission to Condor.

This paper is set out as follows. In Section II we introduce related work. In Section III we describe the OGF recommended standards used within this work before discussing Condor in more detail in Section IV. In Section V we describe the architecture used to allow Condor to interface to the Grid through standards-based interfaces before concluding.

II. Related Work

Many Grid brokering systems exist, including Condor [3], gLite [4] and Maui [11] of which Condor is one of the most popular. Although Condor does not directly support the OGF standards for resource advertising and job submission it is possible to submit tasks from Condor into a number of different Grid frameworks including, Globus [14], NorduGrid [15], UNICORE [2], PBS [16] and LSF [17]. This has been achieved by providing grid specific adapters that sit between the condor system and the remote grid system. Resource descriptions are translated into ClassAds and bespoke code added to submit and monitor jobs. This process is simplified through the use of interfaces in Condor – a command line interface for submitting resource ClassAds and a Grid ASCII Helper Protocol (GAHP [18]) for accessing the DRM. However, as each implementation requires alterations to the Condor source – DRM specific logic added to the client side of the GAHP, this is a non-scalable solution where each time a new Grid system is developed by a vendor community a new 'shim' needs to be developed, tested and deployed.

An alternative approach to ours is used by the GridWay [19] project. GridWay is designed as a Grid-Level meta-scheduler which uses Globus to dispatch jobs to the DRM systems, such as PBS, gLite, Condor. In very much the same way as Condor though the Gridway solution uses proprietary Globus interface descriptions which makes the addition of new remote resources using different local management solutions more problematic. They also do not make use of any automated central resource information service, even including the Globus Monitoring and Discovery Service (MDS) or gLite BDII. Instead it relies on communicating directly with each of the remote resources underlying DRM systems to provide resource status information. This solution, though scalable within a small grid system, becomes problematic where you are connecting large number of systems. In our approach specifically require a centralised set of information providers to be available where all resources registered for the grid are able to publish their status information. This we feel offers a clearer abstraction between the meta scheduler and remote resources and can remove situations where remote system failures can introduce errors into the broker. We are also able to introduce redundant information provider systems into the process which will ensure greater reliability.

III. Grid Standards

In this section we describe in more detail the OGF recommended standards used within this work.

A. GLUE

The Grid Laboratory Uniform Environment (GLUE [6]) schema is an information model used to describe the features and status of a particular resource within a Grid environment. It has been designed to be independent of the implementation thereby allowing for interoperability between solutions. The overall schema is extremely flexible, therefore for this solution we have considered only the representation of a single computational resource. In general though the information model can be broken down into three different types of information,

- Static system information on a resource such as number of nodes, operating system, type of DRM.
- Policy information on the resource, i.e. Maximum queue lengths, maximum length of a running job.
- State of the system, number of queued jobs, running jobs, free slots.

Current implementations from EGEE, NorduGrid, Open Science Grid and the UK NGS make use of an LDAP based repository though this is completely independent of the schema itself. Other contributors to the GLUE schema include Globus, UNICORE, GriPhyn [20], and APACGrid [21].

B. JSDL

The Job Submission Description Language (JSDL) describes a task that a user (or agent acting on behalf of a user) wishes to have executed. The language is an open contents template document, describing what is required rather than how to achieve it, normally rendered in XML. The job description is separated into four main sections.

- Job Identification, mostly human readable information about the job
- Application Description, contains a definition of the job to execute
- Resource Requirements, gives conditions which the resources the job runs on must satisfy
- Data Staging for specifying locations of executable, input or output data and their transport methods

JSDL has now been implemented natively by a number of different groups with mappings to over fourteen DRM systems.

C. OGSA-BES

The Open Grid Services Architecture - Basic Execution Service (OGSA-BES [8]), referred to as BES, is a partner specification to JSDL. While JSDL defines the language used to describe a task the BES specification describes how a

service can consume such a document and how to provide monitoring and job control. The BES specification defines a Web Services [24] interface with two main port-types, those of a factory port to allow users to submit, monitor and control sets of activities, along with a port for management of the BES service. For the purposes of this work we are only interested in the submission, monitoring and control of sets of activities. In order to provide the most interoperability between different BES client and server instances a very simple state model is used which contains the states of Pending, Running, Finished, Terminated and Failed. The valid transitions between these states is illustrated in Figure 2.

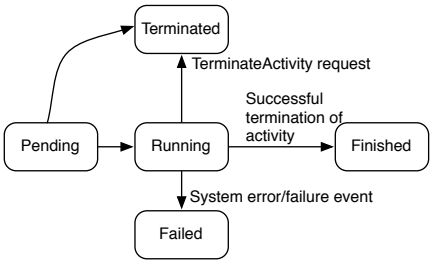

Fig. 2. The OGSA-BES basic State model

In order to allow for more complex descriptions of state without breaking servers or clients which cannot support these more complex states a system of sub-state modelling is adopted. Here any of the primary states can be sub-stated to give more information, while a less complex client (server) can still use the basic state model – albeit without the extra information. An example of sub-stating the execution of a JSDL is given in Figure 3. We work only with the base model as at present as there are no widely adopted profiles to the BES state model.

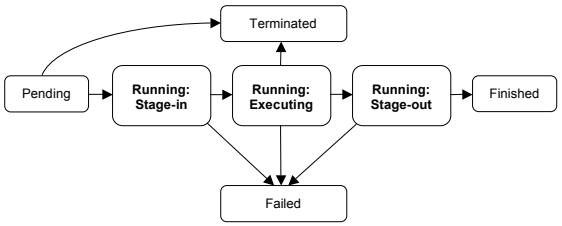

Fig. 3. Sub-states in the BES state model

IV. Condor

Condor was developed originally as a batch job execution service by the University of Wisconsin for the purpose of high-throughput computing. In its default configuration Condor utilises unused computing power within a collection of attached computers. When a computer is unused a Condor daemon indicates this to the Condor Manager which can then use the computer as part of its pool. If someone starts using the computer (either locally or remotely) Condor will vacate, either by terminating the active job or migrating the job elsewhere dependant on the computers configuration.

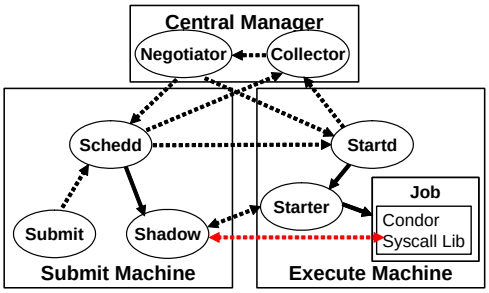

Fig. 4. The standard Condor Architecture

The Condor system is broken up into a number of services as illustrated in Figure 4. A daemon (Schedd) runs within each submit machine, containing information on all jobs submitted from that host. When a user submits a job, the daemon transmits the job description to the central master system, which runs the Collector. On each Execute Machine a daemon runs (Startd) which reports to the Collector the state of that machine. The Negotiator (or matchmaker) looks at the list of submitted job descriptions that it can pair together with the information held about resources in the Collector, when a match is found the corresponding submit and execute hosts daemon process are notified. The submit system Schedd will then send the job to the execute machine Startd and generate a Shadow daemon for the job on the Submit Machine which acts as a local instance of the remote service for file reading etc. The Startd then generates a Starter for the job which launches the job and interacts with the Shadow daemon.

Within the condor system the same description language is used for both job and resource descriptions, the ClassAd. These are a semi-structured document allowing arbitrary elements to be defined of the form “name” equals “eval”, where “eval” can be evaluated to a basic type (number, boolean or string) from operations on values. A value can be a basic type or a reference to a named element either in this document or the matched document. As the documents do not have mandated specific named elements boolean tests can evaluate to true, false or undefined, with associated rules for how these states are combined through operations. Further details may be found in [12] and [13].

Though originally a local DRM system Condor is increasingly being used as a meta-scheduler for the Grid. This is achieved through submission of resource descriptions direct to the Collector, with additional ’grid’ information added such as type of remote grid system and the interface location. It is also necessary to replace the Shadow process on the execute system with a Grid-Manager that can deal with non-condor execution systems by communicating through the Grid ASCII Helper Protocol (GAHP) [18].

V. Grid Standards Architecture

In this section we describe how the standards used to describe both resource and task as well as task submission have been integrated with Condor. Figure 5 illustrates the general architecture with the three main components to support the Broker: advertising of resources, deployment of jobs and

file staging. The user submits a job to the broker using the standard Condor submission tools. However, for a complete Grid solution a BES/JSDL to Condor interface – such as GridSAM – could be placed in-front of Condor completing the encapsulation. Matched jobs in the GAHP server are translated from ClassAds into JSDL documents and then submitted to the remote OGSA-BES instance, in the diagram illustrated by the GridSAM client. Information about these remote resources is captured and described using the GLUE schema which can be stored in a persistent repository such as BDII or Grimories. The remote resource descriptions are retrieved as GLUE documents and then translated into ClassAds. These are then entered into the Condor Collector to enable matchmaking. In the rest of this section we outline in more detail these three main components of the architecture.

A. Resource Information Intergration

The resource broker must have up to date information fed to it on the status of the individual resources so that scheduling decisions are as accurate as possible, thus frequent harvesting of this information is essential. The standard method of achieving this is through aggregation into systems such as LDAP-based [25] information servers, e.g. BDII-based index servers or as XML based metadata within a UDDI-registry [**?**], e.g. Grimories [26]. We have developed a system, comprising of three main components, to gather information on resources from both static and dynamic information sources (Figure 6).

The first component is the information retriever, this interrogates both LDAP-based directory services and UDDI-based registries to collect information on the remote resources. This includes static information such as the number of CPU, memory size and installed DRM systems and dynamic information such as current queue length and number of free CPU. The type of information retrieved is defined in a mapping file which is derived from the version of the GLUE schema being used in each type of remote system and the type of directory service/registry. Information from the system Virtual Organisation Management System (VOMS) [**?**] server is also retrieved at this point. The second part – the Information Translator – translates the incoming documents, LDAP objects or XML, into the required output data format, Condor ClassAds, using static configuration rules. The final part is the writer/verifier, which writes the converted resource information into files. Before the information is written specific attribute values from

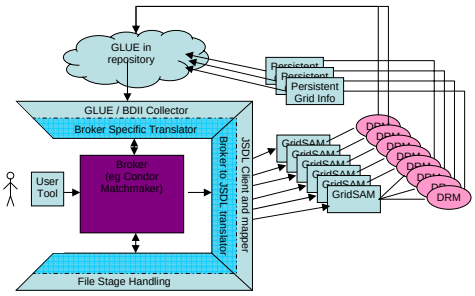

Fig. 5. The GridBS Architecture

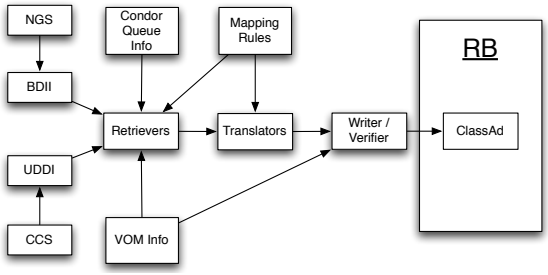

Fig. 6. Information Integration Architecture

the retrieved data are compared with statically retrieved values from the VOMS server. If the information matches then it is written into files in the specified format. In the case of a failure of verification no files are written and a flag is raised.

B. Publishing GLUE information from Microsoft HPC Server 2008

As Microsoft HPC Server 2008 (HPCS08) has no method to publish information in a standards compliant manner, we have provided our own system to perform this role. HPCS08 GLUE information publishing has been implemented using three packages: GLUE-Generation (GLUE-G), GLUE-Translation (GLUE-T) and GLUE-Publish (GLUE-P).

GLUE-G provides access to HPCS08 and retrieves resource information such as number of CPUs, memory, running job numbers. Information is constructed in a key-value form similar to GLUE and streamed to GLUE-T which translates it to an XML format according to the GLUE Schema. As these components are separate they can be developed in different languages – GLUE-G in C# and GLUE-T in Java – thus we can develop different GLUE-G implementations for other DRM systems while re-using GLUE-T. The GLUE-Publish package is a Java web service that addresses the aspects of publishing HPCS08 GLUE information to an XML-based UDDI registry.

C. Job Submission and Monitoring

Figure 7 illustrates our BES/JSDL GAHP integration within Condor. A new JSDL GAHP client side has been developed which runs within the Grid Manager. This is a thin client which passes the ClassAd, rendered in XML, to the GAHP server. The GAHP server, developed in Java, translates the ClassAd into an equivalent JSDL document which it submits to a BES enabled service. In order for the JSDL/BES GAHP server to know the name of the remote BES instance this information is

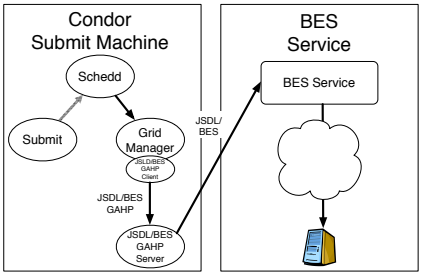

Fig. 7. The JSDL/BES-GAHP architecture

Command	Parameters	Result
JSDL_JOB_SUBMIT	<id> <classad>	<id> <S/F> <jobID> [<error string>]
JSDL_JOB_STATUS	<id> <jobID>	<id> <S/F> <classadR> [<error string>]
JSDL_JOB_RECOVER	<id> <classad>	<id> <S/F> <jobID> [<error string>]
JSDL_JOB_TERMINATE	<id> <jobID>	<id> <S/F> [<error string>]

TABLE I
GAHP COMMAND SET

passed in as part of the ClassAd in the Grid_Resource element. The structure of this element is "Grid_Resource = jsdl <URI> <Type>" where <URI> is the remote BES instance Uniform Resource Identifier (URI) whilst <Type> describes the BES flavour.

Table I lists the commands understood by the JSDL/BES GAHP protocol. As commands are asynchronous each contain an <id> element which is used to locally identify the job and is given by the Condor system job identity. For each command invoked an immediate return code is given indicating if the command was interpreted successfully or not. The remote identity of the job, <jobID>, is then recorded such that later commands may correctly communicate with the remote job. Job describing ClassAds are rendered in XML as single line strings and for JSDL_JOB_STATUS the returned ClassAd describes the status of the job as a Condor status. The status model for Condor and OGSA-BES are very similar and the mapping between these is given in Table II. As BES does not support the Held state this will never be returned by the server. The JSDL_JOB_RECOVER command is provided in case the Grid Manager needs to re-start. The original ClassAd is sent back to the GAHP server and allows the server to re-identify the job.

Condor Job Status	code	BES Job Status
I (Idle)	1	Pending
R (Running)	2	Running
X (Removed)	3	Terminated
C (Completed)	4	Finished
H (Held)	5	

TABLE II
MAPPING CONDOR TO BES JOB STATES

D. Data Staging

Condor uses elements within the ClassAd to list those files that need to be staged to and from the resource, this information is available to the GAHP server as it receives the whole ClassAd. In a JSDL document files are marked for transfer along with a URL indicating where the file should be staged from/to. If the location where the user has placed their files is already exposed through a file transfer protocol (such as FTP, GridFTP, HTTP(S)) then these locations can be used as the URL. If the files are not in exposed locations then the GAHP server itself will copy the files to a location where they can be accessed.

E. Evaluation

The framework developed is operating system and platform independent. It is able to collect information published in the GLUE schema from either BDII/LDAP or UDDI servers about remotely accessible resources. Incoming jobs are translated into standardised JSDL documents and submitted through a standardised (OGSA-BES) interface to the underlying Grid fabric. This allows us to exploit both the benefits of the Condor matchmaking service and the OGF recommended standards on job submission and resource description. This relieves the requirements imposed by many of the existing proprietary systems tying the user into a particular software stack. Since the system builds on the fundamentally strong basis of Condor, evaluation of the performance of the system is limited by that system.

We have built a prototype of the system on the Oxford Campus Grid system. To ensure compatibility with current grid standards as well as OGF recommended systems we have ensured the framework is compatible with all existing production level GLUE environments (GLUE 1.1–1.3). For nine months we have been brokering jobs to around 35 Grid and cluster resources including the UK NGS Grid services, Oxford University departmental Condor resources, Oxford Supercomputing Centre resources and Windows HPC resources. During this time our system has processed in excess of 271000 individual jobs smoothly and adapted to the different job requirements. This has allowed us to develop a whole Grid ecosystem which can make the basis of a campus/other Grid toolkit allowing quick and easy deployment of Grid systems through institutions.

Further evaluation will be completed through the systems ability to support the many current implementations of the OGF HPC Basic Profile that are being used within the demonstration organised by the Grid Interoperability Now community Group [**?**]. This work is currently leading towards a standards compliance system of which this developed framework will become a significant part.

VI. CONCLUSION

In this paper we have shown how the Condor system can be integrated with OGF recommended standards for job submission (JSDL and OGSA-BES) and resource description (GLUE). This allows the use of a powerful brokering service with resources exposed through standards based interfaces. This is a more scalable solution than using GAHP alone as different DRM systems need only develop a BES interface for their existing software rather than developing both a Client and Server interface to integrate with Condor alone. The added advantage is that as their software exposes a BES interface other Grid users can access these resources from standards based clients. Future work includes support for the new GLUE 2.0 standard for both BDII and UDDI and incorporating

the system into the current OGF Global HPC Basic Profile Interoperability demonstration.

ACKNOWLEDGEMENT

This work is funded by the Open Middleware Infrastructure Institute (OMII) UK through the GridBS project.

REFERENCES

[1] I. Foster and C. Kesselman, Eds., *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Jul. 98.

[2] S. Haubold, H. Mix, W. Nagel, and M. Romberg, *Performance Analysis and Grid Computing*. Kluwer Academic Publishers, 2003, ch. The UNICORE Grid and its Options for Performance Analysis, pp. 275–288.

[3] M. Litzkow, M. Livny, and M. Mutka, "condor-a hunter of idle workstations," in *8th International Conference on Distributed Computing Systems*, 1998, pp. 104–111.

[4] EGEE, "glite middleware," http://www.glite.org, Aug 2008.

[5] OGF, "The Open Grid Forum," http://www.ogf.org, May 2008.

[6] GLUE-WG, "The GLUE working group," https://forge.gridforum.org/sf/projects/glue-wg, May 2008.

[7] JSDL-WG, "Job submission description language," https://forge.gridforum.org/sf/projects/jsdl-wg, May 2008.

[8] OGSA-BES-WG, "Basic execution service," https://forge.gridforum.org/sf/projects/ogsa-bes-wg, May 2008.

[9] A. S. McGough, W. Lee, and S. Das, "A standards based approach to enabling legacy applications on the grid," *Future Gener. Comput. Syst.*, vol. 24, no. 7, pp. 731–743, 2008.

[10] Microsoft, "Microsoft compute cluster server," http://www.microsoft.com/windowsserver2003/ccs/default.aspx, May 2008.

[11] Maui Scheduler, "The maui scheduler," http://www.supercluster.org/maui.

[12] M. Solomon, "The ClassAd Language Reference Manual," *Computer Sciences Department, University of Wisconsin, Madison, WI, Oct*, 2003.

[13] M. S. Rajesh Raman, Miron Livny, "Matchmaking: Distributed resource management for high throughput computing," in *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, Chicago, IL., July 1998.

[14] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, Summer 1997.

[15] M. Ellert, M. Gronager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimaki, O. Smirnova, and A. Waananen, "Advanced resource connector middleware for lightweight computational grids," *Future Generation Computer Systems*, vol. 23, no. 2, pp. 219–240, February 2007.

[16] PBS Team, "Open Portable Batch System," http://www.pbsgridworks.com/PBSTemp1.3.aspx?top_nav_name=Products&item_name=OpenPBS&top_nav_str=1&AspxAutoDetectCookieSupport=1.

[17] Platform LSF, http://www.platform.com/products/LSF.

[18] The Condor Project, "Grid ASCII Helper Protocol," http://www.cs.wisc.edu/condor/gahp/.

[19] E. Huedo, R. Montero, and I. Llorente, "The GridWay framework for adaptive scheduling and execution on Grids," *SCPE*, vol. 2006, p. 2007, 2004.

[20] Grid Physics Network, "Grid physics network," http://www.griphyn.org/.

[21] The Australian Grid, "The APAC Grid," http://grid.apac.edu.au/.

[22] Open Grid Forum, "The High Performance Computing Profile Working Group (hpcp wg)," http://www.ogf.org/gf/group_info/view.php?group=hpcp-wg.

[23] A. Savva, "JSDL SPMD Application Extension," http://www.ogf.org/documents/GFD.115.pdf.

[24] W3C Consortium, "Web Services Description Language (WSDL) 1.1," http://www.w3.org/TR/wsdl.

[25] OpenLDAP Project, http://www.openldap.org.

[26] S. Wong, V. Tan, W. Fang, S. Miles, and L. Moreau, "Grimoires: Grid registry with metadata oriented interface: Robustness, efficiency, security," *IEEE Distributed Systems Online*, vol. 6, no. 10, 2005.