Condor Birdbath*

Web Service interface to condor

Clovis Chapman[1], Charaka Goonatilake[1], Wolfgang Emmerich[1,]
Matthew Farrellee[2], Todd Tannenbaum[2], Miron Livny[2],
Mark Calleja[3] and Martin Dove[3]

[1] Dept. of Computer Science, University College London,
Gower St, London WC1E 6BT, United Kingdom
[2] Computer Sciences Department, University of Wisconsin
1210 W. Dayton St., Madison, WI 53706-1685, U.S.A.
[3] Dept. of Earth Sciences, University of Cambridge,
Downing Street, Cambridge CB2 3EQ, United Kingdom

Abstract

The grid community has been migrating towards service-oriented architectures as means of exposing and interacting with computational resources across organizational boundaries. The adoption of Web Service standards provides us with an increased level of manageability, extensibility and interoperability between loosely coupled services that is crucial to the development of a grid infrastructure spanning multiple organizations and incorporating a wide range of different services. Providing support for Web Services in existing middleware and tools would ensure open interoperability with future mainstream grid developments. We cover in this paper the work that we have done in incorporating Web Service support into Condor – a widely adopted and sophisticated *high-throughput computing* software package, and present an overview of the motivations, implementation and achievements of this work. In order to demonstrate Condor's new capabilities, we also present work that we have done in adapting GridSAM, a Web-Service based job submission and monitoring system that endorses the emerging *Job Submission Description Language* (JSDL) standard, in order for it to interact with Condor through its Web Service API – as well as demonstrate the use of this combination of services to deploy real-world scientific workflows in the context of the e-Minerals project using the Web Service based workflow specification standard BPEL (Business Process Execution Language).

1. Introduction

Web Service technology has become an important building block in the design and development of a global grid infrastructure [1]. The ability to decompose resources and the functionality they provide into a set of discoverable and loosely coupled services, which are capable of interaction in heterogeneous environments addresses many of the interoperability issues that can be encountered in large scale grid infrastructures. Through conformance to an established set of XML based standards, such as the Simple Object Access Protocol (SOAP) for communication and the Web Service Definition Language (WSDL) for interface definition, Web Services ensure that independently developed applications and tools can be seamlessly integrated into a large-scale global grid environment.

Bringing current grid middleware and tools in line with these developments is a great opportunity to identify new means of interacting with these technologies and further exploit the capabilities of the resources they manage. The Condor system is a very good candidate for such an undertaking. It is a widely adopted job scheduling and resource management system that offers a wide range of well defined high-throughput computing services. By exposing its functionality as a set of well defined individual Web Services, we enable Condor managed resources to be seamlessly integrated into this emerging service oriented environment: allowing third parties to fully incorporate Condor's capabilities into their own applications, and considerably improve upon Condor's ability to operate across organizational boundaries.

Building on a previous investigation into exposing Condor services in a service oriented grid environment [2], we present here work that has been done in incorporating Web Service support into the Condor architecture, in the context of a project funded by JISC, DTI and Microsoft – support that has now been made part of Condor's latest development release (from 6.7.5 [3]).

* This work has been funded by the Department of Trade and Industry, the Joint Information Systems Committee and Microsoft

Condor is a mature and sophisticated software system, which provides an impressive set of capabilities spread across multiple *daemons*, each responsible for managing different aspects of Condor's functionality: job scheduling, resource allocation, meta-data collection, etc. This particular aspect of the Condor architecture has enabled us to clearly identify and incorporate Web service support into the daemons themselves, ensuring that core features such as multi-phase commit, transaction management and fault handling are maintained on an end-to-end setting. In this first stage, we have provided - as a set of complementary services - the necessary framework for external applications to submit and monitor jobs to a remote Condor scheduler, including the transfer of files to and from the scheduler and query a Condor pool for information about its various resources.

In order to both evaluate and produce a middle-tier service capable of fully taking advantage of Condor's Web Service capabilities – we have implemented a plug-in for gridSAM, a Web Service-based job submission and monitoring system. Currently being developed in the context of an OMII-funded project [4], gridSAM aims to serve as a standard interface for job submission to a number of different resource management and batch scheduling systems such as Condor, LSF and the Sun Grid Engine. It fully endorses the Job Submission Description Language (JSDL), an emerging GGF standard which aims to facilitate interoperability in heterogeneous environments, through the use of an XML based job description language that is free of platform and language bindings. Through our plug-in, we have aimed to demonstrate that the use of Condor's Web Service interface does not only considerably facilitate the development of applications capable of interacting with Condor, but can also bring significant new functionality to the system.

Using this combination of services, we have enabled scientists of the e-minerals project to deploy complex computational workflows on the e-minerals mini-grid, a cross-organizational production level grid infrastructure that incorporates a number of high performance and high-throughput computational and data storage resources. By relying on the Business Process Execution Language (BPEL) [5], a Web Service based workflow specification language, we have enabled scientists to not only specify the sequencing of jobs to be executed on a number of Condor-managed resources, but also to compose workflows incorporating a wide range of independent Web Service-based systems, such as data storage services, within their workflows.

2. Background

2.1 Condor

The Condor system [3] is a batch scheduling and high-throughput computing resource management system, which has been maturing over nearly two decades. It provides means for users to submit compute intensive jobs to a local scheduler in the form of batch executables, and manages the execution of these jobs on suitably selected resources in a pool of heterogeneous machines, based on job requirements and community, resource owner and workload distribution policies.

A feature that has made it particularly popular amongst the grid and the UK e-Science communities, is its ability to harness under-utilized computational resources: Condor can ensure that jobs submitted to a pool are run on idle machines. This particular aspect of its functionality has enabled the condor software to be deployed on existing computational infrastructures with limited impact to their everyday use. For instance, the UCL condor pool – part of the larger e-Minerals mini-grid (section 6) that we use here as our evaluation environment - consists of 940 machines spread over a number of student workstation clusters.

Condor provides a rich and varied range of services, which can be simplified into the following three categories:

- *Job scheduling:* Condor provides means to manage job execution requests as persistent queues of jobs, as well as coordinating and monitoring the remote execution of the jobs on the user's behalf. It provides means for users to specify and queue large number of jobs or specify workflow dependencies between jobs.
- *Resource management services:* A central manager is responsible for collecting resource characteristics and usage information from machines in a Condor pool. It is based on this collected information, and on user priorities, that job requests can be matched to suitable resources for execution.
- *Job execution management:* Based on matches obtained from the central manager Condor manages the remote execution of jobs on the selected resources. Condor provides the ability to checkpoint jobs – saving the state of a job on a regular basis –

ensuring that they can be migrated to other resources in case of failure. It also provides the ability to redirect system calls to the submission machine, as well as file transfer functionality to and from the execution site.

Condor's functionality has been compartmentalized into a number of individual *daemons*. Interaction between these daemons is illustrated in figure 1. Particular daemons of interest to us here are the following:

- condor_schedd: The Condor scheduler is responsible for maintaining a persistent queue of job execution requests and managing the remote execution of jobs. Jobs are maintained as *job classAds* – essentially a list of name/expression pairs that represent the various characteristics of a job (input files, arguments, executable, etc.) as well as its requirements and preferences (memory, operating system etc.). The scheduler has been adapted to provide client side job management capabilities for a number of other resource management systems, such as the Globus Toolkit and LSF (Condor-G) [6].
- condor_collector: The collector is responsible for maintaining meta-data about all resources and other daemons in a pool in the form of *resource classAds*, describing the various characteristics of the resource (memory, current load, Operating system, etc.).

2.2 Web Services

Whilst it is outside of the scope of this paper to cover in detail the inner-workings of Web Services, we briefly cover the capabilities and advantages this technology provides us with. Web Services are essentially a collection of XML-based protocols and standards, which define the ways in which services should be described (through the Web Service Description Language – WSDL [7]), how they can be accessed and how communications should be formatted (Simple Object Access Protocol – SOAP [8]), and finally how these may be discovered by client applications. The reliance on the eXtensible Markup Language (XML) as a common formatting language ensures a degree of platform, programming language and system independence.
As such clients and services developed independently can, through conformance to these standards, be made to interoperate.
There are several advantages to Web Services that should be noted here:

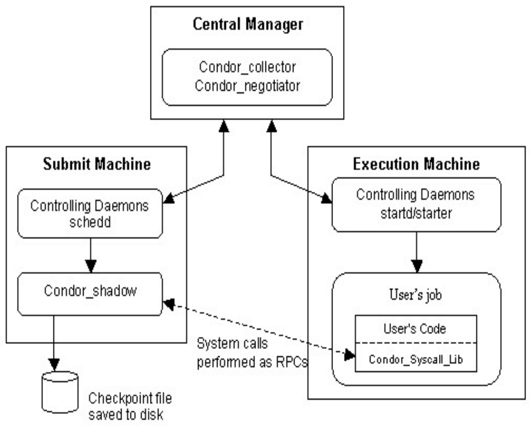

Figure 1: Condor Architecture overview

- *Loose coupling:* The use of published WSDL interfaces and XML-based communication protocols favours a 'black-box' approach to development. Most of the details of the inner-workings of the service are abstracted away from the client, which considerably facilitates the independent development of clients capable of invoking specific Web Services.
- *Development kits:* Numerous development kits (such as gSOAP, Axis, etc.) and transport protocol bindings are available, providing support for a wide range of platforms and programming languages. There are also several hosting environments available such as Tomcat, J2EE containers, which can be used to deploy and manage Web services.
- *Web Service compatible applications and standards:* As Web Services have become increasingly popular for business-to-business interaction, numerous industry tools and standards are available that can be exploited in a grid setting. For example, the industry standard BPEL can be used to compose interactions between grid services according to user defined workflows as we demonstrate in section 6.
- *Firewall management:* Often used as a 'selling point' for Web Service technology is its ability to traverse many proxies and firewalls unhindered. SOAP messages can be bound to the HTTP protocol; often allowed through firewalls for web browsing purposes. Whilst this may not necessarily be desirable from a security standpoint, it should be noted that Web Services enable session management over a single port, considerably facilitating the administration of firewalls.

3. Related Work

3.1 Alternative interfaces to Condor

A number of alternative techniques can be used to enable external applications to interact with Condor, whose limitations we cover in this section.

Command-Line wrappers: A commonly used technique is to create wrappers around the command line tools normally used by users to submit, monitor and manage their jobs, in order for these to be accessed programmatically by an external application - as is done, for example, within the Globus Toolkit GRAM (section 3.2). However the interface to the command line tools is naturally intended for human interaction and is not as rich as the interface to the Condor daemons themselves: core functionality such as fault-tolerance capabilities, multi-phase commits and transactions are not available through this interface and would have to be built on top of the tools by the third party developers. This would naturally not be as robust or as efficient as providing access to the daemons themselves: in this manner we can ensure that transactions and fault handling can be maintained on an end-to-end basis.

GAHP: The Grid ASCII Helper Protocol (GAHP) [9] provides a simple ASCII stream-based protocol for interaction with Condor and other resource managers. It provides richer functionality than the command-line tools such as more comprehensive error handling, multi-phase commit capabilities and support for transactions. Its ASCII based nature also ensures a certain degree of independence from software languages. However it lacks much of the functionality that is inherent in Web Services such as service and session management or type safety through declared WSDL interfaces, and was intended to “fill the gap” until Web Services become commonplace in grid computing .

DRMAA: The Distributed Resource Management Application API (DRMAA) [10] specification aims to define a standard API by which external applications can interact with resource managers such as Condor through local procedure calls. Condor provides, built on top of the command line tools, a DRMAA library that can be linked into external applications. However, apart from the fact that DRMAA is intended for direct interfacing, only a ‘C’ language binding is currently available for Condor and provides weak fault tolerance.

3.2 Alternative Web Service based Job Submission services

The latest version (v4.0) of the Globus Grid Resource Allocation manager (GRAM) [11] enables jobs to be submitted to a Condor scheduler through Web Services, or more specifically using the emerging Web Service Resource Framework - which provides on top of Web Services additional state and property management capabilities. The GRAM is intended to provide a standard job submission interface to multiple underlying resource managers, including Condor. As a generic interface, it hence cannot provide the full range of capabilities available in Condor, nor can it provide them as efficiently: a Condor specific interface ensures that job submission, queue management and other features are exposed in a way that conforms to its architecture and not through an additional layer of abstraction. However this does not imply that Condor Web Service interfaces are intended to supersede in any way the GRAM interface. These are by design complementary and can be combined to provide additional functionality, and we intend to demonstrate this through our implementation of a plug-in for GridSAM, which offers similar functionality to that of the Globus GRAM in section 5.

4. Web Service interfaces to Condor daemons

The primary services for which we have provided Web Service support are the Condor scheduler, and the Condor collector. These two services provide all the necessary functionality for third party applications and portals to interact with Condor according to a *job delegation model* (figure 2): external applications can query a pool collector to determine the type and availability of resources in the pool, and submit their jobs to a scheduler local to the pool so that it can manage the jobs on its behalf. However whilst these two daemons are the main sources of interaction with the system, all daemons have been embedded with some basic Web Service functionality in order to provide the necessary leeway for future extensions.
The strategy adopted has been to leverage existing APIs of these components as individual Web Services interfaces - supplementing Condor’s traditional mode of communication

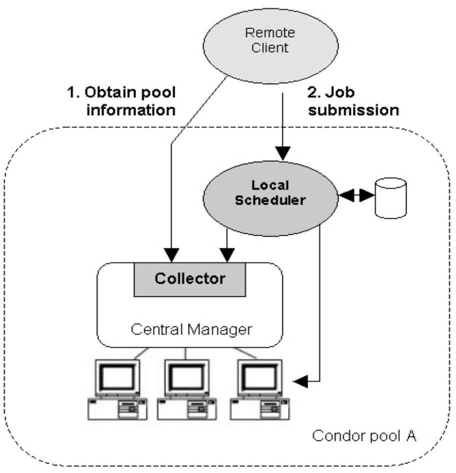

Figure 2: Job delegation to a local scheduler through Web services

(Cedar) in order for daemons to be capable of processing SOAP invocations. We have relied for this purpose on the gSOAP Web Services development kit [12].
This functionality is now present in the Condor development release (from 6.7.5) and can be enabled through Condor's configuration file.

4.1 The scheduler

The scheduler has been extended to provide the following operations through Web Services:

- *Job Submission:* The scheduler provides a number of operations for job submission. Submitting a job requires the creation of a new job cluster (group of jobs), new job id (within the cluster) and the definition of a job classAd, describing all its characteristics.
- *Utility functions:* Due to the complexity of a classAd specification, a template classAd can be obtained from the scheduler - pre-configured with respect to the scheduler environment. Further more a job requirement prediction function can be used to determine the files that are to be made available for a job submission to be successful.
- *Data transfer and management*: The scheduler also provides means to send input files and binaries to the scheduler and retrieve any output files produced through a simple chunk-based file transfer protocol. Files transferred are managed within individual "spools", ensuring that a user does not have to know exact locations of files on the remote host. Spool removal can be requested upon job completion.
- *Job monitoring:* Job classAds are continuously updated throughout the life of a job. They can be retrieved at any time from the scheduler, providing the user with information such as the current status of the job, errors that may have occurred, etc.
- *Queue management operations:* A number of additional queue management functions have been made available, such as the ability to cancel or hold jobs, request a rescheduling, etc.
- *Transaction based system:* Any set of actions described above can be performed as part of a transaction. If any of the operations specified within a transaction fail, the transaction as a whole is rolled back. Job submission for example may require the submission of a classAd and the transfer of one or more input files and binaries. Wrapping these actions in a transaction ensures that all steps have completed before the submission itself is considered as successful.

It should be noted that the scheduler's ability to manage submissions to the Globus Toolkit or other grid resource management systems (Condor-G), the ability to fork jobs on the resource hosting the scheduler (scheduler universe) or the use of other supported Condor universes (PVM, MPI) are fully accessible through these operations. Further more, the ability to checkpoint jobs and migrate jobs across multiple pools should also be available, primarily due to the fact that checkpoint data can be retrieved as a file from the scheduler.

4.2 The collector

As previously covered, the collector stores meta-data about every resource in the pool in the form *resource classAds*. This component has been extended in order for classAd to be retrieved through a set of query operations. Remote users may specify particular constraints to these queries (e.g. retrieve all classAds for all Linux execution machines with more than 512 Mb of RAM), enabling them to narrow the results to specific resources of interest to them.

5. Integrating gridSAM and Condor

5.1 GridSAM overview and objectives

The gridSAM job submission service [4] enables jobs specified using the Job Submission Description Language (JSDL) to be submitted to a wide range of underlying schedulers including Condor. JSDL essentially defines an XML based vocabulary and schema to describe requirements and characteristics of computational jobs – such as input/output files, arguments, resource requirements, etc. - for execution on grid resources – with the aim of facilitating interoperability between client applications, such as portals, etc. and resource managers and decoupling job specifications from actual execution environments. GridSAM attempts to leverage acceptance of this standard by providing a Web Service interface for the submission of a JSDL document and the underlying mechanisms through which this request can be transformed into a scheduler specific submission. This is achieved through the use of various *Distributed Resource Manager* (DRM) connectors, which can be 'plugged' into the system to provide platform specific job-launching capabilities. The previous incarnation of the Condor DRM connector relied on wrappers around command line tools to interact with the Condor system.

By producing a plug-in which enables interaction with Condor through its Web Service interface, our objectives are hence threefold: a) demonstrate the effectiveness of Web Services for the integration of Condor and third-party applications b) bring, through this channel, significant new functionality to gridSAM c) illustrate the ways in which different grid-based job submission services with similar modes of operations can be brought together.

Additionally, an outcome of this work is the development of a middle-tier service that will provide an additional layer of abstraction for clients. The Condor Web Service interfaces are themselves relatively low-level in nature: middle-tier services are required to fully take advantage of these capabilities and define the logic by which these functions are used - hence coordinating the distribution of jobs to one or more pools on behalf of a client application.

5.2 Condor Web Service DRM connector

The Condor WS DRM connector for gridSAM allows us to distinguish two levels of operation as illustrated in figure 3: in a first instance jobs are submitted to the gridSAM service through its provided mechanisms. These are then, through SOAP invocations, delegated to a remote Condor scheduler.

These latter invocations are managed within the Condor WS DRM connector plug-in, whose behaviour can be controlled by an administrator through gridSAM's configuration mechanisms.

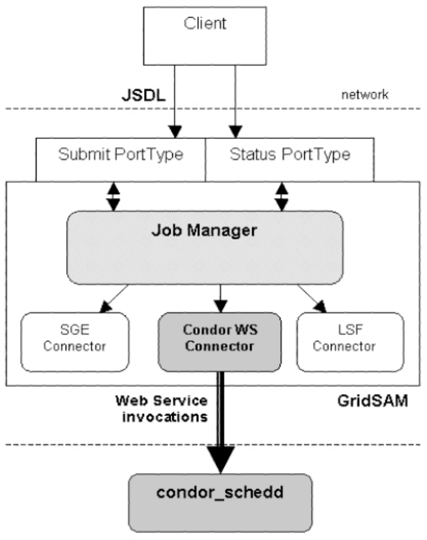

Figure 3: GridSAM architecture

GridSAM's use of HiveMind – an Apache development kit that allows run-time configuration and composition of the components – enables configuration attributes to be injected into our plug-in upon start-up, which we take advantage of here to allow administrators to specify a set of target schedulers - and any scheduler specific attributes and policies – to which gridSAM should relay incoming jobs.

The plug-in will monitor the remote execution of jobs through regular polling and, upon job completion retrieve any output files that have been produced.

One of the obvious drawbacks of this approach is that there is an additional overhead in transferring the job sandbox – including description and files - from the gridSAM host to the remote scheduler. However there are numerous advantages that clearly overweigh this inconvenience:

- *Independent deployment of GridSAM:* Though gridSAM would have normally been required to run on a submission machine local to a specific condor pool, it can now – due to the fact that it can access Condor schedulers remotely – be deployed completely independently, on a resource that does not require an actual condor installation. This ensures that gridSAM can be used to submit jobs to Condor resources without the need to involve the administrators of the resources. We can imagine for example a scenario where a Virtual Organization (VO) may deploy and manage its own gridSAM instances and use these for submissions to non-dedicated resources.

- *Submission to multiple schedulers:* An important feature of our plug-in is the ability for jobs submitted to gridSAM to be distributed to more than one Condor scheduler. The current version of our plug-in enables workloads to be shared between more than one pool or grid resource in a Virtual Organization in a round-robin style. Users can hence submit jobs without having to concern themselves with the actual location of the execution. This will provide an important basis for future research into workload distribution in cross-organizational infrastructures, such as improving workload sharing according to different policies of use.
- *Increased robustness and fault handling:* The Condor GridSAM plugin can track the remote progress of job executions through regular polling, and in cases where errors occur – due to losses in communication or scheduling errors – jobs can be restarted or rescheduled where possible. Condor transactions have been tied into gridSAM's own transaction system, used to manage state changes – ensuring consistency throughout the various stages of the execution.
- *Access to a wider range of Condor functionality:* The Condor WS DRM connector allows access to a wider range of Condor functionality, such as the ability to submit jobs to Condor-G or use PVM, MPI or any other Condor supported universes.
- *Job Specification:* Due to the fact that JSDL is intended as a standard job description language, it cannot support all the attributes that can be defined in a Condor classAd. We have implemented a set of generic libraries that provide all the basic JSDL to classAd mappings to be performed, as well as enabled administrators to specify in the gridSAM configuration file any additional classAd attributes and job policies that should be injected at run-time into the job descriptions for specific schedulers, in cases where special environmental considerations should be taken into account (e.g. condor-G submissions).

All these features contribute to making gridSAM a powerful instrument in the development of Virtual organizations, easing many of the administrative burdens often encountered in VO construction and facilitating the use of multiple independent resources.

6. Deployment of these services on the e-Minerals mini-grid

Finally, in order to demonstrate the use of these services to tackle scientific problems, we have deployed these on the e-Minerals mini-grid [13]: a production level grid infrastructure encapsulating a number of dedicated and contributed compute and data storage resources across six sites in the UK. The compute resources subset of the mini-grid consists of 4 16-node clusters, 2 IBM pSeries computers, a sunfire server, and the Cambridge and UCL Condor pools – the later consisting of more than 940 teaching machines, whilst the data storage subset encapsulates a number of high capacity Storage Resource Broker (SRB) vaults.

The objective has been to enable scientists to execute complex computational workflows, consisting of a number of inter-related jobs, relying on gridSAM and Condor to handle the coordination and distribution of jobs across our infrastructure. For this purpose we have relied on the Business Process Execution Language (BPEL): a Web Service based workflow specification language that allows a series of Web Service invocations to be composed as a single integrated process. whose usefulness as a means of orchestrating scientific workflows has been investigated in previous work [14].

We have used for this purpose the BPEL editor that is being developed at UCL as part of an OMII funded project, which aims to make BPEL more accessible to scientists by enabling them to specify workflows using a graphical user interface that abstracts most of the complexities of writing BPEL documents.

The specific use case we have tackled here is the management and distribution of calculations required to determine, in a systematic way, the mechanisms by which pollutant molecules such as DDT, dioxins and biphenyls, become bound to soil minerals [15]. Using BPEL, we have specified the sequencing of several hundreds of jobs to be executed on our resources, which were submitted to a central gridSAM node – responsible for distributing through our Condor plug-in them to our various computational resources.

From a usability perspective, this has proved to be a very successful exercise: as users do not need to concern themselves with the actual location of the execution of the jobs, workflows can be completely free of any workload distribution concerns. Further more, submissions were not limited to actual Condor pools but also to our Globus managed resources (through the use of Condor-G). Finally it should be noted that this deployment required no changes to our existing environment.

The use of BPEL potentially allows for any Web Services to be incorporated into a workflow. For example, the SRB data management system that we use to handle our data storage need in the e-minerals mini-grid should soon provide a Web Service interface. This would allow Condor executions to be linked to data storage operations, such as enabling input data and output data for our computations to be retrieved and stored back into our storage vaults. We are in the process of defining - using BPEL - a set of similar reusable patterns that correspond to various scenarios of use repeatedly encountered in the e-minerals project.

7. Conclusion and future work

Web Services prove to be a valuable technology in allowing users to aggregate a wide range of computational services according to their own requirements. As a growing number of grid services adopt Web Services, these can be, through technologies such as BPEL, composed into user-defined workflows and used alongside Condor Web Services to provide a single unified grid service.

We have currently enabled job submission and monitoring to be achieved through Web Services, but Condor provides many more services such as accounting and execution management that we hope to eventually make available through these same means.

Security features are also in development. Condor daemons will initially support SOAP over SSL connections and mappings to local accounts through X.509 certificates.

We also hope to take advantage of the capabilities of new emerging Web Service based standards, such as the Web Service Resource Framework (WSRF) or WS-notification – which extend current Web Service standards to provide state management, meta-data querying and asynchronous notification. As we now have an established set of submission services, and used these in practice, we hope that the WSRF and WS-notification extensions will allow us to refine particular aspects such as the ability to use notifications as opposed to polling for job state monitoring.

We have provided some initial groundwork in this domain by incorporating - through gSOAP's support for WS-addressing; on which the WSRF framework relies on to define service endpoint references.

References:

[1] Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002.

[2] Chapman, C., Wilson, P., Tannenbaum, T., Farrellee, M., Livny, M., Brodholt, J., and Emmerich, W., Condor Services for the Global Grid: Interoperability between OGSA and Condor, in *Proc. Of the All Hands Meeting 2004*, Nottingham, 2004.

[3] The Condor Project. *http://www.cs.wisc.edu/condor*

[4] The gridSAM project. *http://www.lesc.ic.ac.uk/gridsam/*

[5] Andrews, T., et al., Business Process Execution Language for Web Services Version 1.1. OASIS, *http://ifr.sap.com/bpel4ws*

[6] Livny, M., Tannenbaum T., Thain, D. Condor and the Grid, in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley, 2003.

[7] Christensen, E., et al., Web Services Description Language (WSDL) 1.1. W3C, Note 15, March 2001

[8] Box, D., et al. Simple Object Access Protocol (SOAP) 1.1. W3C, Note 8, 2000.

[9] Grid ASCII Helper Protocol *http://www.cs.wisc.edu/condor/gahp/*

[10] Distributed Resource Management Application API Working Group. *http://www.drmaa.org/*

[11] The Globus Project. *http://www.globus.org/*

[12] The gSOAP development kit *http://www.cs.fsu.edu/~engelen/soap.html*

[13] Blanshard, L., Brodholt, J., Bruin, R., Calleja, M., Chapman, C., Dove, M., Emmerich, W., Kleese van Dam, K., Tyer, R., and Wilson, P., Grid tool integration within the e-Minerals project, in *Proc. Of the All Hands Meeting 2004*, Nottingham, 2004.

[14] Emmerich, W., Butchart, B., Chen, L., Wassermann, B., and Price, S., Grid Service Orchestration using the Business Process Execution Language (BPEL), UCL-CS. Research Note RN/05/07.

[15] Chapman, C., Wakelin, J., Artacho, E., Dove, M., Calleja, M., Bruin, R., and Emmerich, W., Workflow issues in atomistic simulations, in *Molecular Simulations*, Taylor and Francis Ltd. 2005.