Proceedings of the UK e-Science All Hands Meeting 2006, © NeSC 2006, ISBN 0-9553988-0-0

Simple Grid Access using the Business Process Execution Language*

Clovis Chapman[1], Andrew M. Walker[2], Mark Calleja[3],
Richard P. Bruin[2], Martin T. Dove[2] and Wolfgang Emmerich[1]

[1] Dept. of Computer Science, University College London,
Gower St, London WC1E 6BT, United Kingdom
[2] Dept. of Earth Sciences, University of Cambridge,
Downing Street, Cambridge CB2 3EQ, United Kingdom
[3] Cambridge eScience Centre, Centre for Mathematical Sciences,
University of Cambridge, Wilberforce Road, Cambridge CB3 0EW

Abstract

Scientists require means of exploiting large numbers of grid resources in a fully integrated manner through the definition of computational processes specifying the sequence of tasks and services they require. The deployment of such processes, however, can prove difficult in networked environments, due to the presence of firewalls, software requirements and platform incompatibility. Using the *Business Process Execution Language* (BPEL) standard, we propose here an architecture that builds on a *delegation* model by which scientist may rely on middle-tier services to orchestrate subsets of the processes on their behalf. We define a set of inter-related workflows that correspond to basic patterns observed on the *e*Minerals minigrid. These will enable scientists to incorporate job submission and monitoring, data storage and transfer management, and automated metadata harvesting in a single unified process, which they may control from their desktops using the *Simple Grid Access* tool.

1. Introduction

As grid infrastructures evolve and offer an increasingly important amount of resources and services, it becomes clear that scientists are suffering from physical and software restrictions not addressed by current-generation grid middleware and tools. Scientific processes, such as those defined by the scientists of the *e*Minerals project [1], involve a large number of services and resources, such as job execution services, data stores, visualisation services, etc., and potentially complex interactions between these. The realities of networked heterogeneous environments make the deployment of such processes an extremely difficult task: firewalls, platform incompatibility, software and resource requirements, security, etc. - all these elements can stop the average user from launching and coordinating complex computational processes from their desktops and/or applications of their choice. Ensuring that scientists can retain their current working environments and applications, with minimal or no change, is key to facilitating the transfer to grid environments and enabling the step change in the research that this provides. Existing grid middleware can prove difficult to install, configure and use, and hardly provides the level of integration required to seamlessly incorporate a wide range of services into a unified process.

We have had the opportunity to work on the deployment of several scientific workflows on the *e*Minerals minigrid [2]. While the scientific goals of these processes may differ, they present several similarities and common requirements. Specifically, they require batch job submission and monitoring primitives, the ability to store and retrieve large amounts of produced data and finally the ability to organize and index this data through the definition of corresponding metadata.

The approach that we adopt here is to decompose large scientific processes into a collection of basic patterns that can be fully automated and delegated to third party VO-wide systems, which will orchestrate the execution of these subsets of the process on the user's behalf.

We propose an architecture that builds on this delegation model, relying on the *Business Process Execution Language (BPEL)* [3] and other Web Service tools and standards, such as *GridSAM* and the *Job Submission Description Language (JSDL)*

* Research presented has been funded by NERC through Grant reference numbers NER/T/S/2001/00855, NE/C515698/1 and NE/C515704/1 (*e*Minerals).

[4], to provide VO-wide orchestration and service provision. The increased adoption of Web Services by the Grid community makes the use of this industry-led standard in a Grid environment very appealing: BPEL is an orchestration language, enabling us to build services whose role is to coordinate interactions between Web Services according to specified workflows on a client's behalf.

The architecture relies on the definition and deployment of a number of predefined workflows that correspond to basic patterns observed in our minigrid. Scientists may trigger the execution of these workflows remotely through a *lightweight* self-contained client.

For this purpose, we have implemented the *Simple Grid Access* (SGA) tool: a lightweight, self-contained java-based tool, that enables users to launch job executions and manage submissions from their desktop, including transferring files to and from storage vaults and uploading proxy certificates. SGA can be used directly or incorporated within other applications as an external process, providing users with means of composing more complex workflows at a more abstract level; with applications, tools and languages of his choice – such as simple batch scripting. Alternatively, focus on reusability ensures that users can incorporate our workflows into larger BPEL workflows.

2. Deploying workflows on the eMinerals minigrid

2.1 Scientific workflow requirements

Scientific problems will require the definition of computational processes usually involving the execution of several data management and computational tasks. As an example, we refer to previous work [6], where we tackled the problem of distributing the computations required to study the adsorption of pollutant molecules on a pyrophyllite surface on the e-Minerals minigrid.

The eMinerals is a production level infrastructure encapsulates a wide range of resources across 6 sites in the UK.

It provides essentially 3 categories of services:

- *Compute resources*: consisting of number High Performance Computing (HPC) and High Throughput Computing (HTC) resources. These are typically fronted by Globus 2.4 [5]
- *Data Storage resources:* we rely on the SDSC *Storage Request Broker* (SRB), to provide seamless access to distributed data resources, with a total capacity of 3 terabytes spread across 5 storage vaults.
- *Metadata annotation service (Rcommands):* This service [10] provides means of storing, generating and searching metadata for files stored in SRB vaults to facilitate the indexation and re-use of data.

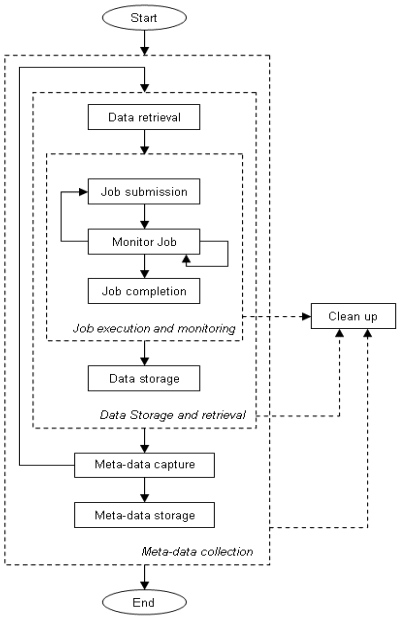

Figure 1: Hierarchical workflow patterns

The number of resources and their geographical distribution clearly highlights the difficulty in deploying and managing complex workflows on such an infrastructure.

The various sites are independently administrated and protected by institutional and/or departmental firewalls. Users also require a certain number of tools, at the very least the Globus toolkit, Condor-G (for job scheduling capabilities), the RCommands front end tools and finally SRB tools for data storage and retrieval. Installation, configuration and maintenance can prove tedious, particularly where administrative access to the machine is not available.

However, a key observation that we have made during our work on workflow deployment is that scientific processes can be decomposed into a number of basic patterns, which take advantage of our services in a unified manner. For each individual job, we require:

- The selection of a target compute resource

- The retrieval and staging of data from our storage resources onto the target resource
- The submission and execution of the job
- The storage of the produced data
- Finally, the automated harvesting of metadata from the produced output, and its storage.

As part of the process, these various stages require fault-handling capabilities and monitoring of state changes. This in itself constitutes a relatively complex but reusable workflow unit (illustrated in figure 1) that can be interleaved with desktop processes, such as retrieving user input for steering purposes, or input and output processing.
The complexity and resource requirements of such a process means that its management and execution is best left to a third party with sufficient resources, for which we rely on the Business Process Execution Language (BPEL).

2.2 The Business Process Execution Language

BPEL aims to enable the orchestration of Web Services, by providing means of composing a collection of Web Service invocations into an executable workflow, itself presented as an *invocable* Web Service.
Much work has been invested in the development of Web Service compliant grid middleware. For example, web service based job submission tools such as GridSAM have been developed, providing a standard job interface to underlying resource management systems such as Condor.
BPEL allows us to specify the process to execute upon request as an XML document. It provides support for controlled flow elements, including sequential or parallel executions, conditional executions, variables, fault handling and XPath and XSLT queries.
Once a workflow has been specified, a BPEL engine will be responsible for orchestrating the workflow on a user's behalf, acting as a middleman between resources and client. Such an approach also provides us with the advantage that the BPEL workflow can be modified independently from the users, considerably easing administration and enabling the transparent addition of new resources and services. Web Services also provide better support for firewalls, by allowing session management over a single port.

3. Implementation

Our implementation is illustrated in figure 2. For job specification, we rely on the Job Submission Description Language (JSDL), an emerging XML based GGF standard [9].

Orchestration Service: At the core of our system we rely on one or more BPEL engines to provide and manage the execution of workflows as specified in section 2.1. We use for this purpose the open source Active BPEL engine to deploy workflows. These workflows, deployed as executable Web Services, will present an interface for clients to submit JSDL documents and corresponding data requirements, and, upon receipt of such a document, will manage the invocations of the various required services.
The Active BPEL engine also provides graphical Web based monitoring tools, providing a detailed view of the execution process, which will ensure that users can check on the progress of their jobs, whilst keeping the client-side tools as light as possible.

Simple Grid Access (SGA) tool: The SGA tool is our self-contained client tool that we have implemented to allow users to specify their job requirements and generate corresponding JSDL and data requirements specifications for submission to a BPEL engine.
We favor here a simple command line interface, which will allow users to specify the various requirements of the job (input files, executables, etc.) as a sequence of arguments, as well as allowing additional attributes (proxy server descriptions etc.) to be obtained from a configuration file, in a predefined location or from local environment settings.
The client also provides additional data staging capabilities. Because client side inbound connections are rarely available, it enables users to upload required files to the SRB vaults before job submission if required through HTTP - as we will explore below, as well as making files to be made available through a variety of means (FTP, HTTP). Files can also be returned to the user's desktop upon job completion. Finally, it also allows users to generate a certificate proxy, which will be uploaded to a credential management service (i.e. myProxy [13]) of the user's choice. Once data and proxy requirements have been handled, the client will invoke the orchestration services to orchestrate the execution of the workflow.
The client has been implemented in Java, ensuring that it will be compatible with most platforms, using the CoG kit [11], Apache Axis [12], Apache HttpClient and other libraries.

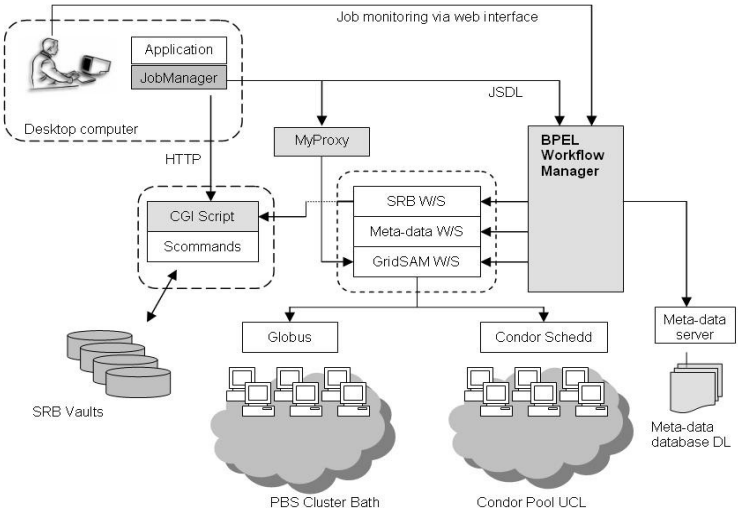

Figure 2: Implementation

Meta-scheduling: We rely here on previous work in which we created a plug-in for the GridSAM job submission service to support submissions to multiple Condor pools, or Globus managed clusters through Condor's Web Service interface [8], effectively transforming GridSAM into a complete metascheduling service. GridSAM also provides support for JSDL, and myProxy, facilitating its integration into the system.

Data Transfer and services: While Web Services are used to control the data transfers, the actual transfer of data is handled through HTTP, in order to avoid the overhead of SOAP based invocations.
For this purpose, we have written a front end wrapper over the SRB tools using CGI that facilitates interaction with the SRB data vaults through HTTP.
Alongside this, we have created a Web Service that will facilitate the remote coordination of transfers between third parties and can be deployed on target resources. The reasons for adopting HTTP are obvious: they facilitate the upload of files for clients in the presence of firewalls, and also enable them to access files through their browser.

Metadata collection: The interface provided by the RCommands Web Service provides means of adding, editing and removing new data sets and studies associated with a particular file, and its physical location (on the SRB).
In addition to basic submission details, detailed information about the data can be obtained by parsing the output files, particularly in XML or CML [7]. We have created an additional metadata collection Web Service that will be responsible for identifying elements of interest given an appropriate RDF file and a suitably rich ontology, relying on the AgentX framework [10].

4. Conclusion

We have begun the process of incorporating the SGA tool into existing workflows and tools used by our scientists.
In particular, we have incorporated SGA into GDIS (GTK Display Interface for Structures), an open source program that allows scientists to visualise molecules, crystals and crystal defects in three dimensions and at the atomic scale [14]. GDIS can act as an interface to create code input and run the simulations on the local machine. By enabling GDIS to invoke our SGA tool, we can now launch calculations onto our minigrid, and take advantage of all our services, with little change to the original code base.

References

1. Dove, M. et al., Environment from the molecular level: an escience testbed project. *Proc. of All Hands Meeting, Nottingham, 2003*
2. Blanshard, L. et al., Grid tool integration within the eMinerals project, in *Proc. Of the All Hands Meeting* Nottingham, 2004.
3. Andrews, T. et al., Business Process Execution Language for Web Services Version 1.1 OASIS, 2003. *http://ifr.sap.com/bpel4ws.*
4. The GridSAM project. *http://www.lesc.ic.ac.uk/gridsam/*
5. The Globus Project. *http://www.globus.org/*
6. White, T. et al., *e*Science methods for the combinatorial chemistry problem of adsorption of pollutant organic molecules on mineral surfaces, in *Proc. Of the All Hands Meeting,* Nottingham, 2005.
7. Murray-Rust, P. et al., Chemical Markup Language and XML Part I. Basic principles, *J. Chem. Inf. Comp. Sci.*, **3**9, 928, 1999.
8. Chapman, C. et al. , Condor Birdbath - web service interface to Condor, in *Proc. of the All Hands Meeting*, Nottingham, 2005.
9. Anjomshoaa, A., et al., Job Submission Description Language Specification v1.0, GFD #: GFD-R-P.056, 2005.
10. Tyer, R. P., et al., Automatic metadata capture and grid computing, in *Proc. of the All Hands Meeting*, Nottingham, 2006.
11. Java Cog Kit http://wiki.cogkit.org/
12. Apache Software Foundation. http://www.apache.org/
13. MyProxy http://grid.ncsa.uiuc.edu/myproxy/
14. Fleming, S., and Rohl, A., GDIS: a visualization program for molecular and periodic systems, *Z. Krist. 200 580*, vol.220 pp.580-584, 2005.