An approach to encapsulation of Grid processing within an OGC Web Processing Service

Andrew Woolf (andrew.woolf@stfc.ac.uk), Arif Shaon (arif.shaon@stfc.ac.uk),
STFC e-Science Centre, Rutherford Appleton Laboratory, Oxon OX11 0QX, UK

Abstract

Increasingly complex environmental problems and the growth in accessible geospatial data in interoperable formats is leading to the investigation of Grid processing as a potential tool for advanced geo-processing within a standards-based environment. The OGC Web Processing Service (WPS) is an open standardised interface for geo-processing, compatible with technologies being adopted by spatial data infrastructures. We investigate the integration of Grid computing capability within a WPS. Our approach is based on the use of the Job Submission Description Language (JSDL) being developed by the Grid community as a standardised description of computational jobs to be executed on heterogeneous Grid platforms. We develop a mechanism to integrate JSDL resource requirement descriptors within a standard WPS request, and show how process execution and status querying may be proxied to a Grid environment through the WPS interface. A proof-of-concept implementation is developed using an atmospheric particle tracing application and the UK National Grid Service.

KEYWORDS: Grid, OGC, Web Processing Service, WPS, Job Submission Description language, JSDL, spatial data infrastructure

1 Introduction

The Geospatial community requires an ability to analyse and process geographic datasets using a combination of spatial software and analytical methods. Increasingly complex geo-processing operations are driving a demand for greater computing capability – it is no longer always possible to rely on a desktop GIS to perform spatial analysis of interest, especially where very large datasets are concerned, or the results of complex simulation models need to be integrated.

There is also a need to be able to share processing and analytical capability across the community in an interoperable fashion. A research group having developed a new analysis technique may wish to make it available to the broader research community, or the owner of a high-performance computing resource may wish to allow use by collaborators for a defined purpose.

Within the context of emerging developments in spatial data infrastructures (SDIs, for instance INSPIRE in Europe), there is a significant potential for Grid computing to play a major role (Padberg, Kiehle 2009). The three key elements of SDIs are metadata, interoperable data, and network services. ISO 19119 (ISO 2006) provides a wide taxonomy of relevant services, including model/information management services, workflow/task management services, and processing services (for metadata, spatial, thematic and temporal processing). Within all of these service categories, Grid computing could play a role. For instance, the Grid community is very active in the area of workflow and service chaining, and dynamic information services that expose the current state of the Grid 'fabric' are especially relevant for real-time geospatial information, e.g. from environmental sensors. For standard geospatial data services (e.g. WFS, WCS), the suite of Grid data management middleware offers possibilities to facilitate integration, for example by combining data from multiple heterogeneous databases, and transforming to GML representations. Conversely, the use of geospatial data and services within Grid infrastructures can be useful for 'e-science' and other advanced computing applications (SEE-GEO 2008).

2 The problem

The recent development of web services provides a powerful technical solution to simplify the sharing of computational resources and algorithms. Software may be exposed for use through simple web-based protocols, and chains of such services may be orchestrated in value-adding workflows. These 'service-oriented architectures' provide a new paradigm for enabling collaboration.

To date, most web-based geo-processing services take a traditional 'RESTful' 'stateless' view of resources: data are not distinguished from their access service; asynchronous interaction sequences are poorly supported;

there is no notion of resource types (e.g. computational) other than data and service instances; there are no efficient means for handling resource-intensive processes. For example, an application for predicting future global climate change may involve analysing large volumes of past weather data collected over a number of years, and running compute-intensive forecast models. Such a complex application will require a large amount of computational resource, such as disk space, memory and CPU power. Executing this application through a standard web service allows no scheduling capability, and could utilise all available computational resources, resulting in significant delays for other processes.

In addition, typical geo-processing web services do not take a sophisticated approach to security-related issues associated with the underlying processes and datasets. In fact, service providers often rely on ad-hoc access control at the client level to ensure security of the resources exposed. This leads to non-interoperability in workflows that require interaction between multiple services, each with different security protocols.

The aforementioned limitations are precisely the niche of Grid computing (Foster, Kesselman 2004). While Grid architectures and technologies vary, they share in common an attempt to abstract models of stateful resource (data, storage, compute, etc.) within a standardised framework (Foster et. al. 2002) in order to simplify the construction on demand of complex workflows. For instance, using Grid technology, execution of a large-scale parallel process may be accelerated by load-balancing across different Grid nodes (Thain et. al. 2003). In addition, Grid middleware enables allocation of specific amounts of computational resource, such as disk space, to a particular process (Huedo et. al. 2004). In terms of access control, many Grid architectures employ a common security framework (e.g. the Grid Security Infrastructure (GSI) (Foster et. al. 1998)) to provide secure, robust and interoperable communications. Authentication in the GSI architecture is based on a public-key infrastructure (PKI) with x.509 certificates containing information for identifying users or services.

From this perspective, geoprocessing applications and services should benefit from integration with Grid computing resources and technologies to enable applications to scale out. This goal is reflected in a Memorandum of Understanding (MoU) on collaboration signed in late 2007 (OGC 2007) by the Open Geospatial Consortium (OGC (OGC 2009)) and the Open Grid Forum (OGF, the principal standards body for Grid computing (OGF 2009)). The initial targets of the collaboration are: integration of the OGC Web Processing Service (WPS) with Grid processing and workflow tools, and integration of geospatial catalogues with Grid data movement tools. From a wider perspective, it is intended to promote the use of Grid-based resources within the Geospatial community.

3 The principles of Grid-enabling WPS

The OGC standard Web Processing Service (WPS (Schut 2007)) interface is designed to facilitate publishing, discovery and use of geospatial computational processes in a standardised and interoperable manner. The WPS interface is implemented as a Web Service, with operations for: describing process functionality in terms of inputs and outputs, triggering its execution, monitoring its status and finally retrieving its output.

An analysis of WPS and Grid processing paradigms indicate that both provide a remote interface for invoking computational processes. WPS functionality includes remote data inputs/outputs, progress monitoring, and asynchronous delivery. Beyond this, Grid job submission mechanisms typically add the ability to stage data, and specify required computational resource requirements (Woolf 2006).

The conceptual overlaps and differences between WPS and Grid processing paradigms can also be demonstrated by a comparison between the WPS specification and the Job Submission Description Language (JSDL, (Anjomshoaa et. al. 2005)) (Figure 1), an OGF specification that provides a standardised description of a computational job and the resources (data and computational) it requires. JSDL provides a normative XML schema for describing a computational job, including job description and identification, the software application to be run, resource requirements (filesystem parameters, disk space, operating system, CPU, etc.), and data staging instructions for input and output. It also includes standardised POSIX extensions for executable filename, command-line arguments, etc. The purpose of JSDL is to enable a standardised specification of job submission to Grid infrastructures irrespective of the back-end schedulers or resource

managers used. JSDL job descriptions may be submitted to consuming services like GridSAM (GridSAM 2009).

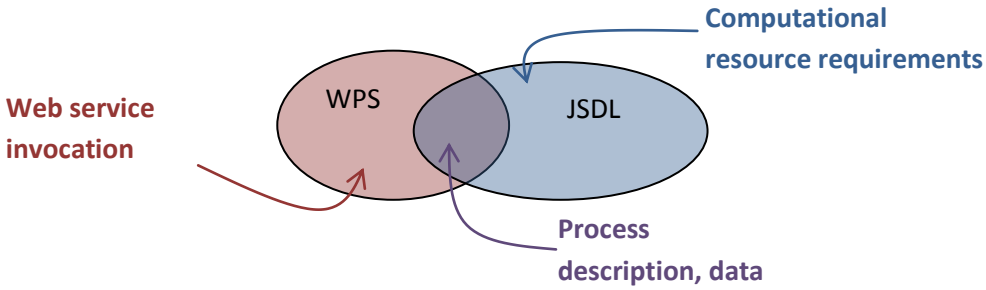

Figure 1: Conceptual overlaps and differences between WPS and JSDL

At the specification level, both WPS and JSDL include process description information (WPS: Identifier; JSDL: JobIdentification/JobName), and process outputs and inputs (WPS: DataInputs; JSDL: POSIXApplication/{Input,Argument,Output}, DataStaging/Source). These analogous parameters are sufficient to produce a valid JSDL document from a standard WPS request; this is fundamental to the approach presented in this paper. Thus, standard WPS data input and output parameters may be used to generate equivalent JSDL DataStaging or POSIXApplication/Input elements. Similarly a WPS process identifier may be mapped to a JobName and POSIXApplication in JSDL. What WPS lacks are any standard input parameters for specifying required computational resources. Vice versa, while JSDL enables the specification of a computational job and its resources, it lacks a standardised web service interface for enabling the submission of processing requests. Our solution combines the unique aspects of WPS and JSDL, using the overlaps to maximise the integration of WPS in a Grid context.

4 Existing Approaches

The most common approach to grid-enabling WPS (e.g. (Di et. al. 2002), (Nativi et. al. 2009), (Baranski et. al. 2008)) involves encapsulating Grid "characteristics" within a standard WPS instance without compromising its compliance with the OGC specification WPS (Schut 2007). This approach is often referred to as "profiling" of WPS. The nature of the encapsulated Grid "characteristics" varies between implementations. However, only a very few instances of the existing WPS-Grid profiling approaches incorporate the ability to specify computational Grid resources (disk space, CPU power, memory) needed to run an application. For example, this has been used in (Baranski 2008) to enable executing geospatial processes on a Grid backend through a standard WPS server. However, the ability to specify computational resource for a process is restricted to a few pre-defined JSDL specific parameters (e.g. DiskSpace, TotalCPUCount). In addition, it is not possible to query the states associated with a process except its execution status. Future WSRF-based approaches may enable other aspects of Grid 'state' to be encapsulated.

5 A WPS-Grid Profile

We have developed a WPS-Grid "profiling" solution that combines the simple web service interface of WPS with the ability of JSDL to specify resource requirements for a large computational process. The WPS-Grid profile enables Grid computing resources to be encapsulated behind a WPS: the WPS acts as an interface to the backend computing resources on the Grid.

In contrast to the existing approaches, our WPS-Grid profile is achieved by enabling encoding of JSDL-related information directly as part of the WPS Execute request defined in version 1.0 of the WPS specification (Schut 2007). The JSDL-related information is then used by the WPS instance for constructing a JSDL document, and submitting the process for execution to a Grid backend through a JSDL-enabled Grid client, such as GridSAM (Figure 2). The rationale of this approach is to incorporate JSDL handling ability into a WPS while ensuring its conformance to the WPS specification. This allows Grid computing capability to be combined with the simplicity of the WPS interface. Thus, this approach improves the mechanism adopted in previous work, by providing users with the flexibility of specifying the computational resource requirements for a process.

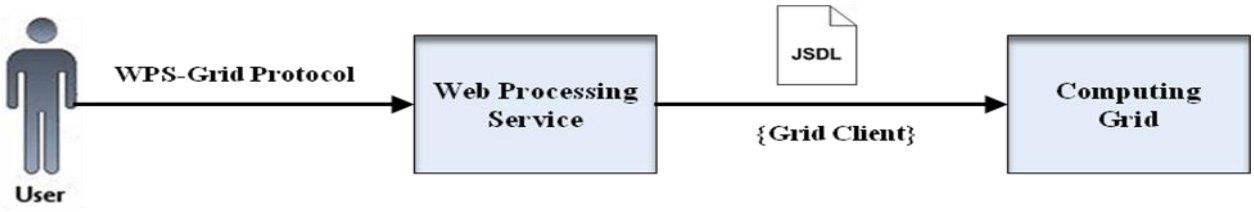

Figure 2: WPS-Grid generates JSDL document for submission to Grid

5.1 WPS Execute request

Two broad options are available for including JSDL-related information within the WPS Execute request:

- *where JSDL-related parameters are regarded as conceptually distinct from other WPS input parameters*: through a specific 'JSDL' parameter as part of the WPS DataInputs
- *where JSDL-related parameters are regarded simply as additional WPS input parameters*: through individual WPS DataInput parameters

We outline in the next two sections (5.1.1 and 5.1.2) these alternative approaches to specifying JSDL input parameters in a WPS Execute request.

5.1.1 Specific 'JSDL' input parameter

Typically, JSDL-related WPS input parameters will be concerned with specifying resource requirements for executing the process on a computing Grid backend, e.g. required disk space, CPU time, etc. Conceptually, such parameters are not process-related; they only determine whether, and perhaps how quickly, a result is computed. The output results themselves are independent of these parameters. There is a strong case, therefore, to regard the JSDL-related input parameters as different in nature to other process-related WPS input parameters. In this case, a specific 'JSDL' WPS input parameter may be used to specify resource requirements. The use of this special parameter 'JSDL' enables a WPS instance to distinguish between JSDL parameters and the other (process-related) parameters.

JSDL input parameters may be supplied in a conformant XML format, or in a proposed microformat, as described in the next two sections respectively.

5.1.1.1 Full JSDL document or valid snippet

In this case, a URL may be provided referencing a complete pre-created JSDL document, or valid snippet (e.g. specifying just the JSDL 'Resource' XML elements), Listing 1.

Advantages of this approach are that the JSDL input may be validated against the JSDL schema.

```
http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=WeatherGenerator&DataInput=other_
inputs=xxx;JSDL=http://www.foo.com/myfoo.jsdl@Format=text/xml&storeExecuteResponse=true&status=true
```

Listing 1: WPS Execution request with URL to JSDL document

Alternatively, the JSDL document or snippet may be URL-encoded and included directly within the WPS request, Listing 2.

```
http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=WeatherGenerator&DataInput=other_
inputs=xxx;JSDL=%3CJobDefinition%3E%3CJobDescription%3E%3CResources%3E%3CTotalDiskSpace%3E%3CExact%3E5
%3C%2FExact%3E%3C%2FTotalDiskSpace%3E%3CTotalCPUCount%3E%3CExact%3E1%3C%2FExact%3E%3C%2FTotalCP
UCount%3E%3C%2FResources%3E%3C%2FJobDescription%3E%3C%2FJobDefinition%3E@Format=text/xml@Schema=h
ttp://server/jsdl.xsd&storeExecuteResponse=true&status=true
```

Listing 2: WPS Execution request with URL-encoded JSDL snippet

Note that such a JSDL document or document fragment may also be supplied very naturally through a HTTP POST request (Listing 3), facilitating the service invocation in XML-based workflows.

```
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
..\wpsExecute_request.xsd">
        <ows:Identifier>WeatherGenerator</ows:Identifier>
        <wps:DataInputs>
                <wps:Input>….</wps:Input>
                <wps:Input>
                        <ows:Identifier>JSDL</ows:Identifier>
                        <wps:Data>
                                <wps:ComplexData encoding="" schema"http://server/jsdl.xsd">
                                        <jsdl:JobDefinition
                                        xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
                                                <jsdl:JobDescription>
                                                        <jsdl:Resources>
                                                                <jsdl:TotalDiskSpace>
                                                                        <jsdl:Exact>5</jsdl:Exact>
                                                                </jsdl:TotalDiskSpace>
                                                                < jsdl:TotalCPUCount>
                                                                        < jsdl:Exact>1</ jsdl:Exact>
                                                                </ jsdl:TotalCPUCount>
                                                        </ jsdl:Resources>
                                                <jsdl:JobDescription/>
                                        </jsdl:JobDefinition>
                                </wps:ComplexData>
                        </wps:Data>
                </wps:Input>
        </wps:DataInputs>
        <wps:ResponseForm>
                <wps:ResponseDocument storeExecuteResponse="true" status="true"/>
        </wps:ResponseForm>
</wps:Execute>
```

Listing 3: WPS Execution request with HTTP-POST JSDL snippet

5.1.1.2 JSDL elements microformat

As an alternative to providing a full valid JSDL snippet, JSDL-related parameters may be represented more simply as key-value pairs, with the keyword deriving directly from the relevant JSDL element name. An Xpath-like syntax can be used (replacing '/' with '#') to specify the JSDL element name.

These key-value-pair JSDL parameters are specified in a microformat as the value of the complex WPS DataInput parameter, 'JSDL' (Listing 4). They are enclosed within square brackets [], with the WPS 'Format' attribute set to the special value 'text/kvp'. (Note that the entire URL must be URL-encoded as per IETF RFC 1738, although for clarity the examples in the listings below are left unencoded.)

```
http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=WeatherGenerator&DataInput=other_
inputs=xxx;JSDL=[TotalDiskSpace=5;TotalCPUCount=1]@Format=text/kvp&storeExecuteResponse=true&status=true
```

Listing 4: WPS Execution request with microformat for JSDL input parameters

As an even simpler alternative to specifying full Xpath JSDL element names, simplified keywords may be designated for predefined common JSDL elements (e.g. 'Source_URI' for the JSDL 'DataStaging/Source/URI' XML element).

Multiple values for a repeatable JSDL parameter are separated using "," (Listing 5).

```
http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=WeatherGenerator&DataInput=other_inputs=xxx;JSDL=[DataStaging#Target#URI=http://www.foo.com/myfoo.xml,http://www.foo.com/myfoo2.xml]@Format=text/kvp&storeExecuteResponse=true&status=true
```

Listing 5: WPS Execution request with multiple JSDL element values

5.1.2 Individual JSDL input parameters

Rather than regarding JSDL parameters as special, they may be regarded as simply additional WPS literal input parameters individually. In this case, they may be included as key-value pairs in a manner similar to that described in section 5.1.1.2 above, except as individual WPS parameters instead of within an aggregate microformat 'JSDL' parameter (Listing 6).

```
http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=WeatherGenerator&DataInput=other_inputs=xxx;TotalDiskSpace=5;TotalCPUCount=1&storeExecuteResponse=true&status=true
```

Listing 6: WPS Execution request with individual JSDL input parameters

5.2 Progress monitoring

A JSDL-enabled WPS is also able to monitor the status of the process on the Grid, for instance using the same Grid client used for the job submission (e.g. GridSAM). The response to a WPS Execute request contains an element indicating the status of the request (process accepted, started, paused, succeeded, or failed). In addition, for longer-running asynchronous requests, the response may include a URL reference that can be polled for status updates. The status returned by monitoring job progress with the Grid middleware may be mapped to the WPS status indicator. For instance, GridSAM is a Grid middleware component that accepts JSDL job submission requests for execution on a Grid. It may also be used to monitor job progress, returning a status of: pending, staging-in, staged-in, active, executed, staging-out, staged-out, done, failed, and undefined. These may be mapped to the abovementioned WPS status codes to enable progress monitoring in the standard WPS manner.

5.3 Generation of JSDL document

The generation by the WPS server of a conformant and appropriate JSDL document for submission to a Grid backend is implementation-defined. Typically, an internal configuration would assign a default JSDL to a specific WPS process, with user-supplied JSDL parameters over-riding the defaults. As mentioned earlier, in fact a conventional WPS request contains sufficient information to generate a minimal compliant JSDL document, even without the Grid-specific WPS extensions proposed above. Such a job description, of course, would include only job identification information and data inputs; it would not include parameters specifying computational resource requirements. An XSLT approach may be used to generate a JSDL document from a template based on WPS request parameters (Baranski 2009).

5.4 Security

Secure access to OGC web services is the subject of considerable ongoing work. Rather than develop a divergent solution, a very lightweight 'placeholder' approach has been taken to security within the WPS-Grid profile. We allow a user to embed MyProxy (Basney et. al. 2005) parameters (host, port, username, password) within the WPS request using the microformat encoding mechanism described earlier (5.1.1.2), as the value of a special DataInput parameter, 'MyProxy' (Listing 7). These are processed by the WPS server and used at job submission for authentication.

```
http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=WeatherGenerator&DataInput=otherinput@datatype=string;JSDL=[TotalDiskSpace=5]@Format=text/kvp;MyProxy=[username=xxxx;password=xxxx;host=xxx;port=xxxx]@Format=text/kvp&storeExecuteRepose=true&status=true
```

Listing 7: WPS Execution request with MyProxy input parameters

Including sensitive information, such as a user's MyProxy credentials in a WPS request does pose security risks. Therefore, this approach assumes that an implementation would incorporate an appropriate transport-layer security protocol (e.g. SSL) to ensure the security of MyProxy information in the WPS request.

6 Implementation

A Grid-enabled WPS service has been implemented within the OGC's OWS-6 activity (Baranski 2009) as a proof-of-concept using an atmospheric particle-tracing 'trajectory service' (BADC 2009) and deployment on the UK National Grid Service (UK NGS (NGS 2009)), which is explicitly mentioned as a target Grid infrastructure in the OGC-OGF MoU. The implementation of this Grid-enabled WPS is fully compliant with the OGC WPS specification 1.0.0, and uses Python as the underlying programming language and Pylons (Pylons 2009) as the integrated web development framework. At an architectural level, it depends on a number of other services and components (Figure 3) to enable invocation and controlling of Grid-based processes through standard WPS requests. We outline below some of the key components.

6.1 Grid-enabled WPS

The implemented WPS conforms to the Grid profile described in this paper, using a specific 'JSDL' input parameter (5.1.1) for JSDL-related information, and using the simplified microformat (5.1.1.2) key-value-pair syntax. A HTTP GET binding was implemented for WPS requests, although a SOAP wrapper was also developed (see 6.3 below).

The deployed application ('trajectory service') makes use of very large four-dimensional (space and time) gridded fields of analysed atmospheric parameters (windspeed and direction, pressure, humidity, etc.) to trace the trajectory of a hypothetical particle released at a given location and time. While such a service is not suitable for full dispersion modelling it was used in a demonstration scenario to simulate the dispersal of a plume of toxic gas released in an emergency incident (Baranski 2009). The overall scenario integrated the trajectory service with other services (including a plume service for simulating the gas plume from a calculated trajectory) using a workflow engine.

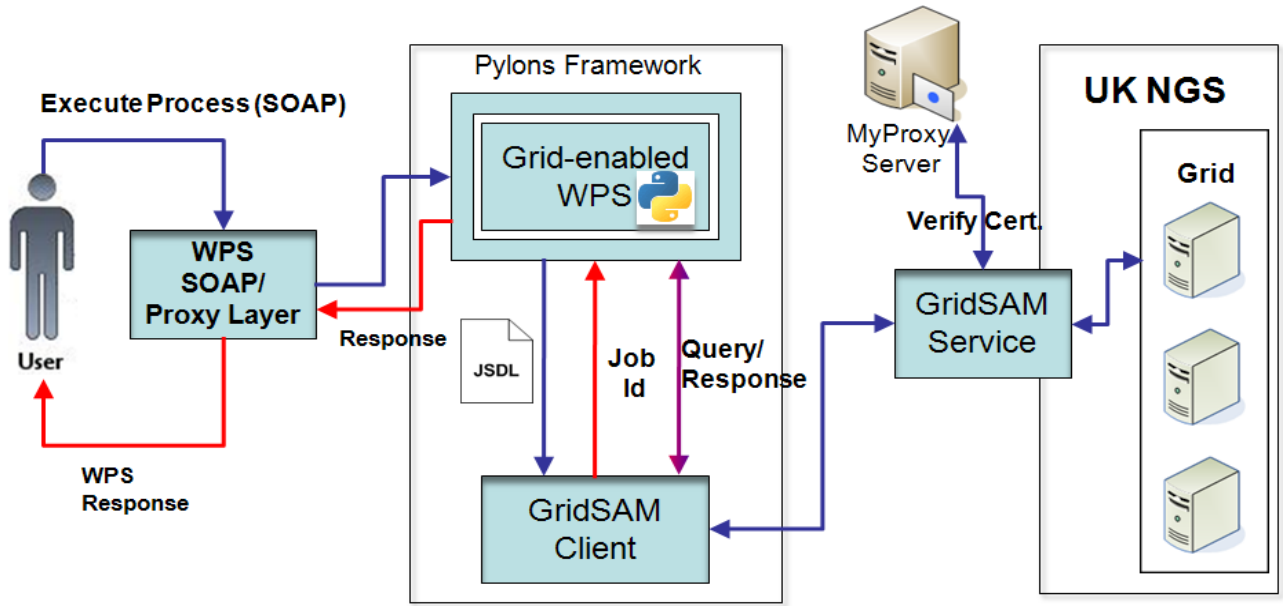

Figure 3: An Architectural View of the Grid-enabled WPS

6.2 GridSAM Service

The aforementioned GridSAM client (5.2) submits the JSDL job description to the UK NGS GridSAM service (installed in the prototype on the Oxford node of the UK NGS), which then executes the requested job on the NGS. This service also receives queries (for example job status check, etc.) from the GridSAM client and responds to them by liaising with the Grid with which it is associated. The GridSAM service is also responsible for retrieving and verifying users' Grid certificates using the MyProxy credentials received along with the JSDL job description from the GridSAM client.

6.3 WPS SOAP / Proxy Layer

We also implemented a Java-based WPS SOAP/Proxy layer that provides a SOAP wrapper outside the network firewall to proxy SOAP requests through HTTP GET requests to the Grid-enabled WPS. The SOAP interface provided by the SOAP wrapper conforms to the WPS specification 1.0.0. In addition, this layer is also used for forwarding other HTTP GET requests to the Grid-enabled WPS, such as a request for downloading process output and status check requests for a process.

7 Conclusions & Future Work

Grid computing provides an efficient means of executing resource-intensive processes, and thus should be beneficial to the Geospatial community that has an increasing need for performing highly complex geo-processing operations involving large geographical datasets. The WPS-Grid profiling approach presented in this paper demonstrates the feasibility of integrating Grid capability within standard geo-processing web services, such as the Web Processing Service. However, there is considerable scope for further work in this area. For example, the suite of OGC web services could be refactored around a resource-oriented view of data (Woolf, Woodcock 2005) within the Grid infrastructure using technologies such as the Web Services Resource Framework (WSRF (Banks 2006)). WSRF standardises the representation of stateful resources within web services. Applied to the Web Processing Service, for instance, computational process instances could be regarded as resources, with associated state corresponding to current job status, computational resource utilisation, etc.

It may also be useful to look into replacing GridSAM as the grid middleware for consuming JSDL documents and job management on the Grid, with any middleware conforming to the 'HPC Basic Profile' (Dillaway et. al. 2007), subject to associated restrictions on the allowable scope of JSDL. The HPC Basic Profile defines a base level of interoperability for high-performance computing systems within a Grid environment, by specifying a restricted use of several OGF specifications, notably JSDL for job description, and the 'OGSA Basic Execution Service' (Foster et. al. 2008) for job scheduling and management. A number of core JSDL elements must be supported; these include job description elements (JobIdentification, JobName, JobProject) and computational resource elements (CandidateHosts, ExclusiveExecution, OperatingSystem, CPUArchitecture, TotalCPUCount). These core elements are obvious candidates to standardise as simplified keywords within the WPS-Grid profile, as described earlier (5.1.1.2), avoiding the JSDL XPath syntax.

Finally, future evolution of the WPS-Grid profile will need to align with best practice for adding security to other OGC service interfaces, possibly by harmonising the Grid Security Infrastructure (GSI) and the current OGC approaches to security.

AUTHORS
Dr Andrew Woolf (andrew.woolf@stfc.ac.uk)
Environmental Informatics lead
STFC e-Science Centre
Rutherford Appleton Laboratory
Chilton, Oxon., OX11 0QX
UNITED KINGDOM

Dr Arif Shaon (arif.shaon@stfc.ac.uk)
Software Research Scientist
STFC e-Science Centre
Rutherford Appleton Laboratory
Chilton, Oxon., OX11 0QX
UNITED KINGDOM

REFERENCES

Anjomshoaa, A.; Brisard, F.; Drescher, M.; Fellows, D.; Ly, A.; McGough, S.; Pulsipher, D.; Savva, A. (2005): "Job Submission Description Language (JSDL) Specification, Version 1.0", OGF document GFD-R.056. Available at: http://www.gridforum.org/documents/GFD.56.pdf, Accessed 08/09.

BADC (2009): "BADC Trajectories - Main Page". http://badc.nerc.ac.uk/community/trajectory, Accessed 08/09

Banks, T. ed. (2006): "Web Services Resource Framework (WSRF) – Primer v1.2", OASIS document wsrf-primer-1.2-primer-cd-02. Available at: http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf, Accessed 08/09.

Baranski, B. (2008): "Grid Computing Enabled Web Processing Service". In: Pebesma, E., Bishr, M. & T. Bartoschek (Eds.): *GI-Days 2008 - Proceedings of the 6th Geographic Information Days*. IfGIprints Nr. 32, Institut für Geoinformatik, Münster.

Baranski, B. ed. (2009): "OGC® OWS-6 WPS Grid Processing Profile Engineering Report", OGC document 09-041r2.

Basney, J.; Humphrey, M.; Welch, V. (2005): "The MyProxy online credential repository", In: *Software: Practice and Experience*, **35**(9), 2005, 801-816. See also http://grid.ncsa.uiuc.edu/myproxy, Accessed 08/09.

Di, L.; Chen, A.; Yang, W.; Zhao, P. (2002): "The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data". Available at: http://laits.gmu.edu/~achen/download/Liping_Aijun.pdf, Accessed 08/09

Dillaway, B.; Humphrey, M.; Smith, C. (ed.); Theimer, M.; Wasson, G. (2007): "HPC Basic Profile, Version 1.0 ", OGF document GFD-R-P.114. Available at: http://www.ogf.org/documents/GFD.114.pdf, Accessed 08/09.

Foster, I.; Kesselman, C.; Tsudik, G.; Tuecke, S. (1998): "A Security Architecture for Computational Grids", 5th ACM Conference on Computer and Communications Security, 1998, 83-91.

Foster, I.; Kesselman, C.; Nick, J.; Tuecke, S. (2002): "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Globus Project, 2002. Available at: http://www.globus.org/research/papers/ogsa.pdf, Accessed 08/09.

Foster, I.; Kesselman, C. eds. (2004): *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan-Kaufmann, 2004.

Foster, I.; Grimshaw, A.; Lane, P.; Lee, W.; Morgan, M.; Newhouse, S.; Pickles, S.; Pulsipher, S.; Smith, C.; Theimer, M. (2008): "OGSA Basic Execution Service Version 1.0", OGF document GFD-R.108. Available at: http://www.ogf.org/documents/GFD.108.pdf, Accessed 08/09.

GridSAM (2009): "GridSAM - GridSAM - Grid Job Submission and Monitoring Web Service". http://gridsam.sourceforge.net, Accessed 08/09.

Huedo, E.; Montero, R.S.; Llorente, I.M. (2004): "A Framework for Adaptive Execution on Grids", In: *Software - Practice and Experience*, **34**(7), 2004, 631-651.

ISO (2006): *Geographic information – Services*, ISO 19119, International Organisation for Standardisation, 2006.

Nativi, S.; Verlato, M.; Pacini, F.; Pugliese, G.; Mazzetti, P.; Angelini, V. (2009):" G-OWS: the Grid-enabled Open-Geospatial Web Services", In: *Geophysical Research Abstracts*, **Vol. 11**, EGU2009-3424, 2009. Proceedings of the *EGU General Assembly 2009*, Vienna, Austria. Available at: http://meetingorganizer.copernicus.org/EGU2009/EGU2009-3424.pdf, Accessed 08/09.

NGS (2009): "National Grid Service". http://www.ngs.ac.uk, Accessed 08/09.

OGC (2007): "Open Grid Forum and OGC Sign Memorandum of Understanding". http://www.opengeospatial.org/pressroom/newsletters/200801#C5, Accessed 08/09.

OGC (2009): "Welcome to the OGC Website | OGC®". http://www.opengeospatial.org/, Accessed 08/09.

OGF (2009): "Open Grid Forum". http://www.ogf.org/, Accessed 08/09.

Padberg, A.; Kiehle, C. (2009): "Towards a grid-enabled SDI: Matching the paradigms of OGC Web Services and Grid Computing", GSDI 11 World Conference, Rotterdam, June, 2009. Available at: http://www.gsdi.org/gsdi11/papers/pdf/174.pdf, Accessed 08/09.

Pylons (2009): "PylonsHQ – Home". http://pylonshq.com/, Accessed 08/09.

Schut, P. ed. (2007): "OpenGIS® Web Processing Service", OGC document 05-007r7. Available at: http://portal.opengeospatial.org/files/?artifact_id=24151, Accessed 08/09. See also http://www.opengeospatial.org/standards/wps, Accessed 08/09.

SEE-GEO (2008): "See Geo Overview". http://edina.ac.uk/projects/seesaw/seegeo, Accessed 08/09.

Thain, D., T. Tannenbaum, and M. Livny (2003): "Condor and the Grid". In: Berman, F.; Hey, A.J.G.; Fox G. (Eds.): *Grid Computing: Making the Global Infrastructure a Reality*, Wiley, 2003.

Woolf, A. (2006): “Wrappers, portlets, resource-orientation and OGC in Earth-System Science Grids”, OGC Technical Committee meeting, Edinburgh. Available at: http://portal.opengeospatial.org/files/?artifact_id=15966, Accessed 08/09.

Woolf, A.; Woodcock, R. (2005): “Grid-enabling OGC: profiling against WSRF”, Workshop on Grid Middleware and Geospatial Standards for Earth System Science Data, National e-Science Centre, Edinburgh, UK. Available at: http://epubs.cclrc.ac.uk/bitstream/1523/OGCWSRF.ppt, Accessed 08/09.