Grid middleware for engineering design search and optimisation

G. E. Pound, M. H. Eres, M. J. Fairman, G. Xue, A. J. Keane and S. J. Cox

e-Science Centre,
School of Engineering Sciences,
University of Southampton,
Highfield, Southampton, S017 1BJ, UK
{gep, eres, mjf, gx, ajk, sjc}@soton.ac.uk

Abstract

Design search and optimisation algorithms can be used by engineers to yield improved designs. Design search involving the analysis of the aerodynamic properties of a design, using Computational Fluid Dynamics, is both computationally and data intensive, making this problem well matched to Grid computing. Evaluation of the quality of a design (the objective function) may require commercial and user supplied software packages to be called in sequence, data transferred to and from suitable compute resources, in addition to pre- and post-processing. To allow engineers to express these necessarily complex workflows we expose a suite of Grid-enabled tools to a high-level scripting language. These tools include client functionality to Globus compute resources, and to a job submission Web service which exposes a cycle-scavenging Condor pool. These tools are exposed as functions to the Matlab environment that can be used directly by the engineer, or intergrated into higher level functions for design search and optimisation. Here the benefits of the scripting approach are discussed, together with details of the Grid middleware used and the implementation of the client functionality.

1 Introduction

The aim of the Geodise project [1] is to deliver a Grid-enabled Problem Solving Environment (PSE) to assist engineers to improve the aerodynamic characteristics of their designs. PSEs are software environments designed to assist the user to solve a target class of problems, whilst concealing the complexities of interacting with the underlying systems [2].

To investigate the aerodynamic properties of interest engineers may analyse a suitable model using Computational Fluid Dynamics (CFD) methods. CFD analysis concerns the behaviour of fluids around objects, for example the lift produced by the flow of air over the wing of an aircraft.

Given a suitable model, design search and optimisation algorithms automate the evaluation of the qualities of a design that are of interest to the engineer. Common optimisation techniques, such as Genetic Algorithms, are used to determine improved combination of design parameters by minimizing (or maximizing) some measure of the quality of a design - called the objective function.

The size and complexity of industrial aerodynamic designs means that large scale design search and optimisation is only viable with the performance available from distributed computing. The Grid offers a framework to share and utilise resources [3] such as the large-scale compute and data resource required for engineering design search. In the context of industrial engineering the Grid has the potential to facilitate project oriented collaboration between individuals distributed between industrial partners and contractors.

We faced a number of challenges in the development of a tool to aid engineering design search and optimisation. User must be assisted to couple all of the packages required to evaluate the quality of their design. These include commercial packages, legacy and user-defined codes; these applications may all require specific platforms or have restrictive licensing requirements.

The dynamic integration of the compute and data resources required for numerous and detailed CFD calculations must be supported. Because of the cost of lengthy CFD calculations it is also valuable to archive and re-use data wherever possible [4].

Of course, efficient optimisation methods are also required, but the user should also be aided to select algorithms suitable for their problem [5]. All of this functionality must be exposed in a flexible design environment that will facilitate engineers in exploring and understanding their designs.

In this paper we outline an interface between the user and the Grid middleware used to support Grid-enabled CFD analysis. It is upon this generic interface which the tools required by the domain specific PSE for design search and optimisation can be built.

In section 2 we describe the use of scripting within the Geodise project to support the complex and multilayered user-defined workflows required for engineering design search and optimisation. We then describe the middleware used to expose computational resources to the engineer. These include Globus resources (section 3) and a Condor pool exposed via a job submission Web service (section 4).

2 Scripting the Grid

The engineer applying design search and optimisation methods to improve the aerodynamic performance of a design must automate the evaluation of the properties of the design. Analysis using CFD typically requires that a parameterised geometry be rendered from the design variables, this geometry is then used to produce a computational mesh, followed by the CFD analysis itself. Therefore each analysis may require several software packages or user-defined codes to be called in sequence.

Additionally pre- or post-processing of data may be required at several points during this workflow. The engineer may wish to place constraints upon validity of design points, for example concerning the structural integrity of a design. Each constraint may therefore require calls to additional software packages at each design point. The workflows required for the evaluation of constraints and objective functions are therefore often peculiar to a user's problem.

In the development of a PSE for engineering design search and optimisation we require a medium to capture and execute these complex workflows. A common solution to the workflow problem that has been adopted when building Grid computing environments has been to use an XML workflow language (such as WSFL or BPEL4WS) to invoke components in sequence [6].

This approach is sufficient for simple workflows. However for the diverse and complex workflows involved in design search and

optimisation we needed a more expressive representation than is possible with current XML workflow languages. Rather we use a high-level scripting language to express user-defined workflows.

Scripting languages are typically interpreted, and often use a grammar and syntax similar to a human language (so-called fourth-generation languages). They are frequently used to support the rapid development of applications, allowing user's to *glue* components together [7]. In addition scripting languages typically support a great deal of high-level functionality which increases the ease of development of scripts that encapsulate the logic of the user's workflow.

By adopting scripting languages we have taken a pragmatic approach which delivers the greatest flexibility. Potentially, scripts could be generated automatically by a GUI, with support from Geodise knowledge services, but these scripts could also be edited and reused by expert-users. We consider this flexibility to be important since ideally the engineer should not be limited by the tools with which they are provided.

Scripting languages occupy a point in a trade-off between quick application development, and the capture and reuse of the valuable logic contained within the script.

By using the constructs available within modern scripting languages, such as exception handling, the user is easily able to cater for a wide range of conditions, including the failures that are unfortunately often a feature of both CFD and Grid computing. The top-level PSE becomes as flexible as the scripting language that it supports.

We support the Matlab scripting language [8] which is widely used within the engineering community, and therefore familiar to many of our users. The language is accessible, but is sufficiently expressive to describe these workflows. The Matlab environment for technical computing provides a large number of toolboxes, including those for visualisation and data analysis. Matlab also integrates seamlessly with Java classes that contain the client functionality to our Grid-services.

In this way we supply the basic building blocks required by the engineer for design search and optimisation on the Grid. The basic functionality required to utilise computational and data resources on the Grid is supplied though several Matlab toolboxes.

3 Computational Toolbox

The Geodise computational toolbox provides certificate management, job submission and file transfer functionality as a collection of Matlab functions. These functions are designed to be consistent with the look and feel of the Matlab environment. They provide access to Globus v2.x resources, and are built upon the Java CoG that provides a Globus client API. This toolbox allows us to exploit the ubiquitous Globus compute resources which at present are the backbone of most computational Grids, for example the UK e-Science Grid [10].

The Globus toolkit v2.x [9] provides middleware that allow the composition of computational grids through the agglomeration of resources which are exposed as Grid services. This middleware provides much of the functionality required by our toolbox including authentication and authorisation, job submission, data transfer and resource monitoring and discovery.

Client software to Globus Grid services exists natively on a number of platforms, and also via a number of Commodity Grid (CoG) kits that expose Grid services to commodity technologies [11]; including Java [12], Python [13], CORBA and Perl. By using client software to Grid services written for these com-

`gd_createproxy`	Creates a Globus proxy certificate for the user's credentials
`gd_destroyproxy`	Destroys the local copy of the user's Globus proxy certificate
`gd_certinfo`	Returns information about the user's certificate
`gd_proxyinfo`	Returns information about the user's proxy certificate
`gd_proxyquery`	Queries whether a valid proxy certificate exists
`gd_jobsubmit`	Submits a compute job to a Globus GRAM jobmanager
`gd_jobstatus`	Gets the status of a Globus GRAM job
`gd_jobpoll`	Queries the status of a Globus GRAM job until complete
`gd_jobkill`	Kills the Globus GRAM job specified by a job handle
`gd_putfile`	Puts a remote file using GridFtp
`gd_getfile`	Retrieves a remote file using GridFtp
`gd_rmfile`	Deletes a remote file using GridFtp
`gd_makedir`	Creates a remote directory using GridFtp
`gd_rmdir`	Deletes a remote directory using GridFtp

Table 1: Functions of the computational toolbox.

modity technologies the developer of a PSE is able to remain independent of platform and OS. This independence motivated us to develop the Geodise toolbox over the Grid service client APIs of the Java CoG kit.

To expose the functionality available from the Java CoG to the Matlab user it was important to present functions which are consistent with the behaviour and syntax of the Matlab environment. This low level functionality may be used interactively by the user, but these functions are also incorporated programmatically into the higher level components of the toolbox.

The set of compute functions in Table 1 describes the minimum functionality required to allow the user to run jobs on Globus compute resources. The functions may be loosely categorised into those concerned with the user's credentials, job submission to the Globus Resource Allocation Manager (GRAM), and file transfer.

The Grid Security Infrastructure (GSI) [14] used by the Globus toolkit is based upon the Public Key Infrastructure (PKI) [15]. Under the PKI an individuals identity is asserted by a certificate that is digitally signed by a Certificate Authority within a hierarchy of trust. In an extension to this standard the GSI allows a user to delegate their identity to remote processes using a temporally limited proxy certificate signed by the user's certificate. The toolbox command `gd_createproxy` allows users to create a Globus proxy certificate within the Matlab environment, essentially creating a point of single sign-on to the Grid resources that the user is entitled to use.

The `gd_jobsubmit` command allows users to submit compute jobs to a GRAM job manager described by a Resource Specification Language (RSL) string [16]. The `gd_jobsubmit` command returns a unique job handle which identifies the job. The job handle may be used to terminate or query the status of the user's job. In addition the `gd_listjobs` command may be used to query a Monitoring and Discovery Service (MDS) to return all the job handles associated with the user's certificate.

File transfer commands are provided to allow Matlab users to transfer files to and from Grid-enabled compute resources. These commands support the high performance file transfer protocol GridFTP [17]. The GridFTP pro-

tocol defines a number of extensions to the FTP protocol to enable transfer of high volumes of data. Functions are also provided to allow users to create and delete remote files and directories from within the Matlab environment.

4 Job Submission Service

In addition to support for Globus compute resources we provide client functionality to a job submission service [18].

The job submission service offers an XML Web service interface to the Condor high throughput resource management system [19]. A cycle scavenging Condor pool can utilize the computational power of idle UNIX and Windows NT workstations. Through the Web service interface we are able to offer remote access to this compute resource that is both platform and language neutral.

The service interface maps between XML messages representing compute operations and the ClassAds language of Condor. The user must be able to specify compute resources with the capabilities to meet the requirements, i.e., memory, processor, disk capacities, database/archive facilities, and also specialist software environments or licensed applications.

The service interface to Condor provides the solution to this problem combining features of Condor and inspection technology for Web services. The Condor system provides a convenient method for the discovery of resources with special capabilities in the Condor pool. When a computer joins the Condor pool, it declares its capabilities by adding corresponding attributes to its Condor system configuration file. These attributes will then be reflected in the information generated by the resource status query performed by Condor, which is passed on to the Web service.

4.1 Web Service Enhancements

A number of enhancements to standard Web service technologies have been applied to the job submission Web service.

The standard Web services technology collection, which includes XML, SOAP, WSDL and UDDI, has not supplied a solution to the management of service security. To address this problem, the WS-Security specification has been employed in the Condor service implementation. WS-Security [20] is a maturing technology for Web service security management. It extends the simple structure of SOAP to establish a standard security mechanism on the message level, which is independent of the underlying transportation methods. Since WS-Security focuses mainly on the infrastructure, rather than detailed security techniques such as authentication and encryption, it allows established security solutions, including Kerberos [21], PKI, and GSI [14] to be integrated so that they can be applied to Web services in a standard and consistent manner.

In the Condor service, we have deployed PKI based asymmetric encryption in order to ensure message confidentiality and perform user authentication. The exchanges of credential information, i.e., the X.509 certificates that contain public keys, are carried out following the WS-Security definition. In addition, we will also apply XML digital signature [22] to the service messages, as proposed by WS-Security, so as to achieve message integrity.

Another problem with interfacing Condor with Web services is the degraded performance in the data transfer required to transport the input and output files required by the computational job. Traditionally, data transfer with SOAP is based on the Base64 encoding, which imposes heavy costs due to the serialisation/deserialisation of the Base64 strings. We address this problem in the Condor service

by exploiting a recently proposed data transfer format named Direct Internet Message Encapsulation (DIME) [28].

DIME defines a MIME-like message format in which variously typed data that does not fit expediently or efficiently into XML can be directly contained and transmitted along with the standard SOAP messages. Significant improvement to the performance on data transfer can be achieved by using DIME, as overheads for data conversion are avoided. Test results for this have been demonstrated in [23].

These enhancements to the Condor service have been implemented with the help of Microsofts Web Services Enhancements [24], which provides support for several emerging Web service technologies based on the .NET framework. Corresponding to the service enhancements, message filters for the client tools have also been constructed using related Java technologies [25, 26].

4.2 Job Submission Client

The client to the job submission Web service provides a number of functions which are implemented both in the form of a low-level Java class library, and a set of high-level commands for the Matlab environment. These functions enable users to submit, query and retrieve the results of computational jobs.

The client is designed to provide transparency to the underlying computational resources. In this manner the user interface has been separated from the underlying message processor which interacts with different compute resources, of which the Condor job submission service is an example.

The characteristics of the user's job and of the required resources are specified in the Matlab environment according to an API that represents the generic semantics of compute operations. The job specification is then mapped into input arguments to the selected resource by the message processor. The message processor is configured to use a chain of filters, each responsible for processing a specific part of the message, to build input messages appropriate for the resource.

This is illustrated by Figure 1 which depicts the implementation of the file upload functionality when accessing the Condor job submission service. The interaction is primarily based on the SOAP protocol [27], and needs to conform to the security regulation set by the service. For data transmission, the service uses DIME. HTTP is used as the underlying communication protocol. Accordingly, a SOAP output filter, a security handler and a DIME builder are loaded in turn to the output chain. Since the response is expected in plain SOAP format, only a SOAP input filter is loaded to the input chain. And at the end of the chains, an HTTP handler is loaded to handle the actual message exchanges.

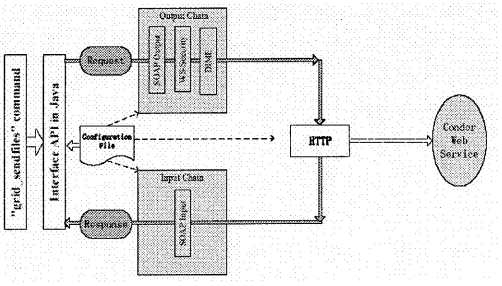

Figure 1: Message filter chains.

5 Conclusions

High-level scripting languages can provide an accessible representation of complex workflows, that are highly flexible, yet allow the rapid development of applications. When providing scriptable components that interfaces with Grid middleware, it is important to be consistent with the syntax of the language, thus

making access to the Grid as transparent as possible.

By adopting middleware technologies that depend upon open standards users are able to plug and play resources as required [18]. Future work in this area includes exploiting the opportunities for collaborative engineering created by emerging Grid technologies, such as the Open Grid Services Architecture [29].

Acknowledgements

This work is supported by the GEODISE e-Science pilot project (UK EPSRC GR/R67705/01). We thank Fluent, Microsoft and Intel for ongoing support.

References

[1] G.E. Pound, M.H. Eres, J.L. Wason, Z. Jiao, A.J. Keane & S.J. Cox. A Grid-Enabled Problem Solving Environment (PSE) for Design Optimisation within Matlab, *Proceedings of the IPDPS 2003*, 50-57, 2003.

[2] E.N. Houstis, E. Gallopoulos, R. Bramley & J.R. Rice. Problem-Solving Environments for Computational Science. *IEEE Computational Science and Engineering*, 4(3): 18-21, 1997.

[3] I. Foster, C. Kesselman & S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organisation. *International Journal of Supercomputer Applications*, 15(3): 200-222, 2001.

[4] J.L. Wason, M. Molinari, Z. Jiao & S.J. Cox. Delivering Data Management for Engineers on the Grid. *EuroPar-2003*, Klagenfurt, 26-29 August, 2003. (*accepted*)

[5] L. Chen, S.J. Cox, C. Goble, A.J. Keane, A. Roberts, N.R. Shadbolt, P. Smart & F. Tao, Engineering Knowledge for Engineering Grid Applications. *Proceedings of the Euroweb 2002*, 12-24, 2002.

[6] G. Fox, M. Pierce, D. Gannon & M. Thomas. Overview of Grid Computing Environments. *Grid Computing Environments - RG*, Global Grid Forum, 2002.

[7] D.W. Baron. *The world of scripting languages*. John Wiley & Sons, Chichester, 2000.

[8] Matlab 6.5. http://www.mathworks.com/

[9] Globus toolkit. http://www.globus.org/

[10] R.J. Allan & D.R.S. Boyd. The UK e-Science Grid: Level 2 Deployment Plan. *The Grid Engineering Task Force*.2002.

[11] Commodity Grid Kits. http://www.globus.org/cog/

[12] G. von Laszewski, I. Foster, J. Gawor & P. Lane. A Java commodity Grid kit. *Concurrency and Computation: Practice and Experience*, 13(8): 643-662, 2001.

[13] K.R. Jackson. PyGlobus: a Python interface to the Globus Toolkit. *Concurrency and Computation: Practice and Experience*, 1 4(13/15): 1075-1083, 2002.

[14] I. Foster, C. Kesselman, G. Tsudik & S. Tuecke. A security architecture for computational grids. *5th ACM Conference on Computers and Communications Security*, San Francisco, California, 1998.

[15] R. Housley, W. Ford, W. Polk, & D. Solo. RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *PKIX IETF Working Group*, 1999.

[16] Resource Specification Language RSL v1.0. http://www.globus.org/gram/rsl_spec1.html

[17] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, & S. Tuecke. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. *IEEE Mass Storage Conference*, 2001.

[18] G. Xue, M. Fairman, G.E. Pound & S.J. Cox. Implementation of a Grid Computation Toolkit for Design Optimisation with Matlab and Condor. *EuroPar 2003*, Klagenfurt, 26-29 August, 2003. (*accepted*)

[19] D. Thain, T. Tannenbaum & M. Livny. Condor and the Grid. in: F. Berman, G. Fox & T. Hey (eds), *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc, 2002.

[20] The WS-Security Specification. http://www.ibm.com/developerworks/library/ws-secure/

[21] Kerberos: The Network Authentication Protocol. http://web.mit.edu/kerberos/www/

[22] The IETF/W3C XML Signature WG. http://www.w3.org/Signature/

[23] G. Xue, G.E. Pound & S.J. Cox. Performing Grid Computation with Enhanced Web Service and Service Invocation Technologies. *Proceedings of the ICCS 2003*, Melbourne, Australia, 2003

[24] Web Services Enhancements 1.0 for Microsoft .NET. http://msdn.microsoft.com/

[25] Java DIME Library v1.0.2. http://onionnetworks.com/dime/javadoc/

[26] Java Cryptography Extension (JCE). http://java.sun.com/products/jce/

[27] Simple Object Access Protocol. http://www.w3.org/2000/xp/Group/

[28] DIME. http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt

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