Implementation of a Grid Computation Toolkit for Design Optimisation with Matlab and Condor

Gang Xue, Matthew J. Fairman, Graeme E. Pound, and Simon J. Cox

Southampton Regional e-Science Centre,
School of Engineering Sciences,
University of Southampton,
Highfield, Southampton, SO17 1BJ, UK
{gx, mjf, gep, sjc}@soton.ac.uk

Abstract. The process of design search and optimisation is characterised by its computationally intensive operations, which produce a problem well suited to Grid computing. Here we present a Grid enabled computation toolkit that provides transparent and stable access to Grid compute resources from Matlab, which offers comprehensive support for the design optimisation processes. In particular, the access and integration of the Condor resource management system has been achieved by using the toolkit components that are enabled by Web service and service enhancement technologies. The use of the computation toolkit for a four-dimensional CFD parameter study with Matlab and Condor is considered as an exemplar problem.

1 Introduction

Engineering design search and optimisation [1] is the process whereby engineering problems are modelled and analysed to yield improved designs. This process involves identifying design parameters that the engineer wishes to optimise, computing a measure of the quality of a particular design (the objective function) using an appropriate model, and using a number of design search algorithms to generate additional information about the behaviour of a model over the parameter space, as well as to optimise the objective function to improve the design quality. It is potentially computationally intensive, as lengthy and repetitive calculations of the objective function with regard to the design variables may be required.

The demand for compute resources by the design optimisation process can be well satisfied by the adoption of Grid computing technologies. Grid computing provides the infrastructure and technologies to enable large-scale resource sharing and the construction of Virtual Organisations (VOs) that address various science and engineering problems [2]. When applied in design optimisation, it allows the process to discover and access resources from heterogeneous environments in a transparent and consistent manner for the required compute tasks. An important aspect of the use of Grid technology for this purpose is to establish links between the design optimisation process and various compute resource management systems, which organise and manage most of the compute resources on the Grid. Notable examples of such systems include Globus [3], Condor [4], Maui [5] and UNICORE [6]. In our previous work [7], access to the Globus system has been provided to Matlab users by

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 357–365, 2003.

building on the Java CoG toolkit [8]. Here we focus on the integration into Matlab of another important target, the Condor system, using an approach more strongly grounded in Web service technologies.

Condor is a resource management system that *creates a High-Throughput Computing (HTC) environment by harnessing the power of UNIX and NT clusters and workstations* [9]. Whilst it can manage dedicated clusters, it can also exploit pre-existing resources under distributed ownership, such as computers sitting on people's desks. For the design optimisation process, Condor provides a robust computational environment, which is able to find resources with specific requirements, carry out the compute operations, and manage the computation jobs for the process. The HTC style of Condor also suits the need of design optimisation, which can yield better results with further calculations.

The integration of Grid resource management systems with the design optimisation process is mainly based within the Matlab environment [10], which is selected as the interface to the Geodise PSE [7]. The Matlab package provides a fourth generation language for numerical computation, built-in math and graphics functions and numerous specialised toolboxes for advanced mathematics, signal processing and control design. It is widely used in academia and industry for algorithm prototyping, and for data visualisation and analysis. From version 6.5 Matlab also contains a number of Just-In-Time (JIT) acceleration technologies to improve the performance of native Matlab code.

In order to facilitate access to compute resources on the Grid, especially the Condor system, for users in Matlab and potentially other problem solving environments, a computation toolkit has been developed in the form of a set of client utilities and replaceable middleware components. The toolkit adopts a layered, highly flexible and easily configurable structure, which allows using a consistent user interface to access different compute resource systems with the help of corresponding middleware components. In particular, a middleware component based on Web service technology has been constructed to expose Condor as a Grid service, and hence make it accessible to the toolkit users.

In the rest of the paper, we first focus on the design and implementation of the computation toolkit, including the Web service interface to the Condor system. We then demonstrate the application of the toolkit in the Matlab environment for engineering design optimisation problems.

2 The Computation Toolkit

The computation toolkit provides users with a set of basic functions to perform Grid based computation tasks, which are described below together with the technologies behind and the design of the toolkit.

2.1 Functions for Grid Based Computation

In our computation toolkit, a number of functions are implemented both in the form of a low-level Java [11] class library, and a set of high-level commands for the Matlab environment, which enable users to query the resources, submit and manage the

compute jobs, perform data transmission, and configure security settings. The Java class library provides mappings between the APIs and the compute functions. It is therefore possible to perform a Grid based compute operation programmatically. The commands for Matlab users listed in Table 1 are written in the interpretive Matlab language, so that the compute functions are presented in a manner consistent with the behaviour and syntax of the Matlab environment. The commands can be used to start compute operations based on interactive user inputs, or directly on script files.

The `grid_createcredential` command needs to be called before any other commands in the toolkit can be used. It sets up the user credentials according to the security requirement policy of the target compute resource. The user credential can be as simple as a username/password pair, or as complicated as a PKI (Public Key Infrastructure) [12] based user certificate. When all compute operations are finished, the `grid_destroycredential` command is used to disable the user credential, so that it won't be misused or stolen.

The `grid_jobreqest`, `grid_sendfiles` and `grid_startjob` commands represent the major steps in starting a compute job submission process. Users start the compute tasks with requests to the target resources for job submission. Once accepted, the job files are delivered to the resources using the data transfer command. Users then send out instructions to start the computation. In addition, a few job management commands are also provided to enable users to monitor and manage the running of jobs. Once the jobs are finished, job results can be easily retrieved using the `grid_retrievefiles` command.

Table 1. Compute Commands

Function Name	Description
grid_createcredential	Loads the user credential to the toolkit for compute operations
grid_destroycredential	Destroys the user credential stored by the toolkit
grid_jobrequest	Requests the submisson of a compute job with specification for the compute resources and description of the job. If accepted, a job handle is returned.
grid_startjob	Starts the submitted job identified by a job handle.
grid_killjob	Terminates the job identified by the job handle.
grid_getstatus	Queries the status of submitted jobs identified by the job handles.
grid_listjobs	Returns job handles for all jobs submitted to the target resource that belong to the user.
grid_sendfiles	Uploads the job files required for job execution to the target resource.
grid_retrievefiles	Retrieves all result data/files generated by the finished job identified by a job handler.
grid_jobsubmit	Automates the process of job request, job files transfer, and job start.

2.2 Design of the Computation Toolkit

One of the most important features of Grid technologies is to provide transparency in accessing computational resources spread across the Grid. It is important because of the heterogeneity of the Grid environment. For instance, compute resource systems that can be targeted by the design optimisation processes include Globus, Condor, and

UNICORE. Each of them has a different user interface, incompatible security policy, and diverse workflow model. Traditionally, separate client packages would be needed to access these different resources, which is clearly unscalable and not ideal for system integration. Moreover, these tightly bound client packages are less adaptable to potential changes made by the target systems, and would force frequent modification to design optimisation applications.

To provide the essential transparency, the computation toolkit has been implemented in a distinctive structure, which separates the user interface from the underlying message processor that is responsible for interactions with different remote resources based on various protocols. The user interface is an API that represents the basic semantics of compute operations and is designed to be simple but stable. In contrast, the message processor is much more dynamic and powerful. It is implemented as two chains of filters for input and output message processing. Each filter is responsible for processing a specific message part, or even the entire message based on the protocol it is responsible for. At the end of the chains, there is a communication handler, which is responsible for direct interactions with the compute resource or compute service middleware. By using Java reflection, the filter chains are dynamically constructed at the runtime based on the information loaded from a configuration file, which is detached from the client applications and can be easily modified. It is therefore possible to access different resource systems or adapt to changes by loading different set of filters.

Figure 1 shows the implementation of the file upload functionality when accessing the Condor service. The interaction is primarily based on the SOAP protocol [13], and needs to conform to the security regulation set by the service. For data transmission, the service uses DIME [14]. 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.

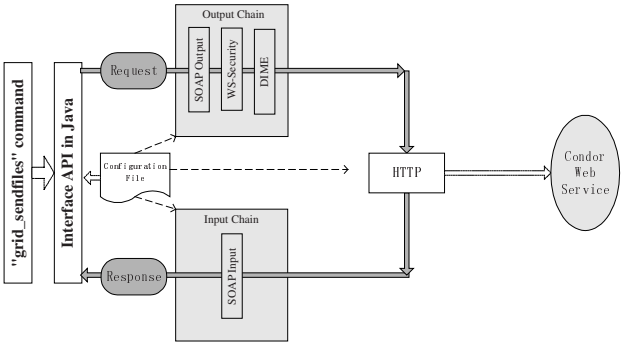


Fig. 1. Implementation of Data Upload with the Message Filter Chains

The flexibility brought by the structure of the computation toolkit allows different compute resource systems accessible in a client-server mode to be integrated into the toolkit. Apart from the middleware component for the Condor system provided in the

current implementation, we will also construct message filters based on our previous work on Globus client tools, so that the Globus system can also be accessed using the toolkit.

3 The Web Service Enabled Interface to Condor

The computation toolkit consists of a set of Web service based middleware components that expose the Condor system. Several enhancements have been applied in addition to the standard Web service technologies to provide features required for Grid based operations. We have also exploited features of Condor and Web service inspection technology to support compute tasks with specific requirements on the resources.

3.1 The Web Service Interface

The Web service enabled interface to the Condor system has been constructed in order to achieve programmatic, platform and language neutral access to the resources managed by Condor. It provides mappings between the computation toolkit functions and the resource management mechanisms of Condor. The interface is primarily an interpreter between XML messages representing the compute operations and the ClassAd language of Condor. It also hides from the users details of resource management that are proprietary to Condor, so as to achieve the desired transparency.

In addition to interactions with Condor, the service also implements general Grid service functions that are not provided by the Condor system, such as management of security and process status. Furthermore, the service has also adopted several enhancements from future Web service technologies, which are described in detail in the following section.

3.2 Enhancements to the Service for Compute Operations

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 and make the service compatible with common Grid security management practise such as GSI, we have employed the WS-Security specification in the Condor service implementation. WS-Security [15] 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 [16], Public Key Infrastructure (PKI), and GSI [17] 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 keep 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 [18] to the service messages, as proposed by WS-Security, so as to achieve message integrity.

Another problem about interfacing Condor with Web service is the degraded performance in data transfer. 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). 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 [19].

The enhancements to the Condor service have been implemented with the help of Microsoft's recently released WSE1.0 [20], which provides sound 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 [21] [22].

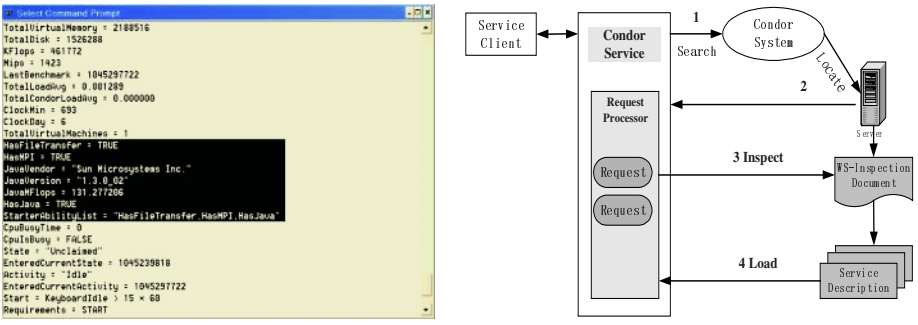
3.3 Discoveries and Access of Resources with Special Capabilities

We define resources to encompass compute facilities with certain features, e.g. memory, processor, disk capacities, database/archive facilities, and also specialist software environment or licensed applications. Design optimisation requires access to all of these resource types at various stages of the process. It is therefore necessary for the users to be able to discover all resources on demand, and manage access to these resources in a consistent way to familiar requests for particular compute capabilities.

Our service interface to Condor provides the solution to this problem by 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. The service will therefore be able to make the discovery by identifying the target attributes. Figure 2(a) shows the Condor status query results for a machine that supports Java and MPI.

Additional regulations that need to be enforced for computing on special resources can be presented in XML Schema and WSDL file that extends the original ones of the Condor service. Once the service makes the discovery, it follows the convention defined by the WS-Inspection specification [23] to locate the WS-I documents on the resources, which point to the extended schema and WSDL files. The service will then use the extended service definition to process the incoming requests. The entire process is illustrated in Figure 2(b).

This model of resource discovery and inspection can be extended to suit the situation where resources are presented to the Web service as Web service themselves. In that case the sub-services are registered directly with the interface service, which becomes a service broker in addition to its original functions.



(a) (b)

Fig. 2. (a) Discovering Resources of Special Capabilities in Condor. (b) Process of Special Capability Discovery through the Service

4 The Computation Toolkit Application Exemplar

To demonstrate the possible use of our Grid-enabled computation toolkit in Matlab, we choose a simple problem of fluid dynamics, which is a parameter study of the fluid dynamics of a 2D parameterised geometry of the nacelle for an aircraft engine. A suitable objective function was calculated across four design variables which describe curves on the upper and lower surface of the nacelle. In this example, each simulation is set up as a compute task, which is submitted to a Condor managed resource from Matlab using the toolkit command, as shown in Figure 3(a).

Figure 3(b) shows the results of 648 simulations that were performed. The jobs were distributed by the Condor system over a cluster of 12 NT nodes. A parameter study such as this allows the engineer to easily evaluate the impact of different design variables upon the quality of a design.

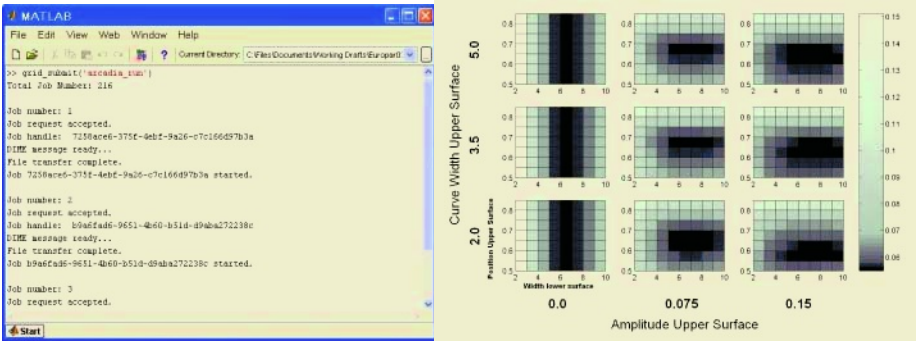


Fig. 3. (a) Starting Grid Computation Tasks Using the Computation Toolkit. (b) Visualised Result of a Four Dimensional CFD Parameter Studies

Once an engineer has developed a Matlab script based on our toolkit for design studies like this, improvements to the robustness and reliability of the underlying Condor service, or additional Condor-enabled resources and capabilities can be exploited in a seamless and transparent way without further modification to the script.

5 Conclusion and Future Work

The significant demand for computation resources in the design optimisation process can be satisfied in a transparent way using Grid computing. The computation toolkit presented in this paper facilitates access to computation resources, in particular the Condor system, from an engineering PSE such as Matlab. The implementation of the toolkit is based on a highly flexible structure and a few future Web service technologies. As an exemplar, we have demonstrated the use of the toolkit for a simple CFD design optimisation problem. Future work on the toolkit will mainly focus on the implementation of filter and middleware components for the access of different computation resources.

Acknowledgements. This work is supported by the Geodise e-Science pilot project (UK EPSRC GR/ R67705/01). We thank the authors of the Condor system at the University of Wisconsin, Madison.

References

[1] The Geodise Project. http://www.geodose.org
[2] I. Foster, C. Kessleman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organisations, International Journal of Supercomputer Applications, 15(3):200-222, 2001
[3] The Globus Project. http://www.globus.org
[4] The Condor Project. http://www.cs.wisc.edu/condor/
[5] The Maui Scheduler. http://supercluster.org/maui/
[6] UNiform Interface to COmputing Resources. http://www.unicore.de/
[7] G. E. Pound, M. H. Eres, et al. A Grid-Enabled Problem Solving Environment (PSE) for Design Optimisation within Matlab. Proceeding of IPDPS 2003, Nice, France.
[8] Commodity Grid Kits. http://www.globus.org/cog/
[9] Miron Livny, and the Condor Team. Condor User Manual. http://www.cs.wisc.edu/condor
[10] Matlab 6.5. http://www.mathworks.com
[11] Java 2. Sun Microsystems Inc., http://java.sun.com
[12] Public Key Infrastructure. http://www.ietf.org/html.charters/pkix-charter.html
[13] Simple Object Access Protocol. http://www.w3.org/2000/xp/Group/
[14] DIME. http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt
[15] The WS-Security Specification. http://www.ibm.com/developerworks/library/ws-secure/
[16] Kerberos: The Network Authentication Protocol. http://web.mit.edu/kerberos/www/
[17] Grid Security Infrastructure. http://www.globus.org/security/
[18] The IETF/W3C XML Signature Work Group. http://www.w3.org/Signature/

[19] G. Xue, G. E. Pound, S. J. Cox. Performing Grid Computation with Enhanced Web Service and Service Invocation Technologies. Proceedings of ICCS 2003, Melbourne, Australia.
[20] Web Services Enhancements 1.0 for Microsoft .NET. http://msdn.microsoft.com/
[21] Java DIME Library v1.0.2. http://onionnetworks.com/dime/javadoc/
[22] Java Cryptography Extension (JCE). http://java.sun.com/products/jce/
[23] Web Service Inspection Language 1.0. http://www.ibm.com/developerworks/Webservices /library/ws-wsilspec.html