

Databases, Workflows and the Grid in a Service Oriented Environment

Zhuoan Jiao, Jasmin Wason, Wenbin Song, Fenglian Xu, Hakki Eres,
Andy J Keane and Simon J Cox

School of Engineering Sciences, University of Southampton, UK
{z.jiao, j.l.wason, w.song, f.xu, hakki.eres, ajk, sjc}@soton.ac.uk

Abstract. As the Grid moves towards adopting a service-oriented architecture built on Web services, coupling between processes will rely on secure, reliable, and transacted messages and be supported by databases. We have built a generic toolkit targeted at design engineers, which provides convenient methods to access a grid-enabled repository. In this paper we report how we have developed it further, and integrated it into a workflow toolkit to support a range of activities that design engineers have previously attempted to perform by multiple ad-hoc methods in the workflows used to improve designs. It also presents opportunities for improving the process of design search in a variety of ways that would have been otherwise hard to implement. We show the potential of our grid-enabled data repository in the context of workflow management, engineering optimisation process monitoring and steering.

1 Introduction

Engineering design search and optimisation (EDSO) is a computationally and data intensive process. Its aim is to achieve improved designs by exploiting engineering modelling and analysis. The quality of a particular design is measured by the value of an objective function. The design search process systematically modifies the variables which describe the design to increase, or reduce, this quality measure, whilst ensuring that the design variables satisfy various constraints. At the heart of this process lie workflows which, through a series of computational experiments, can build a repository containing samples of the design variables with their corresponding objective function values. In previous work [1] we focused on making grid-enabled databases more accessible to engineering designers so that they could be used routinely as a repository. Files and data structures are associated with metadata and services are provided to archive, locate (by querying the metadata), and retrieve them.

In this paper, we now demonstrate that this repository may in fact be exploited in a much wider variety of ways throughout the whole process that engineers follow to locate improved designs efficiently. We give some additional background in section 2 and describe our repository architecture and workflow construction environment in section 3. We then use case studies drawn from engineering design practice to illustrate, in sections 4-6, scenarios in which the database is used to store, monitor, and steer optimisation workflows. Section 7 summarizes our current and future work.

2 Background

The aspirations of the Grid to allow resources to be shared seamlessly and securely address many of the challenges encountered in engineering design search, which requires coupling of computing power, data resources and software applications. Many commercial engineering design search packages exist, for example iSIGHT [2] and ModelCenter [3]. However, they suffer from a limitation that data is managed internally in proprietary formats making it difficult to share with processes running outside of the package environments. Another common practice is to use the file-system as a medium for transferring data between various processes, which works fine in a local environment, but falls short in the Grid environment where processes can be dispatched to run on different machines with varied hardware and software configurations: in such an environment secure, reliable, transacted messages coupled to databases offer a better way to provide resilience and quality of service.

Our approach is to bring Grid technologies into an environment and mode of working familiar to engineers. In particular the routine use of a database-driven repository allows data to be passed from process to process on the Grid, and retrieved and exploited by other tools or packages. To achieve this we have developed the Geodise [4] database toolkit [1] which provides a set of Web services and a Java client API to allow higher level applications to manage data on the Grid. The toolkit is used by engineers at the University of Southampton working on a variety of design optimisation problems and by the GENIE [5] climate modelling project. In [6] we described in detail how the toolkit was used successfully in the Matlab [7] environment together with the Geodise Computational Toolkit. Matlab is a scripting language popular in the engineering community for its ease of use and rich functionality. We provide a number of Matlab functions to access the repository through our API, which users can incorporate into their scripts to manage data. It also provides a powerful execution engine for our workflows, and is available on a wide variety of platforms and operating systems. Other environments, including existing design packages, could be equally supported by writing routines in the appropriate language to access the API.

Whilst an expert can simply use their favourite editor to write a script, incorporating calls to our grid toolkits, and then run it from a Matlab command prompt, we also provide a workflow construction tool to assist users with the whole sequence of tasks required to develop and execute their design search. In this paper, we will focus on demonstrating how the repository is exploited within our workflow tool environment throughout each step in the design process, particularly application monitoring.

3 Architecture

The Geodise Database Toolkit is designed to help engineers manage the large amounts of data produced by their applications. We provide the means to store and share files and variables (primitive values and data structures) in a Grid-enabled repository. Each file or variable has some standard metadata associated with it (e.g. archive date, file size), which may be supplemented with additional user-defined custom metadata. Related data, such as that referring to a whole design job, may be

logically grouped together as a datagroup and metadata can be added so that the entire collection can be described. Data may belong to multiple datagroups, which may also contain sub-datagroups, so users can describe and exploit relationships or hierarchies.

A data file is archived to a Globus [8] server using GridFTP [9] and upon archiving it is assigned a Universally Unique Identifier (UUID) which allows it to be physically located and retrieved later. Its associated metadata is stored in a relational database (Oracle 9i [10]): standard metadata is stored in tables for efficient access, whilst custom metadata is stored as native XML for flexibility and this can be transparently generated from user-defined Matlab structures with our XML Toolbox for Matlab [11]. Variables are stored in a similar way and the database is queried using a simple syntax to locate files/variables based on their characteristics. Scalability of the system is dependent on the underlying database software, particularly the efficiency of querying large quantities of unstructured XML. The concept and benefits of using a relational database containing metadata on top of a transacted file system is also at the heart of the WinFS file-system in Longhorn, the next generation of Windows [12]. In other work we demonstrate how this metadata becomes even more powerful in knowledge reuse when described in a semantically consistent way to allow key concepts and relationships to be defined and shared using ontological language [13].

All database access is supplied through secure Web services for storage and query of metadata, file location and authorisation, which can be invoked with platform independent client code anywhere on the Grid. Web service communication is secure, allowing certificate based authentication and authorisation using the same delegated credentials that provide a point of single sign-on for computational jobs on the Grid [14]. Secure requests are signed with the user's proxy certificate which the Web service verifies and uses to authorise database access, record data ownership and allow users to grant access permissions to others. The architecture described permits data archiving, querying and retrieval from any location with web access, Java support, the required APIs and a valid proxy certificate. In addition to its use from scripts written in Matlab, the database toolkit has also been integrated with the Geodise workflow construction environment (WCE). The full WCE architecture is described in [15].

Figure 1 shows the main roles of the WCE in assisting engineers to build and reuse workflows, verify resources, execute workflow scripts, and monitor or control the workflow as it runs locally or on the Grid. A user's first task in the WCE is to load-up or visually construct a series of connected tasks (a workflow), shown in Figure 2. The component view on the left pane of the WCE GUI contains a hierarchy of tasks and related computational components (e.g. Matlab functions). The hierarchy is created from an XML document describing the available named components, their inputs and outputs. A component can be dragged onto the main workflow view and may be linked to other components. The default initial values of function inputs and outputs can be altered in the component property window. Dataflow is configured by making associations between the output and input parameters of different components. The complete workflow description is saved as XML for reuse in the graphical editor, but more importantly as a Matlab script (e.g. `optim.m` in Figure 1) which can be stored in the Geodise repository, reused and edited outside the WCE without engineers learning new workflow formats, and run on compute resources with Matlab support. The available compute resources are pre-described in a user-editable configuration file and displayed by the WCE, which is able to verify the resources are accessible to the user

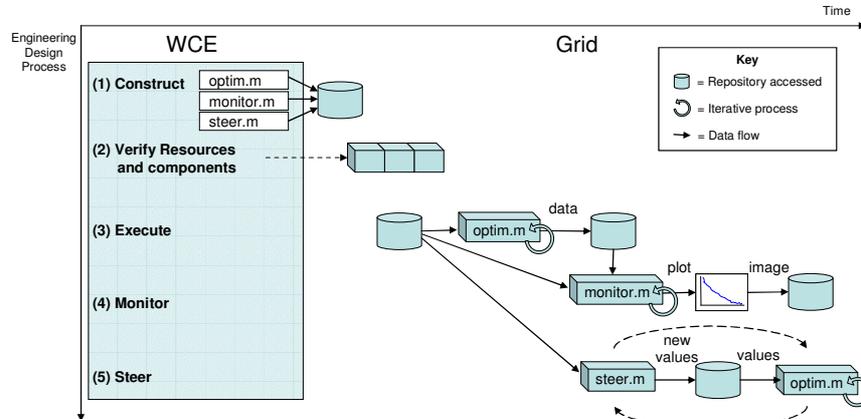


Fig. 1. - Roles of Workflow Construction Environment (WCE) in workflow construction, resource verification, execution, monitoring and steering.

and configured properly before execution. The WCE also assists in constructing scripts to monitor and steer workflow execution (e.g. `monitor.m`) which may be stored in the repository for later use. We now give examples of how workflows can be managed, monitored and steered.

4 Storing Workflows

An example of constructing a workflow for a two-dimensional airfoil design optimisation problem is shown in Figure 2.

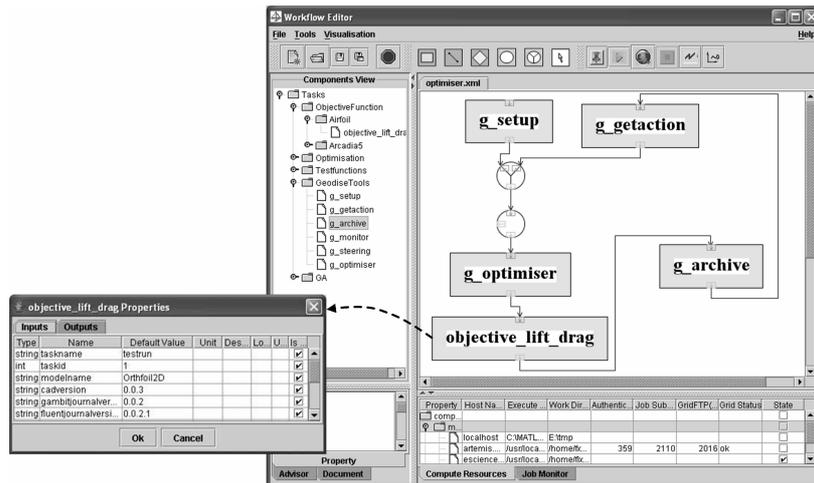


Fig. 2. - The workflow construction environment allows users to select and configure workflow components, run workflows locally or on the Grid and monitor their progress.

The goal of the optimisation is to minimize the drag/lift ratio by varying the weights of six basis functions and the thickness-to-chord ratio [16]. After construction a user may save the workflow locally, or archive it to the repository for sharing and reuse. The WCE calls the database API to store and group the workflow script, its XML description, and some additional metadata which can be later queried to help locate the workflow efficiently. By using datagroups it is also possible to link the script to any data generated during its execution.

5 Monitoring

Optimisations based on high fidelity analysis are typically time consuming, therefore it may be desirable to monitor the progress of the search as it is running and make changes if necessary. Most integrated packages [2], [3] implement monitoring on the local machine controlling the optimisation. However, it is difficult, if not impossible, at present to monitor the process from a separate machine. By exploiting the central data repository it is possible to retrieve and post-process monitoring data independently and asynchronously from the running script. Furthermore, the post-processed information (e.g. a plot or image) may itself be returned to the database, from where it can be accessed and retrieved from any location, or even sent to a mobile device.

Whilst technologies exist that allow monitoring of Grids for scheduling, performance analysis and fault detection, e.g. the NetLogger Toolkit [17] and Gridscape [18], we focus on monitoring the progress of the user's application. This progress is described by the ongoing deposition of variables and data into the repository as transactions within and between grid processes occur. The user can supply a script (`monitor.m` in Figure 1) which periodically performs a query to retrieve the data, render it, and then return the image to a predefined location, e.g. the database. Data deposited for monitoring purposes are aggregated together into a datagroup assigned with an identifier (a UUID) and additional metadata (e.g. job index, job name) to help the monitor script, or a user operating away from the WCE, query and retrieve the data.

Along with bespoke visualisations a user might develop for their data, in design optimisation, a number of standard plots are used to monitor the progress of the search, such as viewing the objective function at the sample design points explored, or its convergence. These predefined views need only be passed (by value or reference) the data they require. In the pass-by-reference model data is organized in a prescribed way when it is archived and is obtained by passing a datagroup ID to query. Pass-by-value places no restriction on the data organization but requires the user to provide a function that takes a datagroup ID and performs the necessary query and mapping to return the values correctly to the plotting routines. The WCE can assist the user in linking their data to the monitoring script and specifying polling frequency, and displays the latest plot, which may be retrieved from the database, in a monitoring panel.

Example: the results from a design of experiment analysis can be used to build a response surface model (RSM), a meta-model which interpolates from the design points where the objective function was calculated to those where it was not [16]. Searching this meta-model for potentially better designs is then rapid, and further expensive simulations need only be performed to validate areas which the meta-model

identifies as promising. Whilst this cycle of building, searching and tuning the RSM is automated as far as possible, monitoring the evolution of the response surface as the design search progresses can sometimes reveal additional ways in which the search can be accelerated. For example, the sampling process for the initial build can simply be terminated if the important features of the response surface remain the same as more design points are added. Figure 3 shows a response surface model built from results with 16 runs of the airfoil problem with two design variables. The model is updated as more data becomes available in the database or at user specified intervals.

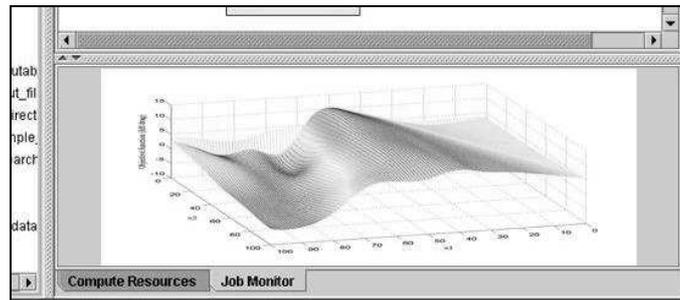


Fig. 3. – Response Surface Modelling monitoring view screen shot.

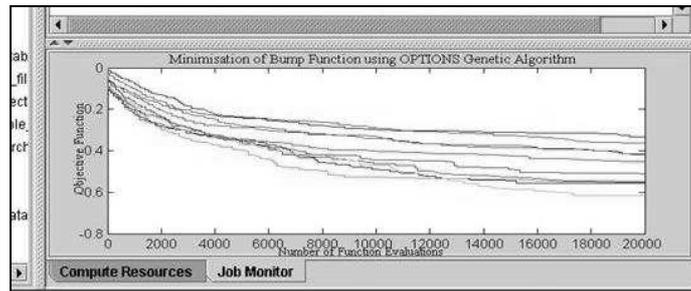


Fig. 4. - Improvement in objective function as more design points are searched using a genetic algorithm optimiser. Multiple curves are shown for searches using different start points.

Designers may wish to view detailed information about each design point, such as geometry or analysis results. These might reveal unforeseen problems with the analysis procedure or geometry generation in certain regions of the design space, which could yield misleading results from the design search and significant wasted time and resource if only discovered at the end. As the picture of the design space emerges, a skilled designer may be able to use physical insight to explain features or patterns in the good (or bad) designs to derive a more efficient formulation of their problem.

Another unknown at the start of the design process is how long it is worth continuing to search. If analyzing further designs is unlikely to yield any further improvement then the job can be terminated. If the converse is true it may be desirable to extend the search. Figure 4 shows how the minimum value of the objective function improves as more design points are analysed during a genetic algorithm optimisation.

6 Steering

We have described how the repository provides a convenient location to deposit and then monitor variables as the search progresses. Steering is achieved by updating values in the database which are retrieved by the running script to control the design search. A number of uses for steering have been identified [19], e.g. to modify the design variables when the objective functions show no further improvements early in the search. In practical implementation from our scripts, two important issues arise: 1) Side-effects from arbitrary variable updates should be carefully understood and controlled; 2) Information in our database is ‘write-once’, so data is not modified or overwritten, instead scripts query to locate the latest version of the variable. Thus as variables are updated, a complete log of all intermediate values is recorded, which is important for provenance or repeatability.

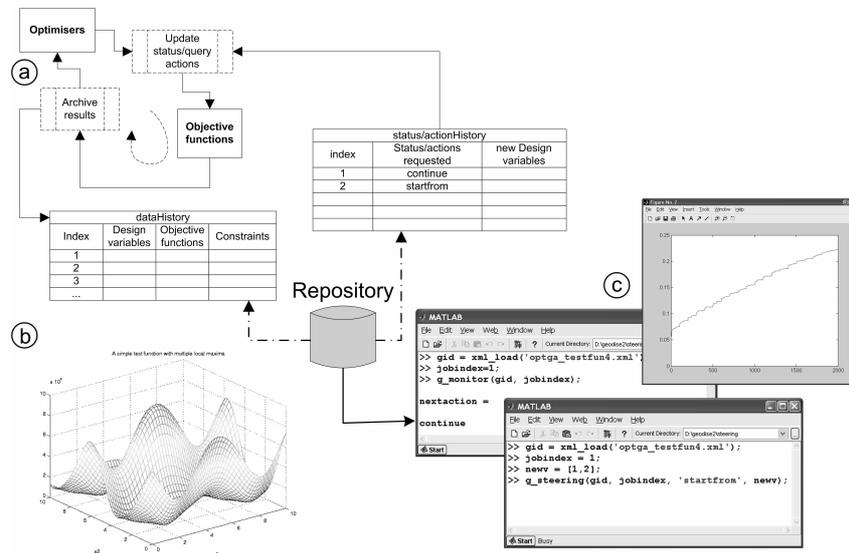


Fig. 5. - Illustration of optimisation monitoring and steering using a grid-enabled database.

Traditional optimisation workflows are augmented with a set of grid-enabled query and archive functionalities that users can plug into their existing scripts at various points (Figure 5a). In each iteration, the optimisation process retrieves the current user action request from the action history in the database, and takes the appropriate action to achieve steering. To demonstrate how the system works we provide a simple example using a typical test function with multiple local optima. During optimisation, a user can monitor progress and interactively steer the process. The landscape of the test objective function is shown in Figure 5b. The search progress is monitored from another Matlab session based on the datagroup ID which provides the entry point to the database. The search history is shown in Figure 5c along with the effect of the user’s steering activity.

7 Conclusions and Future Work

In this paper we have shown a number of roles for our repository throughout the process of EDSO. The toolkit enables engineers to write scripts in a familiar language and archive them along with data produced at various stages of the design optimisation process to a grid-enabled database. This makes the data available outside the optimisation package boundary, and accessible from various locations by authorised users. This in turn allows monitoring and steering of the optimisation, and thus enables new ways to accelerate, improve and reduce the cost of the design process.

An important issue for future consideration is provision of a redundant failover system for the metadata services and database. Future work will also include investigating data lifetime management and cleanup strategies, and integrating Geodise semantic and knowledge tools into the WCE.

References

1. Wason, J.L., Molinari, M., Jiao, Z., Cox, S.J.: Delivering Data Management for Engineers on the Grid. Proc. Euro-Par 2003, LNCS (2003) 413-416
2. iSIGHT, <http://www.engineous.com/index.htm> (2004)
3. ModelCenter, <http://www.phoenix-int.com> (2004)
4. Geodise Project, <http://www.geodise.org>
5. GENIE (Grid ENabled Integrated Earth system model) project, <http://www.genie.ac.uk>
6. Eres, M.H., Pound, G.E., Jiao, Z., Wason, J.L., Xu, F., Keane, A.J., Cox, S.J.: Implementation and utilisation of a Grid-enabled Problem Solving Environment in Matlab. Future Generation Computer Systems (in press).
7. Matlab 6.5. <http://www.mathworks.com>
8. The Globus Toolkit 2.4, <http://www.globus.org>
9. GridFTP, <http://www.globus.org/datagrid/gridftp.html>
10. Oracle 9i Database, <http://otn.oracle.com/products/oracle9i>
11. Molinari, M.: XML Toolbox for Matlab, GEM/Geodise (2004) http://www.soton.ac.uk/~gridem/xml_toolbox
12. Longhorn Developer Center, <http://msdn.microsoft.com/longhorn>
13. Chen, L., Shadbolt, N.R., Goble, C., Tao, F., Cox, S.J., Puleston C.: Managing Semantic Metadata for the Semantic Grid. Submitted to ISWC2004, Japan (2004)
14. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A Security Architecture for Computational Grids. Proc. 5th ACM Conference on Computer and Communications Security Conference (1998) 83-92
15. Xu, F., Cox, S.J.: Workflow Tool for Engineers in a Grid-Enabled Matlab Environment. Proc. UK e-Science All Hands Meeting (2003) 212-215
16. Song, W., Keane, A.J., Eres, M.H., Pound, G.E., Cox, S.J.: Two Dimensional Airfoil Optimisation using CFD in a Grid Computing Environment. Proc. Euro-Par 2003, LNCS (2003) 525-532
17. Gunter, D., Tierney, B., Jackson, K., Lee, J., Stoufer, M.: Dynamic Monitoring of High-Performance Distributed Applications, Proc. IEEE HPDC-11 (2002) 163-170
18. Gibbins, H., Buyya, R.: Gridscape: A Tool for the Creation of Interactive and Dynamic Grid Testbed Web Portals. Proc. 5th IWDC International Workshop (2003) 131-142
19. Shahroudi, K. E.: Design by Continuous Collaboration Between Manual and Automatic Optimization, Technical Report (1997) <http://ftp.cwi.nl/CWIreports/SEN/SEN-R9701.pdf>