Integrating Data Management into Engineering Applications

Zhuoan Jiao, Jasmin Wason, Marc Molinari, Steven Johnston & Simon Cox
School of Engineering Sciences, University of Southampton, UK
{z.jiao, j.l.wason, m.molinari, s.j.johnston, sjc}@soton.ac.uk

Abstract. Engineering design search and optimisation is the process whereby engineering modelling and analysis are exploited to yield improved designs. It is a computationally and data intensive process - an application domain well-suited to Grid technology. Many on-going activities focus on utilising computing resources on the grid. However, it is equally important to manage efficiently the vast and varied amount of data produced by grid applications. In this paper we describe the tools we have implemented in the Geodise project to transparently integrate database technologies into engineering applications. The toolkit has been incorporated into Matlab which is popular in the engineering community for its ease of use and rich functionality. We adopt open standards and a service oriented approach to leverage existing database technologies. Databases are accessed through web services and Globus is used for file transfer and authentication. The Geodise data management architecture consists of flexible, modular components which can be utilized by higher level applications for managing data on the Grid. We demonstrate our toolkit in the context of an industrially relevant exemplar problem.

Keywords: Database applications, Grid Computing, XML, XML Schema, Web services, Matlab, engineering design search and optimisation.

1. Introduction

Engineering design search and optimisation (EDSO) is an application domain well suited to Grid technology, involving computationally and data intensive procedures to obtain improved engineering designs. Optimisation tools modify a design to increase, or reduce, some measure of merit (called the objective function) whilst satisfying various constraints. The objective function measures the quality of a particular design and is computed by an optimising algorithm which adjusts each of a selected set of design variables to determine how they affect the performance. This is coupled with an appropriate engineering analysis code, such as Computational Electromagnetics (CEM) or Computational Fluid Dynamics (CFD) code, to analyse the properties of a design, and seek a solution that optimises the objective.

The EDSO process often involves a number of files and data parameters in addition to lengthy and repetitive calculations requiring access to significant computational resources. This makes the problem domain of EDSO using CEM/CFD well-suited to the applications of Grid technology [1] which allows the sharing of computing power, data resources and software applications over the Internet. By providing scalable, secure, high-performance mechanisms for discovering and accessing remote resources, Grid technology makes scientific collaborations within a Virtual Organisation (VO) achievable in ways that were previously impossible. In Geodise [2] our goal is to develop sophisticated but easy-to-use toolkits to help engineers carry out their daily tasks efficiently by making use of the compute and data resources available on the Grid.

While much ongoing Grid research is focused on the compute aspect of the Grid, it is equally important to manage efficiently the vast amount of data created by Grid applications, such as EDSO. Traditionally, data in many scientific and engineering disciplines has been organized in application-specific file structures, and a great deal of data accessed within current Grid environments still exists in this form [3]. When there are a large number of files it becomes difficult to find, compare and share the data. We solve this problem by allowing additional information (metadata) describing the nature of the files to be defined and stored in databases, so that files can be located easily by querying the metadata.

The OGSA [1] Data Access and Integration (DAI) project [3] is tackling issues regarding to database integration with the Grid in general. Our focus is on providing database services in an environment that is familiar to engineers, to help them manage the large amount of data created by their grid applications. We have chosen Matlab [4] as such an environment, as it is popular with engineers for its ease of use and rich functionality. We adopt open standards and a service oriented approach [5] to develop a set of tools to extend the Matlab functionalities so that engineers can make use of grid computing and data management services easily. This set of tools consists of the Geodise computational toolkit [6], XML toolbox [7] and a database toolkit which is the focus of this paper. These toolkits can be adapted to work in other scripting environments, such as Jython [8].

As shown in Figure 1, the building blocks of the engineering problem domain, as well as the data management functionality provided by Geodise, are wrapped as Web/Grid services where appropriate, making use of Java and Grid technologies. A user accesses this functionality via Geodise provided functions which in turn call a client side Java API to communicate with the services. The knowledge service, as described in [9] provides intelligent support to the users through an ontology service and dynamic advice based on data stored in a database.

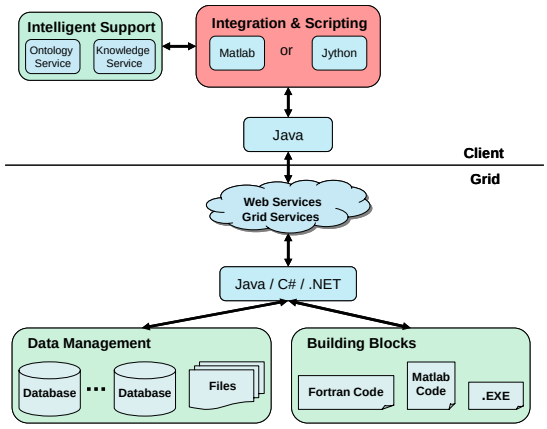

Figure 1 The Geodise architecture, where a scripting environment (e.g. Matlab) can be used to integrate grid-based compute, data and knowledge resources for engineering applications.

The remainder of this paper is organized as follows: section 2 describes the architecture of the Geodise database toolkit; section 3 explains the various service components and how they can be incorporated into the Matlab environment; an example is given in section 4, and section 5 presents conclusions and future work.

2. Architecture

A major aim of the Geodise data management architecture is to provide service components that can be utilised by higher level applications for managing data on the Grid. Another objective is to provide a simple, transparent way for engineering users in a VO to archive files along with additional metadata, without needing to know the underlying storage mechanism. Although files can be archived and retrieved based on unique identifiers, storing additional metadata in a database makes it possible to locate a file based on its characteristics (e.g. the values of variables it contains). An optimisation may take a long time to run and it is desirable to store important data for later re-use. It should also be possible for others to reuse the data, and, to encourage data sharing, users need a way to specify who else can discover and retrieve their data.

To support these goals, we have implemented the Geodise database toolkit. It extends an engineering environment by using open standard technologies as shown in Figure 2. Files are stored in file systems while various types of technical and application specific metadata about files, their locations and access rights are stored in databases.

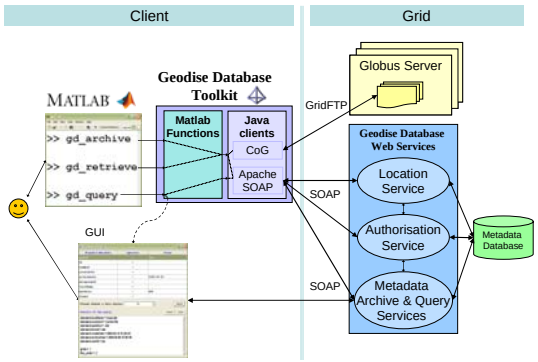

Figure 2 A high level set of functions can be written on top of a client side Java API to provide a straightforward scripting interface to data management Web service functionality and secure file transfer. A user script may run locally as shown, or on a remote compute resource with the user's delegated authorisation credentials.

To enable data sharing by users and applications at different locations in a VO, files need to be transported efficiently between sites. We use the platform independent Java CoG kit [10] to utilize the Grid Security Infrastructure (GSI) [11] for authentication and security, and GridFTP [12] for secure file transfer between user's local machine and the file stores. Files are archived on the file storage servers using generated UUIDs (Universally Unique IDentifiers) [13] as handles.

Access to databases is provided through Web services [5], invoked using the Simple Object Access Protocol (SOAP) [14] which uses a combination of XML and HTTP to transfer data between the services regardless of their underlying programming language or platform. Client jobs running remotely on the Grid can also access these services to retrieve input files from and archive results to a repository. This provides a central location for applications running on the Grid to exchange data and allows the user to query and retrieve job results when convenient.

In next section, we describe the various services provided by the Geodise database toolkit in detail, and show how they have been incorporated into the Matlab environment.

3. Service Components

3.1. Database Services

The Geodise database toolkit consists of service components upon which higher level applications can be built, as shown in Figure 2. The file location service keeps a record of file handles and locations, in terms of host and directory, in a database so that a handle is all that is required to retrieve a file. The metadata archive service complements file archiving by allowing the storage of additional descriptive information detailing a combination of technical characteristics (e.g. size, format) and application domain specific metadata. Files that are related (e.g. all the files for one design optimisation) can be logically grouped together as a *datagroup*. Metadata can also be added at the datagroup level so that it describes the whole problem, rather than an individual file. The query service performs queries on the metadata database allowing data to be located without the need to remember file or datagroup handles.

Authorisation is implemented as a service interface to a database of registered users, keeping track of permissions on data and mapping between user IDs in the VO and Globus Distinguished Names (DN), which are globally unique identifiers representing individuals. The authorisation service filters query results and only returns metadata about files the user owns and files others have granted them access to. Authentication is achieved with GSI which uses Public Key Infrastructure (PKI) [15] and Secure Sockets Layer (SSL) [16] to provide secure communication over the Grid. Every user must have a private key and a certificate, containing their DN and public key, which is signed by a Certificate Authority.

Java client tools are responsible for transferring files between the local machine and remote file servers using GridFTP. The client side toolkit also calls the web services using SOAP to exchange automatic and user-provided metadata with the databases.

3.2. XML Metadata and XML Schemas

Metadata with a standard, fixed structure such as automatically generated technical file information can be stored in relational database tables. However, user defined application specific metadata is often complex, nested and dynamic in nature, preventing it from being specified by predefined database schemas. We have found that such engineering data can be better represented in XML than in the relational data model. To efficiently manage this non-structured data along side structured data, we have chosen Oracle9i [17], a commercial relational database management system with XML capabilities. User defined metadata is sent to the archive service as XML and stored in an XMLType column. The query service translates queries it receives into a combination of SQL and XPath [18] to be executed on the database. Results are returned from the service as a collection of XML documents containing a specified subset of elements.

Although user-defined metadata may not conform to a predefined XML Schema, they may share some common characteristics within a particular engineering design and hence a schema may be derived from them. The benefits of having an XML Schema for a collection of user-defined metadata include the ability to create graphical query interfaces, identify similar data and perform categorisation, and provide a better storage strategy for improved query performance.

We generate XML Schemas from XML documents using a modified version of the XMLInstance2Schema tool provided by Castor [19], an open source project for data binding between Java, XML and SQL. The XML Schemas describe the structure of some user-defined XML metadata which may be similar to previously stored metadata. If this is the case a single XML Schema can be used to describe the set of metadata instances, which we shall refer to as a collection. Metadata is added to a collection based on the results of the SchemaEvolver utility, which is still under development. This utility analyses the existing collections in two stages; the first stage compares the new Schema with all currently stored XML Schemas and depending on the structure they describe, assigns a similarity rating. The second stage allocates the new metadata to a collection depending on its similarity rating. If there is an exact match then the metadata is assigned to that collection and if there is no similar match then the new XML Schema is added to the database and associated with the metadata, to start a new collection. If there is a similar XML Schema in the database then that XML Schema is modified to describe the structure of both the metadata in the existing collection and the new user-defined metadata. This evolved schema is then added to the database and associated with the collection and the user-defined metadata is added to the collection.

The SchemaEvolver utility currently concentrates on comparing XML Schemas and producing a similarity weighting, the next stage is to produce an evolved schema from two similar schemas so that all versions of the metadata conform to it.

3.3. Application Interfaces

To incorporate the core services we have described into the Matlab environment, we have implemented a range of Matlab functions on top of them so that they can be used programmatically in scripts. Matlab is a powerful scripting environment containing a large number of toolboxes tailored to the needs of scientists and engineers. Its database toolbox [20] uses JDBC to enable insertion and retrieval of data between Matlab and relational databases. This is a helpful tool for those users who already administer a relational database and know how to maintain the schema when their data structures change. However, it is less suitable for engineers who wish to store a variety of data structures that will change

over time, but are not concerned with the underlying database storage schema. Our toolkit allows users to focus on what is stored rather than how it is stored.

Using the Geodise database toolkit, the basic tasks an engineer needs to undertake to manage and share their data are to generate the data using their standard engineering tools, store it in the repository using `gd_archive` function, search for data of interest using the `gd_query` command, and retrieve results to their local file system by `gd_retrieve` function. The wrapping of the core services that enable these tasks is straightforward because much of the logic of the client side components is written in Java, which can be exposed to the Matlab environment using thin Matlab language wrappers. These Java components may be exposed to other high-level scripting environments as required. For example it has been proven straightforward to write wrappers to expose the client side functionality of the Geodise computational toolkit to Jython [8], a pure Java implementation of the Python interpretive scripting environment.

The `gd_archive` function stores a given file in a repository for an authenticated user. The function is able to generate automatically standard metadata for the file, such as its local name, size and format. The user may add additional metadata using Matlab structures and variables, for example custom application specific information, and a list of users who may access the file. The function then transports the file to a server and converts the metadata into XML using our XML Toolbox for Matlab [7] before sending it to the metadata service for storage. The `gd_archive` function returns a unique handle which can be used to retrieve the file at a later date. The locations of databases and file servers can be set in a configuration file by an administrator of the system. The `gd_retrieve` function will locate a file based on a given file handle and return it to a local directory.

The metadata can be queried by an authorised user with the gd_query command, to discover files that have certain characteristics and obtain information about them, such as their handle for retrieval. Users specify the queries in their scripts using a combination of named metadata variables and comparison operators. For example,

```
gd_query ('file.archiveDate >
2003-02-01 & param.radius = 2.3',
'file.ID')
```

will return the IDs (handles) of files which were archived on 2003-02-01 and have variable `param.radius` equal to 2.3. An interactive, graphical query interface is also provided in which selection criteria are specified in a Java GUI generated from standard metadata. When a script based query is performed the XML metadata results are converted back into a Matlab structure before returning them to the user. The bi-directional conversion routines are provided by our XML Toolbox for Matlab. The toolbox includes functions `xml_format` and `xml_parse` to convert Matlab variables into XML and vice versa. We use the XML toolbox as an underlying tool that the user does not see when calling our database functions. As far as the user is concerned they are archiving, querying and working with Matlab structures, not XML.

4. Application Example

The framework described above to utilize database capabilities from within Matlab has initially been developed for Computational Fluid Dynamics applications, however, we demonstrate with a specific example that it can be as easily applied to a Computational Electromagnetics problem.

The specific application we are interested in from the GEM project [21] is the search for good photonic crystal (PC) designs. A PC consists of a periodic micro-structure of holes drilled into a slab of dielectric material. The size and density of the holes determine the transmission and reflection properties for light through the crystal. To investigate the characteristic photonic bandgap (PBG) of the crystal, an engineer samples a range of parameters (e.g. the radius *r* and spacing *d* of the holes) with each sample point giving rise to a different

value of the objective function (the bandgap).

Initially, a large number of designs is explored which yields many solutions and large amounts of data. All of these solutions yield valuable information and need to be preserved in a way that allows for easy searching and retrieval of data and related files.

Figure 3 shows sections of Matlab scripts used for (a) file generation, (b) archiving of files, (c) querying of metadata, and (d) data and file retrieval using the implemented database functionality. The first stage involves the user creating a certificate proxy so they can be authenticated and then generating data files and defining custom metadata about the geometry parameters and resulting bandgap as a Matlab structure, *m*.

Then the spectrum results file is stored using the `gd_archive` function along with the metadata structure, which is transparently converted into XML. `gd_archive` then transfers the file to a server and the metadata to the metadata service for storage in a database.

a
```
gd_createproxy;

m.model = 'pgb_design';
m.param.d = […];
m.param.radius = […];
...
compute_pgb(m.param, infile, outfile);

m.result.bandgap = postprocs_pbg(outfile);
```

b
```
gd_archive(outfile, m);

...
```

c
```
Q = gd_query('model = pbg_design &
result.bandgap < 99.7');
Q: 4x1 struct array with fields standard,
model, param, result
```

d
```
gd_retrieve({Q.file.ID},
'/home/Eng007/pbg_files/' );

visualise_pbg_landscape
('/home/Eng007/pbg_files/*' );
```

Figure 3 Example scripts to generate, archive, query, retrieve and post-process data.

This script is run a number of times with different parameters. After the computations have finished and the design results are available in the database, the engineer can check the results with a simple query (c). The query is formed using a combination of named variables and comparison operators or alternatively through a graphical interface. The returned data *Q* is a vector of structures containing the metadata of all PBG designs which correspond to a bandgap less than 99.7 nm. In this case, four designs match the query and the engineer can retrieve the associated files to the local file system with the `gd_retrieve` command (d).

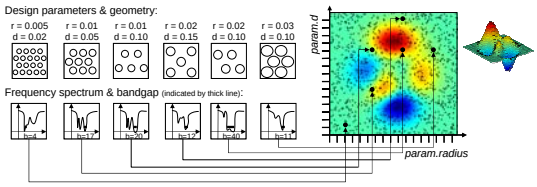

Figure 4 Example of CEM design search using Geodise database technology. Shown are design geometries, the computed frequency spectrum with the bandgap, and representative data for the objective function landscape (dots indicate sample points in parameter space).

Figure 4 shows typical data we obtained for various design parameters. The simulation results from varying r and d form a landscape of the objective function of the photonic bandgap from which a further design optimisation may be performed. As the data and files are kept 'on the Grid', later re-use of the results by the engineer from other locations is also possible.

5. Conclusions and Future Work

In this paper we have described a framework which provides engineers with access to databases on the Grid from within a familiar working environment. We have implemented a suite of services using an architecture which combines a commercial problem solving environment (Matlab) with a core framework of open standards and service oriented technologies, namely Grid computing, Web services, XML and databases. The functions we have implemented to extend the Matlab environment allow engineers to share and re-use data conveniently from existing scripts. The automatic generation of standard metadata and support for user-defined metadata allows queries to be formed that represent the engineer's view of the data.

With a specific example we have shown how design search in electromagnetics can

be supported by the Geodise database toolkit. The transparent integration of database tools into the engineering software environment constitutes a starting point for database applications in engineering design optimisation, and is only one of many potential applications in the engineering domain (CFD, CEM, Civil Engineering, etc.). The toolkit has also been recently used by the GENIE [22] climate modelling project, where data was archived from a Java program then queried, retrieved and visualized in Matlab.

We shall continue our work on generating and evolving XML Schemas. There are some cases when an appropriate schema change cannot be derived by the code and user interaction is required. We intend to incorporate a GUI that has been written for this purpose into our toolkit. The next stage will be to use the XML Schemas to improve query performance and generate query GUIs for custom metadata. We will also investigate how we can reference concepts from our EDSO ontologies to semantically enrich the metadata.

Geodise will provide a graphical user interface to help engineers constructing their workflows, which will need to interact with databases to provide users with up-to-date information. We plan to extend our existing database toolkit to support the workflow construction interface. We will also evolve our Web service based components to OGSA-DAI compliant Grid services, which are a combination of Web services and Grid technology described in the Open Grid Services Architecture (OGSA) [1], the next generation of the Globus Toolkit.

6. Acknowledgements

This work is supported by the Geodise e-Science pilot project (UK EPSRC GR/R67705/01) and the GEM [21] DTI-funded project. The authors are grateful for many helpful discussions with researchers on the Geodise, GEM and GENIE [22] projects. We acknowledge ongoing support from Microsoft, Intel and Oracle.

7. References

[1] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[2] Geodise Project - Grid Enabled Optimisation and Design Search for Engineering, http://www.geodise.org/

[3] M.P. Atkinson, V. Dialani, L. Guy, I. Narang, N.W. Paton, D. Pearson, T. Storey and P. Watson. Grid Database Access and Integration: Requirements and Functionalities. UK e-Science Programme Technical Report http://www.cs.man.ac.uk/grid-db/papers/DAIS:RF.pdf

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

[5] Web Services Activity, http://www.w3.org/2002/ws/

[6] G. Pound, H. Eres, J. Wason, Z. Jiao, A. J. Keane, and S. J. Cox, A Grid-enabled Problem Solving Environment (PSE) For Design Optimisation Within Matlab. To appear in - IPDPS-2003, April 22-26, 2003, Nice, France

[7] M. Molinari. XML Toolbox for Matlab, GEM/Geodise, 2002 http://www.soton.ac.uk/~gridem/Pages/xmltoolbox.htm

[8] J. Hugunin. Python and Java: The Best of Both Worlds. 6th International Python Conference. San Jose, California, USA, 1997.

[9] L. Chen, N. R. Shadbolt, F. Tao, S. J. Cox, A. J. Keane, C. Goble, A. Roberts, P. Smart. Engineering Knowledge for Engineering Grid Applications, Euroweb 2002, p12-24, 2002.

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

[11] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch. National-Scale Authentication Infrastructure, *IEEE Computer*, 33(12):60-66, 2000.

[12] GridFTP, http://www.globus.org/datagrid/gridftp.html

[13] M. Mealling, P. J. Leach and R. Salz. A UUID URN Namespace, IETF, October 2002. http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-00.txt

[14] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H.F. Nielsen. SOAP Version 1.2, W3C Candidate Recommendation, 2002
[15] IETF PKIX Working Group. http://www.imc.org/ietf-pkix/
[16] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0, November 1996.
[17] Oracle9i Database, http://otn.oracle.com/products/oracle9i/
[18] J. Clark & S De Rose, XML Path Language (XPath) Version 1.0, *W3C Recommendation*, 1999 http://www.w3.org/TR/xpath/
[19] The Castor Project, http://castor.exolab.org
[20] Matlab Database Toolbox, http://www.mathworks.com/products/database/
[21] GEM project - Grid Enabled electroMagnetic optimisation, http://www.soton.ac.uk/~gridem
[22] GENIE project - Grid ENabled Integrated Earth system model, http://www.genie.ac.uk