Delivering Data Management for Engineers on the Grid[1]

Jasmin Wason, Marc Molinari, Zhuoan Jiao, and Simon J. Cox

School of Engineering Sciences, University of Southampton, UK
{j.l.wason, m.molinari, z.jiao, sjc}@soton.ac.uk

Abstract. We describe the design and implementation of a database toolkit for engineers, which has been incorporated into the Matlab environment, to help manage the large amount of data created in distributed applications. The toolkit is built using Grid and Web services technologies, and exchanges XML metadata between heterogeneous Web services, databases and clients using open standards. We show an application exemplar of how this toolkit may be used in a grid-enabled Computational Electromagnetics design search.

1 Introduction

Engineering design search and optimization (EDSO) is the process whereby engineering modelling and analysis are exploited to yield improved designs. This may involve lengthy and repetitive calculations requiring access to significant computational resources. This requirement makes the problem domain well-suited to the applications of Grid technology which enable large-scale resource sharing and coordinated problem solving within a virtual organisation (VO) [1]. Grid technology provides scalable, secure, high-performance mechanisms for utilizing remote resources, such as computing power, data and software applications over the Internet.

While compute power may be easily linked into applications using grid computing middleware, there has been less focus on database integration, and even less still on providing it in an environment familiar to engineers. Traditionally, data in many scientific and engineering disciplines have been organized in application-specific file structures, and a great deal of data accessed within current Grid environments still exists in this form [2]. When there are a large number of files it becomes difficult to find, compare and share the data. If database technologies are used to store additional information (metadata) describing the files, they can be located more easily using metadata queries. The Storage Resource Broker (SRB) [3] uses its Metadata Catalog (MCAT) for dataset access based on attributes rather than names or physical locations. However, MCAT has limited support for application specific metadata which is often essential in assisting engineers to locate data specific to their problems. In Geodise [4] our goal is to develop sophisticated but easy-to-use toolkits to help engineers make use of grid resources from environments they use daily, such as Matlab [5] and Jython [6]. These consist of the Geodise computational toolkit [7], XML toolbox [8] and a database toolkit, the focus of this paper, in which we adopt an open standards and service oriented approach to leverage existing database technologies and make them accessible to engineers and suitable for their problems.

---

[1] The full paper is available at [4]. This work is supported by the Geodise e-Science pilot project (UK EPSRC GR/R67705/01) and the DTI-funded GEM project.

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 412–416, 2003.

2 Architecture

A major aim of the Geodise data management architecture is to bring together flexible, modular components for managing data on the Grid which can be utilized by higher level applications. Another objective is to provide a simple, transparent way for engineering users to archive files in a repository along with metadata. Various technical and application specific metadata about files, their locations and access rights are stored in databases. Files are stored in file systems and transported using the Globus Toolkit [9] which provides middleware for building computational grids and their applications. We use the platform independent Java CoG kit [10] to utilize the Grid Security Infrastructure (GSI) [11] for authentication, and GridFTP [12] for secure file transfer. As shown in Fig. 1, client side tools initiate file transfer and call Web services [13] for metadata storage, query, authorisation and file location.

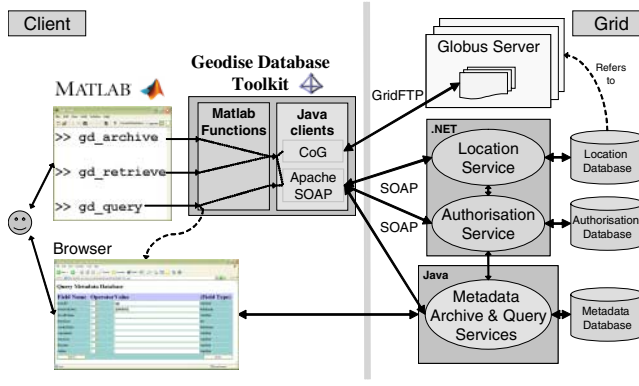

Fig. 1. A high level set of scripting functions sits on top of a client side Java API to provide an interface to data management Web service functionality and secure file transfer.

Access to databases is provided through Web services, which may be invoked using the Simple Object Access Protocol (SOAP) [14], to transfer data between programs on heterogeneous platforms. For example, our Java client code running on Linux or Windows can communicate with .NET Web services on a Windows server and Java Web services on a Linux server.

A unique handle is all that is required to locate and retrieve a file, as the *file location service* keeps a record of where it is stored. The *metadata archive service* allows the storage of additional descriptive information detailing a combination of technical characteristics (e.g. size, format) and user defined application domain specific metadata. The *metadata query service* provides a facility for engineers to find the data required without the need to remember the file names and handles.

We use relational tables for structured data and XML for more flexible storage of complex, deeply nested engineering specific metadata. We require a set of services that allow us to access and interrogate both types of data storage in a standard way. We currently provide APIs to specific databases with tailored Web services and will use these on top of implementations compliant with proposed standards from the Data Access and Integration Services Working Group [15] of the GGF [16].

The *authorisation service* uses a database of registered users, containing data access permissions mapping between VO user IDs and authenticated Globus

Distinguished Names, globally unique identifiers representing individuals. Query results are filtered and only metadata about files the user has access to are returned.

3 Problem Solving Environment and Application Example

The basic tasks (Fig. 2) for an engineer to undertake to manage and share their data are to (A) *generate the data* using standard engineering tools, (B) *store it* in the repository, (C) *search for data* of interest, and (D) *retrieve results* to the local file system. The wrapping of the core services enabling these tasks is straightforward as much of the logic of the client side components is written in Java, which can be directly exposed to Matlab or other high-level scripting environments, such as Jython [6]. Matlab contains a large number of toolboxes tailored to scientists and engineers, and makes it easy for them to quickly generate, analyse and visualize their data. We have implemented a range of Matlab functions on top of our core database services so that they can be incorporated programmatically into the user's scripts in a way consistent with the behaviour and syntax of the Matlab environment.

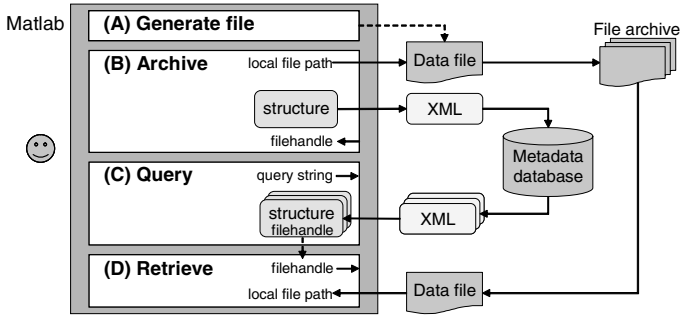

Fig. 2. Data flow of files and metadata: (A) file generation, (B) archive of file and user metadata, (C) querying of metadata, and (D) file retrieval.

We have applied the Geodise database toolkit in a real world example of design search in Computational Electromagnetics (CEM). The GEM project [16] is developing software for industrial use to improve the design of optical components for next generation integrated photonic devices. A specific device is the photonic crystal with certain light transmission and reflection properties which depend on the size and distribution of structures drilled into a slab of dielectric material. To investigate the characteristic photonic bandgap (PBG) of the crystal, an engineer samples a range of parameters (e.g. radius *radius* 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 are explored which yield many solutions and large amounts of data. All of these solutions, whether good or poor, may yield valuable information for future simulations or devices and need to be preserved.

Fig. 3 shows two scripts, one to create data and archive it, the other for query and retrieval of data. The first stage (*a*) 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 (*b*), along with the metadata structure, which is converted into XML by our XML Toolbox for Matlab [8]. `gd_archive` then transports the file to a server and the metadata to the metadata service for storage in an XML 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.standard.fileID}, '/home/Eng007/pbg_files/' );
visualise_pbg_landscape ('/home/Eng007/pbg_files/*' );
```

Fig. 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 can be formed using a combination of named metadata variables and comparison operators or alternatively through a graphical interface. `Q` is a vector of structures containing the metadata of all PBG designs with a bandgap less than 99.7. In this case, four designs match the query. For further investigation or visualization, the Engineer can retrieve files associated with the above four designs to the local file system (*d*).

Fig. 4 shows typical data we can obtain for the various design parameters. The simulation results form an objective function landscape of the photonic bandgap from which a full design search and optimization may be performed. The storage of the results in a database as well as the transfer of files to a file store on the Grid additionally allows data re-use by engineers.

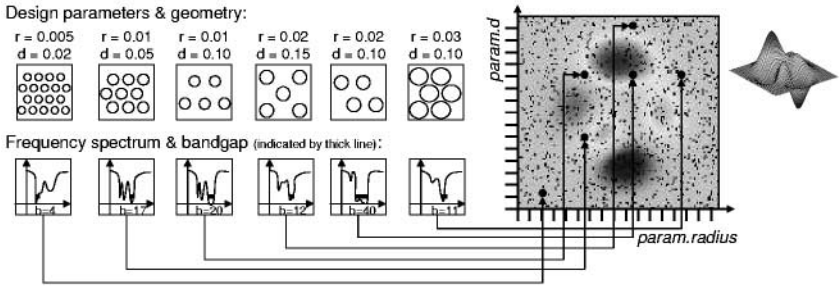

Fig. 4. Example of CEM design search using Geodise database technology. Shown are design geometries, the computed frequency spectrum with the bandgap, and representative data for the objective function landscape (dots indicate sample points in parameter space).

5 Conclusions and Future Work

We have described a framework which allows the use of databases on the Grid in an engineer-friendly environment. We have implemented a suite of services which combine a commercial PSE (Matlab) with a core framework of open standards and service oriented technologies. 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 EDSO, and is only one of many potential applications in the engineering domain (CFD, CEM, etc.). The functions we have implemented to extend the Matlab environment allow engineers to share and re-use data conveniently. 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.

Future work will involve providing tools to generate, compare and merge XML Schemas describing users' custom metadata. We will also evolve our Web service based components to GGF DAIS-WG compliant Grid services.

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] 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
[3] The Storage Resource Broker, http://www.npaci.edu/DICE/SRB/
[4] Geodise Project http://www.geodise.org/
[5] Matlab 6.5. http://www.mathworks.com
[6] J. Hugunin. Python and Java: The Best of Both Worlds. 6th International Python Conference. San Jose, California, USA, 1997.
[7] G. Pound, H. Eres, J. Wason, Z. Jiao, A. J. Keane, and S.J. Cox, A Grid-enabled Problem Solving Environment (PSE) For Design Optimization Within Matlab. IPDPS-2003, April 22–26, 2003, Nice, France.
[8] M. Molinari. XML Toolbox for Matlab. http://www.soton.ac.uk/~gridem
[9] The Globus Project. http://www.globus.org/
[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] Web Services Activity, http://www.w3.org/2002/ws
[14] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H.F. Nielsen. SOAP Version 1.2, W3C Candidate Recommendation, 2002
[15] Global Grid Forum, http://www.ggf.org/
[16] The GEM project, http://www.soton.ac.uk/~gridem