A Web Service Architecture for GECEM

Maria Lin and David W. Walker
School of Computer Science
Cardiff University
PO Box 916
Cardiff CF24 3XF

Yu Chen and Jason W. Jones
Civil and Computational Engineering Centre
School of Engineering
University of Swansea
Swansea SA2 8PP

Abstract

The GECEM project is developing Grid technologies to enable large-scale and distributed engineering research in computational electromagnetics (CEM). We present a Web service portal architecture for supporting collaboration among distributed partners using the Grid. The GECEM grid takes a model geometry generated at one location, and transfers it to a second location to be meshed. The computational mesh is then transferred to a third location where it is used to solve a CEM problem.

1 Introduction

Large-scale computational electromagnetics (CEM) simulations are computationally intensive and can involve access to resources that are intrinsically distributed. For example, in a CEM simulation, the geometry of a component may be created at one location, a mesh conforming to this geometry may be generated at a second location, and a CEM simulation based on the mesh may be performed at a third location. Finally, the output from the simulation may be analysed and visualised at one or more other locations.

One aim of the GECEM project is to build an application portal to access the partially-shared distributed hardware, software, and data resources used in a CEM simulation. The main user-level software resources are the mesh generation and CEM solver application codes, while the main data resource is the geometry file specifying the physical system to be simulated. In the GECEM grid both the software and data resources are exposed as Web services, and the GECEM portal hides the complexity of the service-oriented Grid system from the end user. The work presented here extended that done previously at Cardiff and Swansea Universities and presented at AHM 2003[6] and AHM 2004[3].

2 Design and Implementation

The portal interface to the GECEM grid has been created using GridSphere [2] running on Apache Tomcat. The GridSphere framework is based on a “portlet model”, which provides a portlet implementation as well as an architecture for supporting the development of re-usable portlets and portlet ser-

vices. A portlet is a window that provides a user-friendly interface to allow a user to access distributed resources in a grid-enabled environment. Currently the supported functions, including credential delegation, data transfer, computing resource allocation and job request distribution, have been prototyped as a JSP/JavaBean based framework on top of the Java CoG Kit, Globus, and MyProxy.

2.1 GECEM Portal

Figure 1 presents the GECEM portal architecture in which the users logon to GridSphere, search the registry for files and Web services, select the files and contact the Web services for mesh generation and CEM simulation.

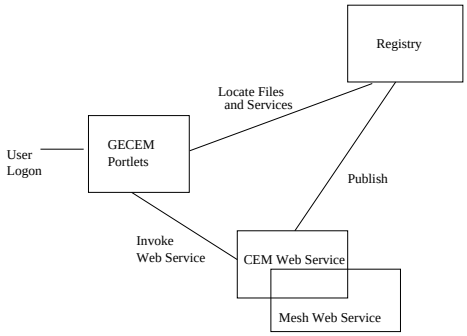

Figure 1. GECEM portal architecture.

2.2 GridSphere Portal

The GridSphere portal offers the following features:

User Management The portal makes use of the infrastructure provided by GridSphere which includes a set of core portlets and portlet services that provide the basic infrastructure required for developing and administering Web portals. It also provides useful information to manage user activities such as session management which provides context for a user, a logon service with role-based access, and subscription of portlets.

Security The GECEM service architecture is based on credentials using Globus based GSS-API credentials and the MyProxy credential repository. By configuring GridSphere with the GridPortlets application, users can logon using a MyProxy credential. The GridPortlets application provides a number of portlets to access core grid services. Particularly useful for our portal are the credential portlets, which provide a certificate management service and MyProxy configuration for single logon. Both the portlet client and the Web service require the user to retrieve their credentials to invoke the Web services.

2.3 GECEM Portlets

Portlets are a special type of servlet which are easily pluggable. They are implemented using a Portlet API provided by GridSphere. The following portlets are implemented.

- The *resource portlet* allows a user to select a machine in the GECEM grid.
- The *data portlet* then displays the geometry data files, input files for mesh generation, and input solver files available on the selected machine, and allows the user to select from these to specify a complete job.
- The *UDDI portlet* allows the user to search the UDDI registry for information on the location of the available mesher and solver services, thereby allowing them to be discovered dynamically. A user can publish services and query services through the UDDI registry, and select Web services according to some user-specified criteria.
- The *web service portlets* allow a user to initiate the web services described in Section 4 for meshing and CEM simulation and to monitor and analyse the results of the execution. The portlets make use of the resource and service registries to locate the available files and the services.

3 Web Services

Two kinds of service shave been deployed in GECEM. One is from the set of Globus core grid services and the other has been developed from legacy codes.

Open Grid Services Architecture (OGSA) recasts Grid capabilities as Web Services using WSDL descriptive conventions. The Globus core grid services deployed here include the Reliable Transfer File (RTF) service, which is an OGSA-based service that provides an interface to transfer files from one location to another. The RTF service is provided by the GridPortlets web application.

For generating meshes and solving CEM simulations, legacy libraries are employed and will be described in Section 4.

4 The Adaptation of Applications to Grid Environment

We have adapted two legacy applications to an OGSI (Open Grid Services Infrastructure) compliant architecture to achieve collaborative mesh generation and numerical simulation across a geographically dispersed Virtual Organization. These legacy codes not only typically represent large-scale investments that cannot be discarded, but also are high-quality software. Two strategies are adopted to grid-enable the legacy applications, based on Globus middleware: batch-oriented and service-oriented.

4.1 Batch Oriented

With the batch-oriented strategy, existing applications are run on available computers in a Grid environment. Originally our legacy applications were invoked from the command line. Input files and other configuration information were specified in the command line parameters. We adopt a well-documented and industry standard approach, Resource Specification Language (RSL) [1], to express our requirements. Resource consumption information and other job information are provided, such as the name of the executable file to run, the files to stage in and stage out, file clean up, maximum memory required, CPU usage, etc. A job defined in this way can be sent either via Globus commands or submitted programmatically to Globus GRAM. Globus provides convenient and efficient mechanisms to manage the job.

In this strategy, existing applications do not require any modification, and the application contains no references to grid middleware. The grid middleware is responsible for security, resource control and scheduling, and so on. Existing applications are run simply as executables in a Grid environment. Executables can be in the form of a script (JAVA, Perl, etc.), or an executable application that tends to have special requirements for different platform versions, such as collections of libraries, JAR files, and any other environmental conditions. The heterogeneous nature of computing resources still remains a significant barrier in this context. Pre-process work on executables and input data is required to tailor to various environments. The job broker may need to allocate suitable resources for the executable according to its specific requirements.

The same applications are structured to run concurrently, which means multiple instances run independently on behalf of different users. We set up a unique account for each user to make sure that multiple users can run their own data using different instances of an application concurrently without interference. This gives the overall application far more throughput than having only one instance running at a time.

4.2 Service Oriented

In a service-oriented strategy [5], the client and server are loosely coupled. For service providers persistent factory services (for volume mesh generation service and electromagnetic simulation service) are defined and provided, which create transient service instances on demand. The client side uses the grid middleware to invoke the service and receive a serviceLocator for newly-created service instances. The factory implements the *Factory* portType, which provides a standard operation to create Grid service instances. *NotificationSource* portType and *GridService* portType are also implemented. The WSDL is created for the application grid services. These services are compatible with OGSA's well-defined interfaces and specific conventions addressing discovery, dynamic service creation, lifetime management, notification, and manageability. They also ensure high security based on authentication, authorization, and incorporate credential delegation.

In our case, the legacy mesher and solver applications were previously written in FORTRAN. These codes have required minimal modifications and are wraped as a C library. We have developed runtime support for easily plugging simulations into the Grid Services Framework. Java Native Interface [4] is adopted to enable the integration of grid service code written in Java with legacy code written in other languages, and to Java code to operate with existing applications and libraries. The details are described below:

- Define the Java wrapper with native methods. This Java wrapper loads and links to the native implementation.
- Run the `javah` to get the header file, which will be included in the C wrapper.
- Change the FORTRAN `main` to a subroutine, with command line arguments passed as parameters.
- Create a C wrapper to invoke FORTRAN subroutines. Add control code in the C wrapper to control the FORTRAN computation loops and data transfer.
- The FORTRAN code and C wrapper are compiled together as a dynamic link library, which can be loaded and linked into the Java Virtual Machine. This shared library needs to be present where the grid service is provided.
- Finally the Java wrapper is exposed as the grid service implementation.

Service providers can publish detailed service descriptions to allow easy discovery through community registries. Our services can be easily invoked via a command line interface, grid portal or other remote procedure call without going through repeated program initiation and termination. But such grid services cannot be run simply across the grid environment like a batch job. They must be installed and deployed within the Grid service host environment (such as Apache Tomcat servlet container) on the compute node that runs the service-oriented application.

5 Conclusion and Future Work

We have presented the design and implementation of a web-based portal architecture to use resources, and have introduced several core portal services. Future work will focus on integrating other web services that are useful for CEM simulation, and on extending our work to other domains.

References

[1] Globus. WS GRAM: Developer's guide. `http://www-unix.globus.org/toolkit/docs/3.2/gram/ws/developer/mjs_rsl_schema.html#SchemaDefinitions`.

[2] GridLab. The gridsphere portal. `http://www.gridsphere.org/gridsphere/gridsphere`.

[3] Maria Lin and David W. Walker. A portlet service model for GECEM. In *Proc. UK e-Science All Hands Meeting 2004*, August 2004.

[4] Sun Microsystems. `http://java.sun.com/developer/onlineTraining/Programming/JDCBook/jni.html`.

[5] Borja Sotomayor. The globus toolkit 3 programmer's tutorial. `http://www.casa-sotomayor.net/gt3-tutorial/`.

[6] David W. Walker, Jonathan P. Giddy, Nigel P. Weatherill, Jason W. Jones, Alan Gould, David Rowse, and Michael Turner. GECEM: Grid-Enabled Computational Electromagnetics. In *Proc. UK e-Science All Hands Meeting 2003*, pages 436–443, September 2003.