Proceedings of the UK e-Science All Hands Meeting 2005, ©EPSRC Sept 2005, ISBN 1-904425-53-4

Combining physical simulations and chargeable Web service applications in engineering workflows

Marc Molinari and Simon J Cox

Southampton Regional e-Science Centre
School of Engineering Sciences
University of Southampton, SO17 1BJ
m.molinari@soton.ac.uk

Abstract

We demonstrate with a case study extending previous work how to integrate and consume a distributed framework for chargeable Web services providing a specialist software package for engineering applications and show with an example how this service can be integrated into scientific research problems in science and engineering. We discuss the complexity of the problem and list some of the issues encountered.

1. Introduction

Physical scientific simulations and engineering design optimisation processes frequently require specialist expensive software packages which often run only on dedicated hardware. Businesses have the increasing choice of using such software as services via well-defined web interfaces instead of the ordinary licensing channels. This can reduce spending on purchase, maintenance and upgrade expenses, possible on a pay-per-use basis, thus effectively reducing the cost of ownership in the long-run.

The use of chargeable web services on demand will complement the purchase of software in the future and will open the market to small enterprises for which the access to these resources on a full-purchase basis might not be commercially viable. Demand from Grid industry is on the rise as a recent publication in BusinessWeek [1] shows.

This paper is a follow-on article on previous experiences [2]. We demonstrate how a client user can integrate and use a specialist finite element meshing web service provided through the University of Swansea or the University of Southampton with payment infrastructure located at Imperial College and resource usage service located at Manchester. The application, compute and database facilities and tools are part of the GeodiseLab project [3], the framework for chargeable web services is part of the Computational Markets project [4].

As part of the client functionality, we present a client script which integrates the negotiation and Web service interaction process into an engineer's scripting environment. The integration and orchestration takes place behind the scenes of a standard commercial Matlab problem solving environment (PSE). Thus, it can also take advantage of the OMII-sponsored GeodiseLab project with interaction of a wider range of Grid technologies such as database and compute pool usage.

Our application demonstrates how to use the available Web service in an engineering workflow to search for novel physical properties in the area of computational electromagnetics and we show an application of design optimisation.

We will show how the components of this distributed framework interact as part of the user's application and how to implement this in a real engineering workflow.

2. Physical Simulation

A sample problem in the field of electromagnetic design optimization is the parametric design study of a component for next-generation integrated photonic devices [5]. This approach, however, can be extended and applied in the same way to other science or engineering applications.
In this application, the scientist wants to optimise the electromagnetic transmission properties of a Photonic Crystal device for a specific

high-performance optical communication device. A base geometry of the Photonic Crystal that leads to bandgaps in the energy spectrum is a periodic array of holes etched into a slab of dielectric material. The scientist will be able to vary the hole distance and radius to obtain different energy bandgaps. To perform such an investigation and design optimisation numerically before manufacturing the device, the scientist will need to perform a number of computational tasks as indicated in figure 1.

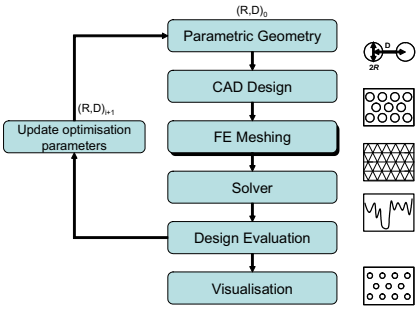

Figure 1: Computational tasks of a scientific workflow in design optimisation, including the call to finite element meshing software.

The optimisation of the design is an iterative process during which the parameters are updated to give and improved design. Each iteration requires the use of a number of software components which can be very complex and often require the purchase of specialist software or even hardware to run on.
Here, we focus on the finite element meshing software package, indicated by the field "FE Meshing". We detail in the following how this component can be integrated as a web service into a GeodiseLab workflow script of a scientist or engineer.

3. Web Service Framework

A functionality often required by engineers is a finite element meshing application which takes a geometric design as input and converts this into a topological assembly of finite elements representing the area or volume under investigation. We have implemented such a meshing application as a Web service and extended its functionality with the framework provided by the Computational Markets project so that its use can be charged for on both a pre-pay and pay-per-use basis.

This framework includes the following functionalities usable in this example.

- Negotiation
- Payment Service
- Application Web Service
- Resource Usage

3.1 Negotiation cycle

In the negotiation cycle which takes place before any others, clients can specify limits to the parameters within which the meshing service should operate. These usually include the price the client is willing to pay and the number of runs required as part of a bundle as well as an expiry of the agreement made. Possible extensions are the size of mesh or output data and the availability of the service for a specific time period.

3.2 Payment Service

The payment service delivers the accounting backbone for (virtual) money transfers. Accounts of service providers and client users are maintained and the money transfers that take place are authorised by X.509 certificates which are countersigned by the owners. All traffic is encrypted on the message level (via SSL connections) and also signed using these certificates.

3.3 Application Web Service

The application Web service wraps a meshing executable exposed to the user by a standard WS interface WSDL document. The owner has the possibility to set the negotiation parameters through an incorporated web page interface.

Technical details on the framework and the provided services can be found in the Computational Markets paper in these proceedings.

3.4 Usage diagramm

Figure 2 shows the distributed framework currently in use by our example application. The payment service is hosted at LeSC, the negotiation takes place between the user and the application web service site, and the resource usage is registered in Manchester. We have the choice of integration two meshing web service applications hosted in Southampton (for 2D meshing) and in Swansea (for 3D meshing).

The numbers in figure 2 describe the procedural order of process steps that has evolved from our web service integration tests.

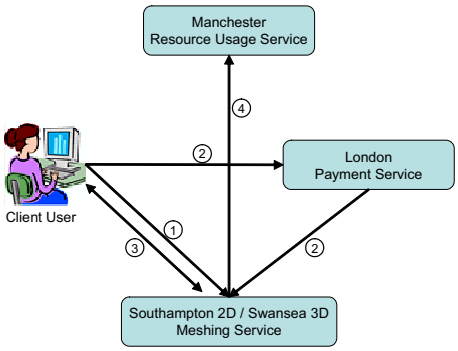

Figure 2: Distributed Computational Markets framework for chargeable Web Services.

The process of integrating this web service in the scientist's script can be described as:

0. Find Web Service (UDDI register).

1. Negotiation
   a. Contact WS for an offer
   b. User accepts offer or specifies new requirements and re-negotiates
   c. Service provider sends acceptable offer
   d. User accepts offer
   e. User provides account details

2. Payment authorisation
   a. User authorises transactions from own account to service provider's account
   b. Payment service checks details with provider
   c. User obtains contract details in form of a digital ticket

3. Consumption
   a. User prepares service by sending agreement ticket
   b. User sends input data stream to service
   c. Invoke service with *process* command
   d. Poll service until finished
   e. When service finished, WS contacts payment service to transfer money
   f. User retrieves data

4. Resource Usage
   a. WS sends usage data to resource usage service (pending).

4. Combining Application and Web Services in Engineering Scripts

In section 2 we have described the parametric design search to solve the problem of finding an optimal solution for a photonic crystal design. Section 3 contained the details of the chargeable web service framework which offers the finite element meshing application for our workflow.

To combine these two together, an engineer working with a scripting problem solving environment such as Matlab or Python has to be able to integrate the functionality easily. As an engineer is usually more familiar with their scripting language rather than Grid programming knowledge, we tried to incorporate the complex client functionality in a small number of essential functions.

These aim to separate the negotiation and payment authorisation from the actual consumption of the Web service application. Consuming the meshing service may then actually be performed automatically without user interaction if enough executions have been purchased as part of the agreement.

In the Matlab scripting environment, the procedure looks as follows, where the function calls contain the necessary client calls to the Java API's provided by the Computational Markets framework:

Negotiation & payment authorisation

```
% load user details such as keystore location
userDetails = cfg_load('user.cfg');

% get offers, if unsuitable re-negotiate
% prints out offers in PSE and waits for
% user confirmation
selectedOffer = getOffers( userDetails );

% accept offer and authorise payment
TicketFile = acceptOffer( selectedOffer );
```

Consume Web service from within PSE

```
% update selectedOffer
selectedOffer.ticket = TicketFile;

% specify input files
InputFiles = {'fe_mesh.inp', 'fe_mesh.ctrl'};
OutputFile = 'fe_mesh_result.msh';

% consume service and store results
consume_meshing_ws( InputFiles, OutputFile, …
 seletedOffer )
```

This creates – if successfully completed – an output file with the finite element mesh data in the local directory. The transaction of the payment has taken place as can be seen on the payment service account web page interface.

This functionality can now easily be extended by the engineer to include access to databases for parameter storage and retrieval as well as metadata for the files created. The analysis of the photonic crystal design is usually carried out on a Condor or Globus compute resource through the Geodise Compute or CondorNative toolboxes. Figure 3 shows the possible extension of the local PSE to integrate these diverse distributed components into the workflow.
Further details can be found in [6] in these proceedings.

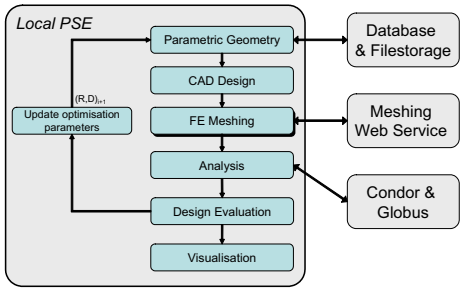

Figure 3: Incorporation of Grid data, compute and Web service resources into scientific workflows in the Matlab PSE, supported by Geodise Toolboxes and the Computational Market client.

5. Experiences

As with all distributed and network-based environments, there are a number of technical, usability and maintenance issues we have encountered:

- Development
  - Remote debugging
  - Certification (SSL, X.509)
- Maintenance of Services
  - Changes in one location affect several others
  - Contract expiry
- Charging Issues
  - non-transparency (sometimes)
  - non-runs charged for
  - error during execution
  - trial periods
- Contractual arrangements
  - Expiry of contracts
  - Per-user / per-group agreements
- Client Software
  - Versioning of libraries
- Script integration
  - Training of implementation details

Each of these has been more or less difficult to address in the course of this work.

6. Conclusions & Future Work

We have set out to show how a scientific problem and a web service which holds the functionality to part of the solution can be combined to solve an engineering design search. Within the framework of the Computational Markets project which provides facilities to make this meshing web service chargeable with distributed negotiation, payment and resource usage services and the usage of the GeodiseLab Toolboxes for database, file storage and compute resource access, we managed successfully to federate the services and combine them in a Matlab script.

It is worth noting that the OMII software stack is likely to supersede the GMarkets framework by offering a freely downloadable, open source web service container with WS-Security enhancements. It will provide process based access control and extension for secure and accountable file and compute grid access.

References

[1] Hamm S (2004). Getting A Grip On Grid Computing. BusinessWeek Online, 18 Oct 2004

[2] Molinari M, Nammuni K, Cox S (2004). Integration of chargeable Web Services into Engineering Applications. e-Science All Hands Meeting, Nottingham Sept. 2004.

[3] The GeodiseLab Project, www.geodise.org, Southampton eScience Centre

[4] The Computational Markets Project, www.lesc.ic.ac.uk/markets, London eScience Centre

[5] Parker G and Charlton M (2000). Photonic Crystals. Physics World 13(8), 29-34

[6] Pound G, Wason J, Molinari M, Jiao Z, Cox S. GeodiseLab: Making the Grid Usable. In these proceedings.

[7] See www.omii.ac.uk for the omii_2 software stack.