

Grid Services and Microsoft .NET

Daragh Byrne Ally Hume Mike Jackson
D.Byrne@epcc.ed.ac.uk A.Hume@epcc.ed.ac.uk M.Jackson@epcc.ed.ac.uk

EPCC,
University of Edinburgh, James Clark Maxwell
Building,
Kings Buildings,
Edinburgh
EH10 5JE

Tel: +44(0)-131-650-5141
Fax: +44(0)-131-650-6555
WWW Site: <http://www.epcc.ed.ac.uk/~ogsanet/>
E-mail: ogsanet-queries@epcc.ed.ac.uk

Collaborators: EPCC Microsoft Research Limited National e-Science Centre

Abstract

With the advent of the Open Grid Services Architecture and its underlying infrastructure – the Open Grid Services Infrastructure – there is an increasing interest within Grid communities worldwide in Microsoft .NET Web Services technologies and their applicability to Grid services. In this paper we describe a collaboration between Microsoft Research Limited and NeSC (represented in this project by EPCC) which has set about exploiting this interest by:

- Developing an implementation of the Open Grid Services Infrastructure (OGSI) using .NET technologies.
- Developing a suite of Grid Service demonstrators – including an OGSA-DAI demonstrator – that can be deployed under this .NET OGSI implementation.
- Developing training courses and materials to educate and inform the UK e-Science community about .NET and its applicability to Grid applications.

We describe our OGSI on Microsoft implementation and present an overview of our first OGSI on Microsoft .NET training course to be held on 9th-10th September 2003.

funded by the UK Department of Trade and Industry)
which is exploiting this interest by:

1 Introduction

With the advent of the Open Grid Services Architecture (OGSA) [1] and its underlying infrastructure – the Open Grid Services Infrastructure (OGSI) [2] – there is an increasing interest within Grid communities worldwide in Microsoft .NET Web Services technologies [3] and their applicability to Grid services. The MS.NETGrid project is a collaboration between Microsoft Research Limited and NeSC (represented in this project by EPCC and

- Developing an implementation of OGSI using .NET technologies.
- Developing a suite of Grid Service demonstrators – including OGSA-DAI demonstrators [4] – that can be deployed under this OGSI implementation.
- Developing training courses and materials to educate and inform the UK e-Science community about .NET and its applicability to Grid applications.

- Delivering training courses to delegates from the UK e-Science community.

The goal of the MS.NETGrid project is to provide a practical demonstration to the UK e-Science community of the applicability of Microsoft .NET technologies to the hosting, development and deployment of Grid Services. A complementary goal is facilitating understanding of the Grid and e-Science within Microsoft.

The project – which started in March 2003 – has a duration of 12 months and is making its deliverables freely available to encourage further use of Microsoft .NET within the UK e-Science programme. In addition, to facilitate the development of an OGSI .NET community, close links are being fostered with both Globus [5] and the Grid Computing Group at the University of Virginia [6] who are also developing an OGSI implementation on .NET.

In section 2 Grid services, the Open Grid Services Infrastructure and Microsoft .NET are briefly introduced. In section 3 the design of an implementation of OGSI on Microsoft .NET – MS.NETGrid-OGSI – is described. Section 4 gives an overview of Grid service demonstrators developed by the project and which run on MS.NETGrid-OGSI. Section 5 provides information on the content of the first of a series of OGSI on Microsoft .NET training courses which will be run by the project from September 2003 to February 2004.

2 OGSI on Microsoft .NET

Our work is focused around an implementation of the Open Grid Services Infrastructure under Microsoft .NET called MS.NETGrid-OGSI. In this section the background of MS.NETGrid-OGSI is described.

2.1 Grid Services

Grid computing is a way of organizing heterogeneous computing and data resources so that they can be flexibly and dynamically allocated and accessed [1][7]. Often this sharing of resources occurs across organisational boundaries. A collection of organisations contributing resources in such a manner is often known as a *virtual organisation* [7]. The connection of heterogeneous resources requires an open, platform-independent global architecture. Efforts to define this architecture have been made in

[1][2], yielding an *Open Grid Services Architecture (OGSA)* in which resources are represented by *services*. The term service is used to describe a network-aware entity (for example a software component) that provides certain well-defined *functionality*, accessed via a known *interface*, which accepts and processes a certain set of *messages*. A service may also have a *state* associated with it which changes in response to received messages. An important distinction between Grid services and Web services is that Grid services have *state* and support explicit *lifetime management* by clients. In addition, the behaviour of a service should not be strongly-coupled to its implementation. An example of a common service is a File Transfer Protocol service.

The *Open Grid Services Infrastructure (OGSI)* [2][10] defines such a service-oriented view of Grids in terms of modern Web services technologies, Web services being small, discrete, building-block applications that connect to each other – as well as to other, larger applications – via the Internet. OGSI is intended to be the building block upon which OGSA Grid services can be constructed. OGSI defines the capabilities and interfaces of Grid services in terms of an enhancement to the Web Service Description Language (WSDL) [11]. WSDL is used to define input and output messages which are, in turn, used to define *operations*. These operations are aggregated into *portTypes* which describe some part of an interface to a service. PortType definitions can be combined with concrete network protocols – for example HTTP – and message formats – for example SOAP [13]– to form *bindings* which can then be associated with network addresses to yield *ports*. A service is represented by an aggregation of ports. OGSI extends¹ WSDL to allow for portTypes to extend other portTypes and for so-called *service data* – used to represent the state of a Grid service – to be associated with a portType.

One OGSI portType – GridService – is required to be implemented by all Grid services – this portType provides operations for the querying and updating of service state and lifetime management. In addition OGSI defines a basic collection portTypes for the construction of fundamental Grid services including factories, service groups, location services, and notification services. It is intended that these fundamental Grid services will form the building

¹ These extensions are proposed for WSDL 1.2.

blocks for higher-level OGSA services such as job schedulers.

2.2 Microsoft .NET and Web Services

Fundamentally, Microsoft .NET [3] is a set of Microsoft software technologies for connecting an individual's world of information, people, systems, and devices. It enables an unprecedented level of software integration through the use of XML Web services, provided by a component of .NET called ASP.NET. .NET's success is central to Microsoft's corporate strategy and its development has been the company's major focus for the past 3 years. Microsoft .NET provides a suite of tools for Web service description using WSDL, Web service development, deployment and client authoring.

2.3 Grid Services on Microsoft NET

If Grid services are to be platform-agnostic, implementations of OGSI on a wide range of contrasting platforms are necessary to demonstrate this in practice. Implementing OGSI on .NET provides such a demonstration on a platform that not only is Microsoft's platform of choice for software development for the foreseeable future but is becoming increasingly prevalent within the UK e-Science community. Indeed, some e-Science developers are already using it as a quick and simple prototyping platform that provides well-structured, easy to use Web service functionality.

3 The Design of MS.NETGrid-OGSI

Since the MS.NETGrid project is concerned with demonstrating the applicability of Microsoft .NET to Grid services development, our central design goal was to exploit and demonstrate as many relevant .NET – particularly ASP.NET – capabilities as was possible and sensible, especially when given that Grid services are essentially an extension of Web services. Other important design goals were:

- To provide an implementation that demonstrates the key concepts of OGSI namely the core portTypes, service state and service data management and lifetime management.

- To support SOAP over HTTP as our communications protocol, this being the prevalent protocol in the area of Grid services at time of writing.
- To aim for interoperability as far as possible, highlighting any interoperability issues that arise.
- To consider performance-related issues during design, implementation and testing and to be prepared to identify the areas of our design and implementation that are performance-critical and options for improving these.
- To utilise any features of .NET technology that contribute to a rapid development time, due to the restricted time available for development arising from the short duration of the project.
- To implement in Microsoft C# – a Java-style object-oriented programming language that exploits many .NET features.

Full details of the design and use of MS.NETGrid-OGSI are provided with the MS.NETGrid-OGSI software. The software and documentation may be downloaded from <http://www.epcc.ed.ac.uk/~ogsanet>.

3.1 High-level Architecture

Our design was inspired by features from both the Globus Toolkit 3 implementation of OGSI [9] and Virginia's OGSI.NET implementation of OGSI on .NET [6][8]. Our design is similar in spirit to both of these implementations, in that we solve the same core problem of building stateful Grid services from stateless Web services. A distinguishing feature of our design is in the use of an ASP.NET Web Application as a Grid service container, this Web Application running under the Microsoft Internet Information Server (IIS) Web server. Our container also mimics the ASP.NET service deployment model, a feature which could prove attractive to developers already experienced in the production of ASP.NET Web Services.

3.1.1 Converting Stateless Web Services into Stateful Grid Services

Consuming a traditional Web service usually entails a client sending a request – expressed in, for example, SOAP – to a network end-point somewhere. At the end-point, the message is interpreted by the Web

service hosting container, for example ASP.NET. The container extracts from the request information on the Web service to be used and the operation exposed by the Web service that is to be invoked. An object representing the Web service is created, the operation called with arguments from the request, the result of the operation is placed into a response – again, expressed in, for example, SOAP – and these results communicated to the client. The Web service object is then destroyed, or left to be garbage collected. Essentially a traditional Web Service can be said to be inherently *stateless*.

Since Grid services are *stateful* this necessitated that the service object, once created, be maintained so that the next call by the client is handled (conceptually) by the same object instance. Our solution is to store references to service objects and to map client requests to these Grid service objects. A C# object is created within a ASP.NET Web Application that acts as a repository of C# objects corresponding to Grid services. This is termed the *OGSI container*. A stateless Web service object – generated in response to a service request from a client – uses the identity of the Grid service to access to obtain a reference to a Grid service object stored within the OGSI container object. Two successive client requests to the same Grid service will therefore be handled by two different server-side Web service objects – which are created on a per-request basis in the standard Web services manner – but these Web service objects will forward the requests to the same Grid service object.

3.1.2 Service Lifetime

Our OGSI container allows for two types of service lifetime. The first, known as *transient* service lifetime, applies to Grid services that are spawned by other services, for example factory services. A transient Grid service lives until its termination time has passed, when it then will no longer respond to operation invocation requests. However, clients may request extensions to the service instance lifetime via a GridService portType operation, as described in [2].

The second type of lifetime is termed *persistent*. Persistent Grid services are initialised when the OGSI container is created and live as long as the ASP.NET Web Application which holds the OGSI container. Container-created (persistent) services are necessary to bootstrap essential services such as factories.

The distinction of services in terms of their lifetimes – persistent and transient – is consistent with the implementations of [9] and [6], although it is not required by [2].

3.1.3 Service Naming

Since the identity of a Grid service is used to identify the Grid service object which should handle a client request, one important design consideration is Grid service naming. Our solution is that for every persistent service instance, there must be one unique .asmx proxy file resident in ASP.NET. For example, say we have a factory for a service *CounterService*. The proxy for this factory could be located at <http://localhost/Ogsi.Container/services/persistent/CounterServiceFactory1.asmx> and *CounterServiceFactory2.asmx* would be proxies for two different persistent services. Persistent services are identified in the by the path to the .asmx file.

For every *class* of transient service, for example *CounterService*, there must be one unique .asmx proxy file For example, all *CounterServices* could be represented by a proxy located at <http://localhost/Ogsi.Container/services/transient/CounterService.asmx>. We differentiate between different each transient *CounterService* by the use of an `instanceId` parameter in the query string of the URL. So, for example, <http://localhost/Ogsi.Container/services/transient/CounterService.asmx?instanceId=counter1> and <http://localhost/Ogsi.Container/services/transient/CounterService.asmx?instanceId=counter2> represent the URL's of two transient *CounterServices*.

3.1.4 Interacting with Grid Services

Figure 1 shows how a client – implemented in the C# programming language – communicates with a Grid service under our architecture:

- A client application makes a C# method call to a local client-side proxy object which exposes the operations of the server-resident Grid service as methods (1).
- The proxy creates a SOAP request containing information on the location and identity of the Grid service to access, the operation to be called and the arguments passed by the client.

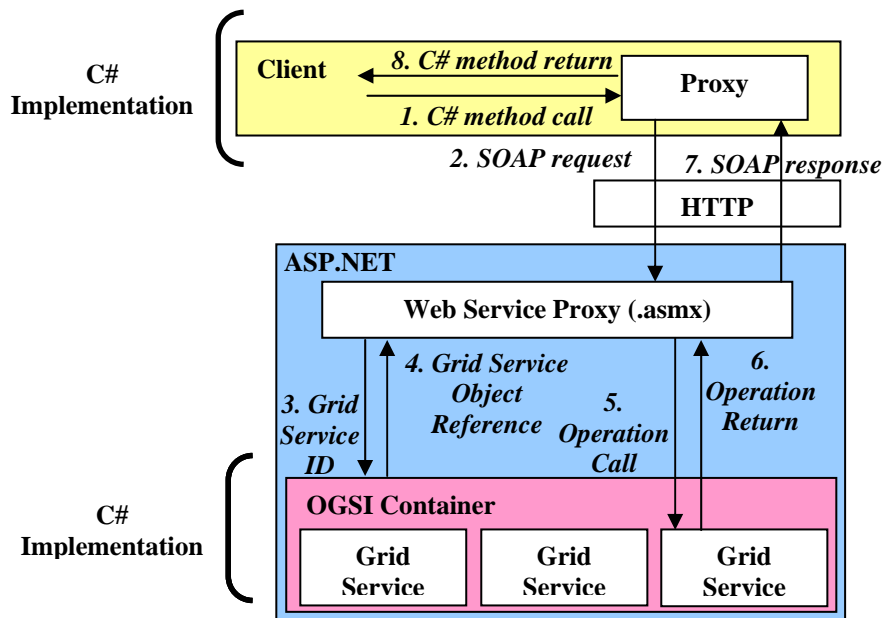


Figure 1: Interacting with Grid services in MS.NETGrid-OGSI

- The proxy transmits this SOAP request over HTTP to an ASP.NET Web service hosting environment running on some Web server (2).
- ASP.NET identifies the .asmx proxy file corresponding to the service the client wishes to access and compiles this into a proxy object.
- This proxy object then extracts from the SOAP request the unique identity of the Grid service to be invoked and passes this to the OGSi container object (3).
- The OGSi container returns a reference to the object corresponding to the desired Grid service (4).
- The proxy object then extracts the operation arguments from the SOAP request and invokes the operation requested by the client on the Grid service object (5)
- The results are then returned from the Grid service object to the proxy (6).
- The proxy bundles the results into a SOAP response and transmits these over HTTP back to the client (7).
- The client-side proxy unpacks the results from the SOAP response and returns these to the client application as a standard C# method return (8).

3.2 Developing and Deploying Grid Services

Under our architecture, developing a Grid service involves:

- Specifying C# classes which implement the functionality of the Grid service by:
 - Inheriting from pre-defined classes we provide which encapsulate functionality relating to the OGSi GridService portType and, for factory services, the OGSi Factory portType.
 - Specifying pre-defined delegate classes which we provide which encapsulate functionality relating to the OGSi GridService portType and, for factory services, the OGSi Factory portType.
 - Implementing service-specific classes providing application-specific portType functionality.
- Authoring a Web Service Proxy by creating an ASP.NET .asmx file.

- Extending the ASP.NET Web Application configuration file to specify the name of the Web Service Proxy file, and associated proxy class, the type of lifetime of the service, Grid service C# implementation classes and any application-specific parameters.

3.3 OGSINET – An alternative approach

The Grid Computing Group at the University of Virginia have also undertaken an implementation of OGSINET on .NET known as OGSINET [6][8]. The design of OGSINET differs from the MS.NETGrid container in a number of ways. The primary difference is that OGSINET is implemented as a stand-alone Windows service. An ISAPI filter attached to an IIS Web server intercepts requests to Grid services and redirects these requests to the stand-alone Windows service. A dispatcher within this Windows service routes the request to an AppDomain – a memory-isolated region within a process – which holds the requested Grid service. Method invocation then proceeds in a manner similar to MS.NETGrid-OGSINET.

Architecturally the two implementations are quite similar. They both follow the following procedures when dealing with Grid service requests:

- A service request is intercepted.
- A Grid service is selected on the basis of information within the service request.
- The requested Grid service operation is invoked via reflection on the Grid service.
- The results are returned to the client.

The difference between implementations lies in the level of control provided to the service developer. In OGSINET the level of control is substantially greater. It is possible to write custom message handlers for each service, allowing fine-grained security features or alternative message formats to be implemented, for example. Security is also provided by OGSINET. However, MS.NETGrid-OGSINET leverages existing functionality of ASP.NET to facilitate ease of use and provide a development model instantly familiar to ASP.NET developers.

3.4 Limitations

The limitations of MS.NETGrid-OGSINET include the following:

- Service data and certain operation messages do not meet the OGSINET specification exactly.
- ASP.NET can generate WSDL descriptions of our Grid services. However this is pure WSDL 1.1 and does not contain information relating to service data.
- Only functionality relating to the OGSINET GridService and Factory portTypes is currently provided.
- There is no support for security.
- There is no support for versioning or digital signing of software libraries.
- Service state is lost if the ASP.NET Web Application which hosts the OGSINET container is shut-down.
- Arbitrary service naming is not supported.

It is intended to address some of these limitations before the project concludes.

4 Grid Service Demonstrators

We have developed a small set of Grid service demonstrators to exercise our OGSINET on .NET implementation. These include:

- A simple counter service which allows clients to increment and decrement a counter.
- A prime factors service which calculates the prime factors of a number provided by a client.
- A mock-up of a load monitoring service factory service which creates services on specific hosts dependant upon their current load.
- An OGSA-DAI Grid Data Service which will now be described in more detail.

4.1 OGSA-DAI Grid Data Service

OGSA-DAI [4] (Open Grid Services Architecture – Data Access and Integration) is concerned with the provision of uniform service interfaces for data access and integration via the Grid. The aim is that through

OGSA-DAI interfaces, disparate, heterogeneous data sources and resources can be treated as a single logical resource.

Central to OGSA-DAI is the notion of a Grid Data Service (GDS) which manages access to some data resource – a database management system and a database – while hiding from the client details as to the implementation of the connection to the data resource. Our Grid Data Service demonstrator provides a stripped-down implementation of a GDS which runs on MS.NETGrid-OGSI. The GDS can be configured to connect to a Microsoft SQL Server database and a client can then issue SQL statements through the operations offered by the GridDataPerform portType which the GDS implements. The results are then returned to the client. A client can therefore query a database in an approach no different to that involved in interacting with any other Grid service.

5 Training the UK e-Science Community

Having developed an implementation of OGSI on Microsoft .NET and a small set of Grid service demonstrators that exercise this implementation we are now on the verge of hosting our first “OGSI and Microsoft .NET” training course. This will be held at the e-Science Institute within the National e-Science Centre in Edinburgh on September the 9th and 10th 2003. Our training course consists of one day of theory and one day of practical work, to provide delegates with hands-on experience of developing Grid services under Microsoft .NET. The first day is structured as follows:

- Grid Services:
 - XML and Web Services.
 - Grids.
 - OGSA.
 - OGSI.
 - OGSI Implementations.
- Essential Microsoft .NET Concepts – concepts relevant to implementing OGSI on Microsoft .NET:
 - A .NET Overview.
 - A C# Recap – an object-oriented language for .NET.
 - Application Configuration in .NET.
 - Assemblies – C# package files.

- Using Attributes ASP.NET – supporting reflection and meta-data.
- MS.NETGrid-OGSI:
 - The MS.NETGrid Project.
 - Why OGSI on .NET?
 - MS.NETGrid-OGSI Container Architecture.
 - Service Development in MS.NETGrid-OGSI.
 - A Contrasting Implementation – Virginia’s OGSI.NET.
- Obtaining, installing and testing MS.NETGrid-OGSI.

The second day is structured as follows:

- Tutorials in Developing Grid Services.
- Conclusion:
 - MS.NETGrid-OGSI Demonstrators.
 - Course Feedback.
 - Discussion.

It is through the delivery of this course – and revisions of the course planned for November 2003, January 2004 and February 2004 – that we hope to realise the prime goal of the project – providing a practical demonstration to the UK e-Science community of the applicability of Microsoft .NET technologies to the hosting, development and deployment of Grid Services.

6 Conclusion

In this paper we have described the work undertaken on the MS.NETGrid project since its commencement in March 2003. We described the design of MS.NETGrid-OGSI – an implementation of the Open Grid Services Infrastructure under Microsoft .NET. This was followed by an overview of Grid service demonstrators we have authored which run on this implementation. Finally, we presented the format of a training course for the UK e-Science community on the application of Microsoft .NET to OGSI and Grid applications. In the following months we look forward to reporting on continued work on extending and revising MS.NETGrid-OGSI, developing e-Science demonstrators that run on our implementation and, most importantly, reporting on our experiences in hosting our training workshops and the reception to these by the UK e-Science community.

Training Courses Information and Registration

Further information on our training courses and registration information is available on the NeSC WWW site:

<http://www.nesc.ac.uk>

References

- [1] I. Foster, C. Kesselman, J. Nick, S. Tuecke. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Draft 2.9)*. Open Grid Services Architecture WG, Global Grid Forum, June 22, 2002. See: <http://www.gridforum.org/ogsa-wg>.
- [2] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman. *Open Grid Services Infrastructure version 1.0 (Draft 33)*. Open Grid Services Infrastructure WG, Global Grid Forum, June 27th 2003. See <http://www.gridforum.org/ogsi-wg>.
- [3] Microsoft .NET. WWW site: <http://www.microsoft.com/net>.
- [4] OGSADAI Project. WWW site: <http://www.ogsadai.org.uk>.
- [5] Globus Project. WWW site: <http://www.globus.org>.
- [6] OGSINET Project, Grid Computing Group, University of Virginia. WWW site: <http://www.cs.virginia.edu/~humphrey/GCG/ogsi.net.html>.
- [7] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, **15** (3). 200-222. 2001. See: <http://www.gridforum.org/ogsa-wg>.
- [8] G. Wasson, N. Beekwilder and M. Humphrey. *OGSINET: An OGS-compliant Hosting Container for the .NET Framework*, Grid Computing Group, University of Virginia. May 17th 2003.
- [9] Globus Toolkit 3 + OGS. Globus Project. WWW site: <http://www.globus.org/ogsa>.
- [10] T. Banks (ed.) Open Grid Service Infrastructure Primer (Draft). Open Grid Services Infrastructure WG, Global Grid Forum, June 5th 2003. See: <http://www.ggf.org/ogsi-wg>.
- [11] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1*, World Wide Web Consortium, March 15th 2001. See: <http://www.w3.org/TR/wsdl>.
- [12] Apache Axis. WWW site: <http://ws.apache.org/axis/>.
- [13] N. Mitra (ed.) SOAP version 1.2 Part 0: Primer, World Wide Web Consortium, June 24th 2003. See <http://www.w3.org/TR/soap12-part0/>.