MyGridFTP: A Zero-Deployment GridFTP Client Using the .NET Framework

Arumugam Paventhan and Kenji Takeda

School of Engineering Sciences, University of Southampton,
Highfield, Southampton, SO17 1BJ, UK
{povs, ktakeda}@soton.ac.uk

Abstract. Large-scale scientific and engineering applications are increasingly being hosted as Grid services using Globus middleware complying to the Open Grid Services Architecture (OGSA) framework. In order for users to fully embrace Grid applications, seamless access to Grid services is required. In working towards this aim we present the design and implementation of Grid clients that utilise the language-independent Microsoft .NET Framework that can be deployed without software prerequisites (zero-deployment). We demonstrate runtime security authentication interoperability between Microsoft Windows-native SSPI and the Globus GSSAPI, with full proxy support. This is demonstrated with a .NET GridFTP client, called MyGridFTP. We believe that this is one of the first implementations to use Windows native security infrastructure to interoperate with the Grid Security Infrastructure in Globus. This paves the way for language-independent .NET clients to be written that are fully interoperable with Globus-based Grid services. This work is part of a larger experimental aerodynamics Wind Tunnel Grid project, which has significant requirements for data management from acquisition, collating, processing, analysis and visualisation.

1 Introduction

The data acquired in scientific and engineering experiments must be stored, processed, and visualized effectively in order for users to be able to analyze the results in a coherent fashion. The resources utilized in these steps are often distributed across the network, at both inter-organization and intra-organization level. Grid technology can help build a coordinated resource sharing environment [1] in this scenario.

Grid clients developed using client side Application Programming Interface (API) allow application users to consume Grid Services. [2] demonstrates how the Java commodity technology can be used to access Grid services and envisages Commodity Grid (CoG) Kit on other platforms. There are two possible approaches to developing Grid clients:*Client-side application and Browser or Web-based application.* The advantage of client side applications is a rich user experience and access to the local file system. The disadvantage is that the necessary software and runtime environment must be installed and configured before

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 374–383, 2005.

it can start on the client machine. Any version change at the server side could affect the client, often requiring installation of a new version of the client. The advantage of Web-based applications is that the up-to-date version can be invoked without having to worry about any installation steps on the part of user. The disadvantages are that it will have no access to the local file system - a problem, for instance, with client initiated file transfers and accessing a local X.509 certificate store for GSSAPI authentication - and it is not configurable to connect to an arbitrary *IP Address/Port number*. For example, GridFTP utilizes port 2811, supports certificate based authentication and may be running on a server other than the Web server front-ending the Grid service. Also, in the case of third party GridFTP transfers the client needs to make control channel connections to multiple servers.

The .NET framework allows another possible approach to the above problem when we assume Windows client [3]. Since the .NET runtime environment and SSPI are integral part of Windows, a .NET based Grid client can be downloaded over HTTP similar to HTML document and run as a client-side application (Fig.1). In this way, it does not require client-side installation or configuration steps combining the advantages of both the client-side application and web-based application. The first time download is cached on the client machine for subsequent invocations and can be configured to only download again when there is a version change. The technology is comparable to Java Web Start. The advantages of zero-deployment using .NET include the ability to leverage rich client capabilities such as highly-interactive user interface; a simple user interaction does not have to take a round-trip for response; access to local filesystem for data transfer services; and ability to make network connections to any server.

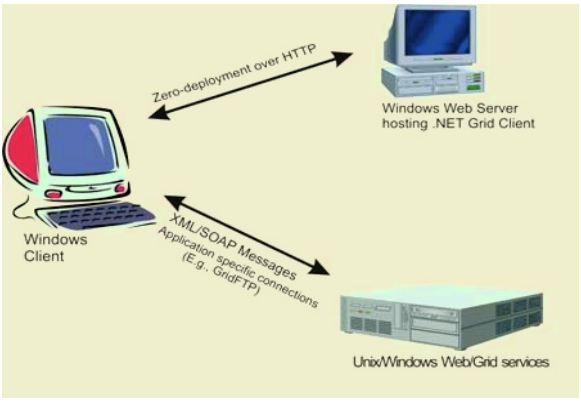


Fig. 1. Zero-deployment architecture

This work is part of a project to develop a Wind Tunnel Grid system aims at consolidating the workflow process for aerodynamics test engineers and scientists in test planning, data acquisition, processing, analysis, visualization, data management and data mining. The raw data from the Wind Tunnel experiments are

stored as flat files and its associated metadata are stored in Relational Database Management System (RDBMS). Authorized users are allowed to access the functionalities exposed as a set of Grid/Web services.

In the following sections we enlist the Wind Tunnel Grid requirements in general and .NET managed GridFTP client MyGridFTP in particular and describe its implementation details.

2 Wind Tunnel Experiments and Grid-Specific Requirements

The School of Engineering Sciences at the University of Southampton has a number of small and large wind tunnel facilities that are used for teaching, research and industrial applications, such as Formula One racing car, aircraft and high-performance yacht design.

A variety of measurement systems, such as particle image velocimetry (PIV), pressure transducers, microphones, video, digital photographs for flow visualisation and laser doppler (LDA) and hot-wire anemometry (HWA) are used. These require a disparate array of software, running on many standalone hardware systems. The data acquired using these systems varies in format, volume and processing requirements. The raw data is in a number of different flat file formats, typically proprietary binary formats and text files. Processing the data is performed using commercial software, mostly provided by the data acquisition hardware vendor, spreadsheets, user-written C and FORTRAN programs and Matlab scripts. Issues that arise from this arrangement include training, support, data compatibility, data access and analysis, and backup.

In order to improve the overall workflow efficiency a Wind Tunnel Grid system is being developed (Fig.2). This will allow experimental data to be managed,

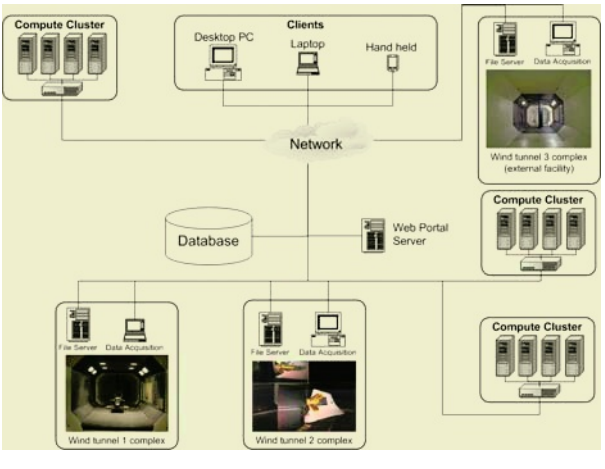


Fig. 2. Wind Tunnel Grid

stored and analysed in an RDBMS. For a given project users spread across multiple sites may perform complementary experiments, and use each other's compute resources for data processing and analysis. In order to provide seamless user access to the system, easy to use, lightweight client software is a major requirement. Typical users of the system include experienced aerodynamics engineers and scientists, undergraduate and postgraduate students, and visiting staff. Extensibility, so that additional data acquisition systems and analysis algorithms/software can be integrated into the system, is also required.

As a first step, the system should provide Globus GridFTP service to users for their data management. Data generated through wind tunnel experiments can be managed via the Wind Tunnel Grid portal from any .NET-enabled laptop or desktop on the network, without any client-side installations.

3 MyGridFTP Requirements

The major requirement of MyGridFTP is to provide GridFTP file transfer features including X.509 certificate based GSSAPI authentication. User should be able to invoke MyGridFTP from the Wind Tunnel Grid portal. This gives users the flexibility of transferring data files from any network location using their laptop or any desktop without having to install any software. MyGridFTP must also provide APIs for the development of custom clients supporting zero-deployment. Users should have options to configure GridFTP features - parallelism, striping, TCP Buffer size etc. From the project specific metadata available as part of the Wind Tunnel Grid system MyGridFTP can provide the user with an application specific automatic upload option. It means that the user does not have to select individual files for transfer, rather they simply need to select the folder and files are selected for transfer automatically. The graphical user interface must allow remote directory browsing of user's project space.

4 MyGridFTP Implementation

MyGridFTP is implemented using the Microsoft .NET runtime environment, as it enables application deployment from a Web location and rich client capabilites. It consists of GridFTP client APIs, a security module and graphical user interfaces. GridFTP client APIs include calls supporting features that are part of GridFTP extensions [4]. The security module uses SSPI and CryptoAPI[1], respectively to mutually authenticate with the GridFTP server and generate proxy certificates. MyGridFTP is hosted as part of the data transfer services in Wind Tunnel Grid Portal. The Wind Tunnel Grid Portal allows users to log into the system, create projects, sub-projects and test-cases, and upload raw data files from experiments. When the user clicks on the MyGridFTP link the .NET executable is automatically downloaded and run on the local machine supporting

[1] Microsoft Cryptographic Service Provider functions.

GSSAPI authentication, Globus Proxies and GridFTP data transfer features. The following sections elaborate the implementation with technical details.

4.1 Security

Grid security is crucial for authentication, authorization and delegation. The Generic Security Services Application Programming Interface (GSSAPI) [5] defines a portable API for client-server authentication. At the server side, Grid Security Infrastructure (GSI) is implemented using GSSAPI and Grid forum recommended extensions [6]. The GSI Message specification [7] defines three types of GSI messages exchanged between client and server as shown in Table 1. During Context-establishment, the MyGridFTP client uses SSPI to exchange SSLv3 handshake messages for mutual authentication with the GridFTP server. The client can delegate its credentials to server by sending a delegation flag. The server responds by sending a PKCS10 certificate request containing a valid public key and proxy certificate extension. A new proxy certificate with full/limited proxy credential is created, DER encoded and signed using CryptoAPI based on the Proxy Certificate Profile [8]. The Windows environment provides each user a personal *"MY"* store to manage user's X.509 certificates.

4.2 Runtime Environment

The .NET Framework consists of Common Language Runtime (CLR) and Framework Class Library (FCL). The CLR operates on assemblies which is a logical grouping of one or more managed modules or resource files. The *assembly* is defined as the smallest unit of reuse, versioning and security. Assemblies can consist of types implemented in different programming languages. The CLR is discussed and compared with the Java Virtual Machine (JVM) for multi-language support in [9]. The .NET development environment compiles high-level language code into Intermediate Language (IL). The CLR's JIT (just-in-time) compiler converts IL into CPU instructions at runtime. More details on architecture of .NET platform, about managed and unmanaged code can be found in [10].

MyGridFTP consists of three assemblies: the security module written in C^{++} for GSSAPI authentication and delegation of user credentials; GridFTP

Table 1. MyGridFTP Mutual authentication, Delegation and Message security

GSI Message Phase	MyGridFTP Client	SSLv3 Message Type	GridFTP Server
Context Establishment	AcquireCredentialHandle InitializeSecurityContext called until 'Finished' (implying successful authentication)	 ⇔Client-Server Hello ⇐ Server Certificate ⇐ Certificate Request ⇒ Client Certificate ⇒ Client Key Exchange ⇒ Certificate Verify ⇔ ChangeCipherSpec ⇔ Finished	gss_acquire_cred gss_accept_sec_context called until 'Finished' (implying successful authentication)
Delegation	⇒ Delegation Flag ⇐ PKCS10 certificate request ⇒ Proxy Certificate Chain	⇒ ApplicationData ⇐ ApplicationData ⇒ ApplicationData	gss_accept_delegation
Application-specific	EncryptMessage DecryptMessage	⇒ ApplicationData ⇐ ApplicationData	gss_unwrap gss_wrap

⇔ Both client and server exchange, ⇐ Server to client message, ⇒ Client to server message

client classes written in C#; and graphical user interfaces written C#. The security assembly uses interoperability services available in the .NET framework to invoke services available as part of SSPI and CryptoAPI. SSPI (Secur32.dll) and CryptoAPI (Crypt32.dll) are currently available as unmanaged implementations in Windows, but it is envisaged that these will be available as fully managed implementations in due course. Interoperability services available in the *System.Runtime.InteropServices* namespace in FCL expose mechanisms for managed code to call out to unmanaged functions contained in Dynamic Link Libraries (DLL). The user requires to make either the MyGridFTP assembly or the MyGridFTP download site as *trusted* by using .NET Framework configuration wizard, so that MyGridFTP client will have read/write access to the local file system.

4.3 Web Server Description

The Wind Tunnel Grid Portal consists of set of Active Server Pages (ASP.NET) web forms (HTML pages) and associated processing logic known as *code-behind files* written in C#. It is hosted using Internet Information Services (IIS) under Microsoft Windows Server 2003. The code-behind files instantiate the data access layer classes for accessing project metadata stored in the RDBMS (SQL server 2000). Some of the Wind Tunnel Grid metadata tables include *UserAccounts*, *Projects* and *TestCases*.

The application specific *Testcases* metadata vary depending on the Wind Tunnel experiment. The *Testcases* ASP.NET web page has the link for MyGridFTP, as shown in Fig.3. When the user clicks on the link the MyGridFTP executable is downloaded and runs on the client machine for GridFTP file transfer. Since the number and names of files to be transferred can be determined based on the *Testcases* metadata available, users can opt for an automatic data transfer option. In auto mode the files are transferred automatically; based on the direction of data transfer the user would select an upload folder or download folder. The user experience is seamless, in that they are not aware that a separate rich client application has been downloaded, installed and run.

4.4 GridFTP Server Configurations

The GridFTP server component part of Globus Toolkit 2.4 is configured on a Linux platform. It runs as an *inetd* network service on port 2811. Since the UserAccounts metadata table holds the home directory information, the system administrator has the flexibility of mapping multiple Wind Tunnel Grid users to the same unix login based on projects or user roles. The /etc/grid-security/grid-mapfile below shows a sample authorization entries.

"O=Grid/OU=GlobusTest/OU=simpleCA-cedc10.eng.soton.ac.uk/OU=eng.soton.ac.uk/CN=Paventhan" wtg

"O=Grid/OU=GlobusTest/OU=simpleCA-cedc10.eng.soton.ac.uk/OU=eng.soton.ac.uk/CN=Kenji Takeda" wtg

"O=Grid/OU=GlobusTest/OU=simpleCA-cedc10.eng.soton.ac.uk/OU=eng.soton.ac.uk/CN=C Williams" wtgadmin

In this example the first two users are authorized to login as 'wtg' (Wind Tunnel Grid User) and the last as 'wtgadmin' (Wind Tunnel Grid Administra-

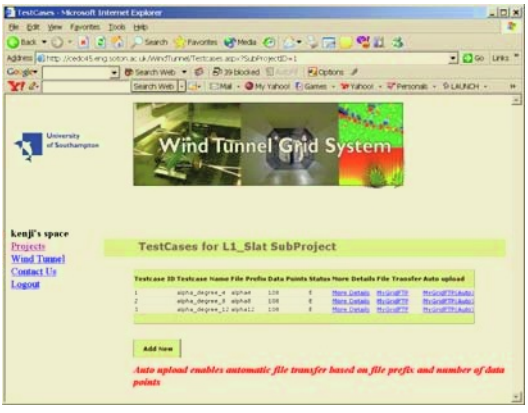


Fig. 3. Webpage hosting MyGridFTP

tor). UserAccounts table holds the actual home directory metadata, which could be, for example, a subdirectory in /home/wtg for the first two users - this gives them different project space. MyGridFTP makes an XML Web service request immediately after authentication and sets the home directory metadata as the FTP root directory during file transfer.

4.5 User Interfaces and Features

Fig.4. shows the user interface when invoked from the MyGridFTP(Auto) http link in Fig.3. Once downloaded MyGridFTP reads the user certificate store, authenticates with the GridFTP server, delegates user credential and performs an auto upload from selected folder. The auto upload feature is based on *Testcases* metadata (see section 4.3). In case of the manual option, the user needs to select individual files for transfer.

MyGridFTP can be configured for parallelism in extended block mode during Upload/Download. At the API level the MyGridFTP class (Table 2) supports GridFTP protocol for extended retrieve (ERET), extended store (ESTO) and striped data transfers (SPAS or SPOR). The ExtendedStore and ExtendedRetrieve allow partial copy (data reduction) of the file to be transferred. Partial

Table 2. MyGridFTP Class and its usage

Class: MyGridFTP **Functions:**	**Usage**
Authenticate(x509Cert.ThumbPrint) ParallelUpload(localFile, remoteFile, nDataPaths) ParallelDownload(remoteFile, localFile, nDataPaths) ExtendedStore(localFile, remoteFile, offset, length) ExtendedRetrieve(remoteFile, localFile, offset, length) Mode(modeString) Type(typeString) Upload(localFile, remoteFile) Download(remoteFile, localFile) List(listParams) ... and other basic FTP calls	*// read user's personal certificate store here* *mygftp = new MyGridFTP(ipaddress, port);* *mygftp.Delegation = true;* *mygrftp.DelegationType = LimitedProxy;* *mygftp.Authenticate(userX509Cert.ThumbPrint);* *mygftp.Mode(MyGridFTP.ExtendedMode);* *mygftp.ParallelUpload(localFile, remoteFile, 2)*

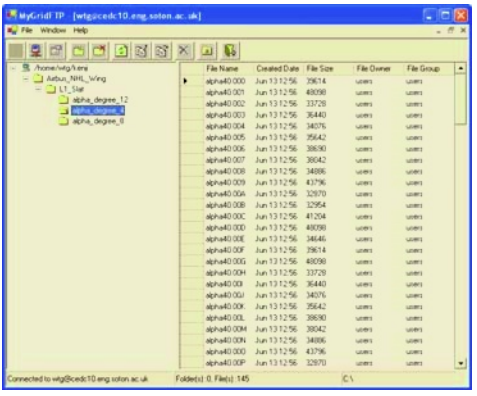
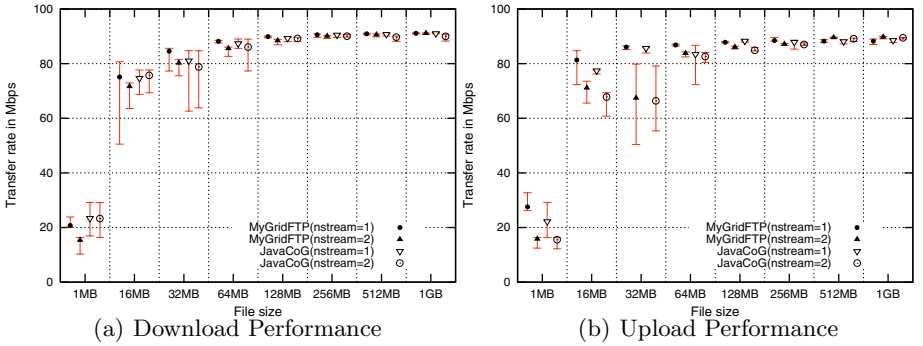


Fig. 4. MyGridFTP graphical user interface

data transfer will be more useful when we implement processing and visualization components of the Wind Tunnel Grid portal. Similarly striping and third party transfers are programmatically possible using the MyGridFTP class.

4.6 Performance

MyGridFTP download and upload performance are compared with Java CoG GridFTP client for various file sizes as shown in Fig.5. The performance test was measured within two separate programs written in C# and Java, respectively, using MyGridFTP APIs and Java CoG APIs. The GridFTP clients were run on a Intel Pentium-4 2.2 GHz Desktop running Windows XP and GridFTP server was configured on a Dual Intel Pentium-III 450 MHz, Linux system. The client and server are connected over a 100 Mbps Ethernet LAN. The time taken for Grid server authentication and authorization was measured separately over number of runs to compare Grid security implementations of MyGridFTP and



(a) Download Performance (b) Upload Performance

Fig. 5. MyGridFTP Performance

Java CoG as shown in Table 3. The X.509 user certificate for this experiment is of 1024 bit key length. The lesser authentication time of MyGridFTP could be attributed to its use of native runtime and security infrastructure. Each file was transferred between client and server a number of times and minimum, average and maximum bandwidths were recorded. As can be seen from the plot, large file sizes have very less bandwidth variability between different runs. Also, parallel streams does not improve the performance for smaller files as the test were run in a LAN environment. Both Java CoG and MyGridFTP measures a maximum download performance of 91 Mbps and upload performance of 89 Mbps for 1GB file size. As a comparison, iperf [11] bandwidth measurement tool gives 92.5 Mbps for 1 GB file input. By default, Windows XP operating system sets the TCP send and receive buffer size to 8K. This value is limiting, especially, for file transfers over wide area networks (WANs). The TCP buffer size could be increased in MyGridFTP programmatically to suit to high bandwidth-delay product subject to Operating Systems upper limits.

Table 3. GSSAPI Authentication

Authentication Time	Minimum	Maximum	Average
JavaCoG	1891ms	2563ms	2122ms
MyGridFTP	734ms	1156ms	848ms

5 Discussion

As the .NET Common Language Infrastructure (CLI) has been ratified as an ECMA standard (ISO/IEC 23271) there is interest in implementations on non-Windows platforms. For example, the Mono project [12] is an open source implementation of the .NET framework for use on Linux, Unix and Windows. Mono enables .NET managed code to run on multiple platforms similar to Java. The Mono.Security.Protocol.Tls namespace part of mono .NET project implements a 100% managed Transport Layer Security (TLSv1.0) and SSLv3. Incorporating GSSAPI authentication using this namespace will make MyGridFTP (and other clients requiring X.509 certificate based authentication) run on multiple platforms. Hence, the approach to Grid client deployment by means of hosting and executing on multiple platforms using the .NET framework will become realizable. Also, the Microsoft .NET Compact Framework targeting mobile devices will enhance the Grid application reach to Pocket PCs, PDAs and Smart Phones.

The Globus Reliable File Transfer (RFT) service could be hosted on Wind tunnel data acquisition system, data management server and compute cluster. This would allow client initiated asynchronous data transfer and notifications. If user is interested in uploading or downloading the data to their local system at any stage of the workflow, client/server style interactive data transfer is required.

The Wind Tunnel Grid portal could selectively implement XML Web services-based approach and GSI-security based approach for services such as GridFTP

for maximum compatability with other Grid resources. As future work, the MyGridFTP API can be extended to include GridFTP protocol improvements [13], resource management and information management making it a full-blown Globus client kit based on .NET.

6 Conclusions

A zero-deployment GridFTP client using Windows native runtime (.NET) and Security infrastructure (SSPI and CryptoAPI) is described. MyGridFTP launches from a web location and runs on the client machine without any software prerequisite, it can access the local file system and allows server-to-server communications via proxy certificates. We believe that this is one of the first Grid client implementations to use the Windows native security infrastructure (SSPI) to interoperate with the Grid Security Infrastructure (GSSAPI) in Globus, and as such it provides the basis for a language-independent .NET-based Commodity Grid (CoG) Kit.

References

1. Foster I, Kesselman C (eds.):The Grid: Blueprint for a Future Computing Infrastructure, Morgan-Kaufmann (1999)
2. Gregor von Laszewski et.al: A Java commodity grid kit, Concurrency and Computation: Practice and Experience, vol. 13 (2001) 643-662
3. Duncan Mackenzie:Introducing Client Application Deployment with *ClickOnce*, Microsoft Developer Network (2003)
4. W. Allock (ed.):GridFTP: Protocol Extensions to FTP for the Grid, Global Grid Forum Recommended Document (2003)
5. Linn J.:Generic Security Service Application Program Interface, Version 2, Update 1, RFC 2743, (2000)
6. Meder S., et al:GSS-API Extensions, Global Grid Forum Document (2002)
7. Welch Von (ed.):Grid Security Infrastructure Message Specification (2004)
8. Tuecke S., et.al:Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, IETF (2004) `http://www.ietf.org/rfc/rfc3820.txt`
9. Erik Meijer:Technical Overview of the Common Language Runtime, Microsoft Research, Technical Report (2001)
10. Jeffrey Richter:Applied Microsoft .NET Programming, Microsoft Press (2002)
11. http://dast.nlanr.net/Projects/Iperf/
12. http://www.go-mono.com
13. I.Mandrichenko (ed.):GridFTP v2 Protocol Description, Global Grid Forum Recommended Document (2004)