A File Transfer Component for Grids

Gregor von Laszewski,[1,*] Beulah Alunkal,[1,2] Jarek Gawor,[1]
Ravi Madhuri,[1] Pawel Plaszczak,[1] Xian-He Sun[2]
[1]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
[2]Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616
[*]Corresponding Author: gregor@mcs.anl.gov

to be published in:
The 2003 International Conference on Parallel and
Distributed Processing Techniques and Applications,
June 23-26, 2003, Las Vegas, Nevada, USA

Abstract

Grids have become a critical asset in large-scale scientific and engineering research. Accessing to distributed data is typically as important as access to distributed computational resources. In this paper, we present the design of Java CoG Kit components and APIs that make handling of data accessed through Grids easier for the novice Grid user and programmer. We introduce the design and architecture of a component for file transfers that separates issues related to requesting, performing and visualizing the actual file transfer. The design of our component is based on object and component based methodologies. Hence, it is assembled from a collection of reusable components promoting customization and reuse. We integrated the drag-and-drop paradigm. Protocol-independent interfaces allow easy adaptation to a variety of data sources. Hence, we not only have developed a GUI but sophisticated middleware to develop advanced Grid file transfer services. These components are distributed with the Java CoG Kit.
Keywords: FTP, GridFTP, RFT, Grid

1 Introduction

Grids [1] are a rapidly emerging form of distributed computing wherein the vision is an environment for coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [2]. To access such a complex environment, Grid computing environments and Grid portals must be provided to enable transparent and convenient access to the Grid [3].

A frequent obstacle to the creation of advanced applications using Grid environments is access to remote data. Many aspects must be considered while dealing with remote file transfer. Such aspects include which protocol is supported during the transfer, how the file transfers are organized, which mechanisms are used to instantiate a file transfer, and how the progress of file transferred can be observed. Additionally, we have to consider the different Grid user communities. These communities include common Grid users that are interested easy-to-use interfaces and Grid developers that are interested in easy-to-use APIs and shell commands. Hence, a file transfer strategy must be considering the Gestalt of the Grid that may look different for each user community [1].

In Grids, and on the Internet, files are stored on servers supporting a variety of protocols, such as FTP [4], GridFTP [5], and WebDAV [6] protocols. GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks that is based on FTP and developed as part of the Globus Project. WebDAV is a standard protocol for distributed authoring and versioning

that defines HTTP extensions necessary to enable distributed Web authoring while providing interoperability among a large number of commodity tools. Designing a single interface to connect to the various servers using different protocols would provide users with a uniform view of the files located on the different servers irrespective of their protocol implementations. Such an interface would enable the user to access a large number of file servers without being familiar with details about the protocol.

An additional concern in file transfers is how file transfer requests are performed. Commodity applications such as SecureFTP [7] and LeechFTP [8] provide easy-to-use graphical interfaces, thus making them powerful tools for the Web community. They provide drag-and-drop mechanisms for the instantiation of file transfers between clients and servers, provisions for viewing the status of the transfer, and book marking. Unfortunately, no such tools are available to the Grid community. Thus, the entry level into Grid file sharing remains unnecessarily high. The factors mentioned above motivate us to develop such high-level components and their supporting services, which must provide features such as multiple file downloads and uploads, recursive directory transfers, transmission management through queues, and quality-of-service guarantees. Besides file transfer we are also concerned with job submission and information browsing components. We have recently started the development of such components as part of a larger effort to produce an Open Grid Computing Environment(OGCE) that promotes easier use of Grids and is based on a multitude of Grid toolkits, including The *Globus Toolkit* version 2 and version 3. Such components may be accessed through based portals and GUI components. It includes basic Grid patterns enabling job submission, file transfer (a form of a job), and the management of jobs through forms, tables, and visual desktops.

In the rest of the paper, we present one of the OGCE components, called File Transfer Component (FTC), that allows protocol-independent data access as well as file browsing capabilities for Grids. We also present a convenient Java client library that allows developing such components in modular fashion by the community. We first describe the requirements and our proposed solution. Next, we describe the design and implementation of FTC. We conclude the paper with use cases, extensions, and future developments.

2 Requirements

The requirements for the design of our architecture follow the usual software engineering principles, such as (a) *uniform access mechanisms* to support wide variety of data sources that may include common stream-based data services such as FTP or HTTP; (b) *transparent and easy access* to data in Grids by using both, a GUI and a command shell; (c) *expandability* by providing mechanisms to integrate new protocols; (d) *reusability* and *composability* within Integrated Development Environments; and (e) *portability* so as to minimize development cost through an appropriate choice of technologies.

Although our requirements are language and framework neutral, we have implemented the component in Java, which allows us to fulfill the requirements of easy porting and deployment as discussed in [9].

3 Architecture of the File Transfer Component

Our architecture is based on a number of reusable components. Figure 1 shows the architecture of the file transfer component (FTC) that separates access, transfer, and display of data related operations cleanly. Hence, it comprises three main components: the access manager, the transfer manager, and the graphical user interface.

- The *access manager* provides uniform access to all the file transfer service providers, independent of the protocol they support, by interfacing to them

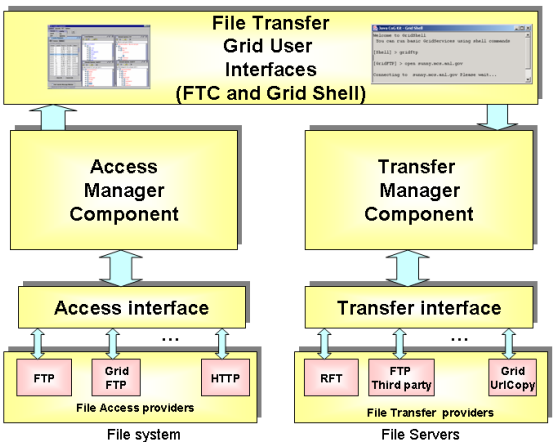

Figure 1: Architecture of the file transfer component.

through a single access interface. The providers we currently support implement FTP and GridFTP protocols. The user initiates the file transfer requests through the access manager.

- The *transfer manager* manages specific data transfers. It performs the actual movement of files from one location to the other. It takes the transfer requests specified through the graphical user interface and schedules the transfers to the servers involved. The data movement can be provided by reusing services such as Grid `UrlCopy` or any FTP server providing third-party transfers. For instance, users who need reliable file transfers between the GridFTP servers can use the Reliable File Transfer server (RFT) to initiate reliable transfers with quality-of-service provisions.

- The *user interface* supports uniform display of files and directories, while at the same time allowing drag-and-drop features for a variety of data sources. This component interacts with both the access manager and transfer manager. It communicates with the access manager to provide graphical representation to the files located in the local as well as remote systems. It gets the file transfer requests from the access manager, queues the requests, and sends them to a transfer manager that performs the actual file transfers.

We integrated these reusable components into a single component that we call the File Transfer Component (FTC) for Grids. We describe each of the other components in detail in the following subsections.

3.1 File Access Providers

As indicated previously, FTC supports multiple protocols. This feature is enabled through the provision of so-called file access providers. We currently support basic providers for FTP and GridFTP protocols. The file access providers allow us to access file system based on API calls that are implemented by using appropriate protocols. These providers are integrated into FTC by using a single interface called `AccessInterface`, which is described in Section 3.2.

The GridFTP [5] provider is of special interest to the Grid community. It is an extension to the FTP protocol and enables ubiquitous, high-performance access to data. GridFTP can perform a concurrent transmission of a file by using a number of parallel streams. This approach increases the efficiency of data transfers. This strategy is also commonly used in other commodity file transfer components between client and servers. However, we have the ability to perform such function between servers which commodity file transfer services do not typically support.

3.2 File Access Interface

The FTC component contains an access interface that provides uniform access to sources supporting multiple file transfer protocols. It defines basic operations such as connecting to a remote server, changing to a directory, listing the files present in a directory, making a new directory and deleting files and directories. The pseudocode for the interface is shown in Figure 2.

```
interface AccessInterface {
   public void connectRemote();
   public void chdir(String dirname);
   public void list();
   public void mkdir(String dirname);
   public void rmdir(String dirname);
   public void rmfile(String filename);
   public void rename(String oldname,
                      String newname);
   public void disconnectRemote();
   . . .
}
```

Figure 2: Pseudocode for the access interface of the FTC.

The purpose of these methods is straightforward. The `connectRemote()` method is used initially to setup the connection with a remote server. The `chdir()` changes the location of the directory. The `list()` method is used for retrieving all the subdirectories and files present in a directory. The other methods provide basic file system operations. The file access providers need to implement just this interface in order to be integrated in FTC.

3.3 File Transfer Providers

In contrast to the file access providers that enable access to the actual file system, the file transfer providers are responsible for transferring the files. File transfer requests, generated by the file access providers, contain the information required for performing file transfers. File transfer providers take these requests and perform the actual transfer. In order to initiate a third-party transfer between two different remote data sources in the Grid, we also require the third-party support from the providers. We use file transfer providers such as the Java CoG Kits [10] `UrlCopy`, which internally supports the protocols http, https, ftp and gsiftp; and the Globus Project's Reliable file Transfer service (RFT)[11].

Internally RFT uses the GridFTP client libraries provided by the Java CoG Kit. It allows the transfer of files with quality-of-service provisions. Thus, if a file transfer is interrupted at one point, the reliable file transfer service will assist in restarting the transfer at another time and will try to complete the interrupted transfer. Problems such as dropped connections, machine reboots and temporary network outages are dealt with automatically.

Currently, we use an alpha version of RFT distributed with the Globus Toolkit version 3. It provides simple access to controlling and monitoring third-party file transfers between GridFTP servers. The client controlling the transfer is hosted inside a Grid service [12] so it can be managed by using the soft-state model and queried by using generic interfaces available to other Grid services. However, as obvious GridFTP is just one way of performing file transfers in Grids.

3.4 File Transfer Interface

The File Transfer Interface provides uniform access to multiple file transfer providers. The pseudocode for the interface is shown in Figure 3.

```
interface TransferInterface {
   public setFromUrl(String url);
   public setToUrl(String url);
   public startTransfer();
   public suspendTransfer();
   public resumeTransfer();
   public cancelTransfer();
}
```

Figure 3: Pseudocode for the transfer interface of the FTC.

The functionality of the methods of the interface are straight forward. The `setFromUrl()` provides the information regarding the details about the file that needs to be transferred. The `setToUrl()` provides information regarding the location where the file needs to be transferred. The string url, which is taken as the parameter, follows the usual syntax [4]:

```
[protocol]://[user]@[host]:[port]:[file]
```

The other methods are needed to monitor the execution of file transfers. A file transfer provider needs to implement just this interface

in order to integrate with FTC. As obvious the use of our interface will also protect other middleware developers from frequent changes in design and implementation that took place during the development of other Grid transfer services such as RFT.

3.5 Graphical User Component

The front end of the file transfer component can be implemented in a variety of frameworks. Currently, we have implemented it based on the Java Swing framework. Nevertheless, we are implementing the component also as a Jetspeed portlet to allow the integration into portals, which can be invoked using web browsers. However, we believe that a more powerful component that supports drag-and-drop is needed for the easy use of Grids. The current generation of Web browsers simply does not support such a modality.

Our prototype component consists of file browsing and file transfer monitoring components that are designed based on the Java Beans [13] component model. Our goal is to be able to import these components in IDEs, such as JBuilder or Eclipse, to provide a convenient development environment for Grids. The file browsing panel offers elaborate directory browsing and directory manipulation functionality, which works identically for both local and remote hosts. The user is able to (a) view the directory entries in tree structure, (b) transfer files with few clicks of the mouse, (c) create new directories, (d) copy files and directories, (e) delete files and directories, (f) rename files and directories, and (g) refresh entries.

The monitoring panel supports (a) queuing of jobs, (b) checking status, (c) setting of transfer options. The graphical user interface reusing these panels within our integrated file transfer component is shown in Figure 4.

3.6 Important Features of FTC

We point out a number of important features of the FTC.

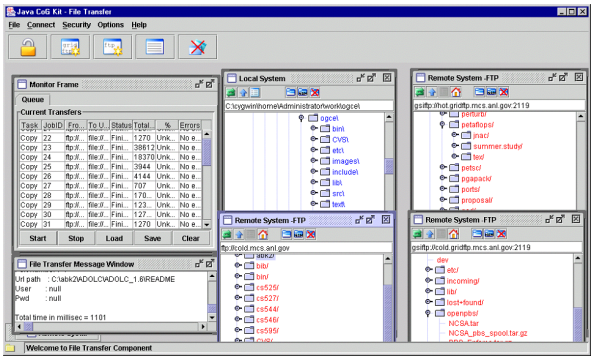

Figure 4: The GUI of the file transfer component.

Authentication. GridFTP uses Grid Security Infrastructure (GSI) [14] for authentication. GSI provides a number of useful services for Grids, including mutual authentication and single sign-on, and it protects critical and sensitive data by providing strong data encryption.

The authentication in FTC is performed through the visual-proxy-init program distributed with the Java Cog Kit. Hence, authentication is enables access to GSI enabled services such as GridFTP and RFT. FTC provides the ability to start a proxy, check the proxy information, or destroy the proxy.

Connecting to Different Transfer Servers. Users have the flexibility to login to any number of FTP or GridFTP servers. Files will be displayed in the tree structure in separate internal frames. The user can perform basic file system operations on any of the files, as well as transfer files and directories by dragging and dropping them between the frames.

Using RFT. RFT can be used for achieving reliable file transfers between GridFTP servers. The settings of the RFT server such as the name and the port have to be registered and set in the FTC. File transfer request will appear in a queue and they can be submitted to the RFT server by clicking on the Submit button. The status queue enables the user to explicitly

control the file transfer at the RFT server.

Java Client API to GridFTP. The Java client API has most recently been developed as part of the Java CoG Kit to support interface to FTP and GridFTP protocols. It implements the following features: file storage and retrieval to and from FTP server (client-server transfer), third-party transfer, ASCII and IMAGE data types, file data structure, nonprint format control, stream transmission mode, operation in passive and active server mode, parallel transfers, striped transfers, restart markers, and performance markers. More information about this API can be found on the Java CoG Kit Web pages [15].

Deployment. We are conveniently distributing the component through the Java Web Start technology [16]. This technology can be used when the user has the ability to download and install files in his user space on the client computer. In essence it is not different than installing a browser plug-in such as Acrobat Reader, a Microsoft program, or an MP3 player. Automatic updates provide an additional benefit while simplifying the maintenance and deployment of up-to-date clients. More information about the use of Web Start within Grids can be found in [9].

4 Extensions

Application developers can benefit from the fact that FTC is based on an object oriented design that uses extensively the interface concept. Hence, we it will be easily possible to enhance our component to access other data services with different protocols such as WebDAV. The task of integration is simplified, as the developer only has to implement a single interface to incorporate into FTC.

Since components in FTC are designed as Java Beans, it will be easy to enhance and integrate them into visual development environment such as JBuilder. For instance, we can develop applications such as file explorer using file browsing bean and an editor bean, which displays the contents of a file.

5 Conclusion

We have developed a user-friendly environment for interactive and transparent file access to Grid users. Such an interactive model is required for assisting scientists in their quest of accessing the Grid in a way that hides much of its complexity and involves a large degree of interaction with the system during the experimentation. Our component design effectively supports the integration of a new protocols that are build to support file transfers. As we use object oriented and component based design methodologies, we can add or replace already components that provide additional functionality or that are more sophisticated ones in the future. We have also demonstrated that our component does work with either version of the Globus Toolkit. We support version 2 and 3. This is a important as a higher level of abstraction that is provided by the Java CoG Kit proofs that this abstraction level is convenient for high level Grid application developer. The file transfer component and its APIs are distributed as part of the Java CoG Kit. However, the Globus Toolkit version 3 will not contain the graphical user interface. Users of GT3 still have to download and install the Java CoG Kit as the GT3 does not contain graphical components such as the once developed to build OGCE by the CoG Kit Project.

Acknowledgments

This work is supported in part by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advance Scientific Computing Research, U.S. Department of Energy under Contract W-31-109-Eng-38. Globus Toolkit research and development have been supported by DARPA, DOE, and NSF.

This work would not been possible without the help of the Globus Project team. We

thank all the members of the Globus Project for their valuable help. Globus Toolkit and Globus Project are trademarks held by the University of Chicago.

References

[1] G. von Laszewski, G. Pieper, and P. Wagstrom, "Gestalt of the Grid," in *Performance Evaluation and Characterization of Parallel and Distributed Computing Tools*, ser. Wiley Book Series on Parallel and Distributed Computing, to be published 2003, http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--gestalt.pdf.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, 2001. [Online]. Available: http://www.globus.org/research/papers/anatomy.pdf

[3] G. von Laszewski, I. Foster, J. Gawor, P. Lane, N. Rehn, and M. Russell, "Designing Grid-based Problem Solving Environments and Portals," in *34th Hawaiian International Conference on System Science*, Maui, Hawaii, 3-6 Jan. 2001. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cog-pse-final.pdf

[4] J. Postel and J. Reynolds, "File Transfer Protocol," RFC. [Online]. Available: http://www.w3.org/Protocols/rfc959/Overview.html

[5] W. Allcock, J. Bester, J. Bresnahan, S. Meder, and S. Tuecke, "GridFTP: Protocol Extensions to FTP for the Grid," Word Document. [Online]. Available: http://www.ggf.org/meetings/ggf6/ggf6_wg_papers/GridFTP.doc

[6] Y. Goland, E. Whitehead, U. Irvine, A. Faizi, S. Carter, and D. Jensen, "HTTP Extensions for Distributed Authoring – WebDAV," RFC, Feb. 1999. [Online]. Available: http://asg.web.cmu.edu/rfc/rfc2518.html

[7] "SecureFTP," Web Page. [Online]. Available: http://www.glub.com/products/secureftp/

[8] J. Debis, "LeechFTP," http://www.iweb.net.au/Downloads/leech.html.

[9] G. von Laszewski, E. Blau, M. Bletzinger, J. Gawor, P. Lane, S. Martin, and M. Russell, "Software, Component, and Service Deployment in Computational Grids," in *IFIP/ACM Working Conference on Component Deployment*, ser. Lecture Notes in Computer Science, J. Bishop, Ed., vol. 2370. Berlin, Germany: Springer, 20-21 June 2002, pp. 244–256. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/papers/vonLaszewski--deploy32.pdf

[10] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf

[11] B. Allcock and R. Madduri, "Reliable File Transfer Service," Web Page. [Online]. Available: http://www.mcs.anl.gov/~madduri/RFT.html

[12] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003, ch. The Physiology of the Grid, pp. 217–249. [Online]. Available: http://www.globus.org/ogsa

[13] "Java Beans," Web Page. [Online]. Available: http://java.sun.com/products/javabeans/

[14] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, "A National-Scale Authentication Infrastructure," *IEEE Computer*, vol. 33, no. 12, pp. 60–66, 2000.

[15] "Java CoG Kit," Web Page. [Online]. Available: http://www.globus.org/cog/

[16] "The Java CoG Kit Web Start Components," Demonstration. [Online]. Available: http://www.globus.org/cog/demo