Web Service Robust GridFTP

Sang Lim, Geoffrey Fox, Shrideep Pallickara and Marlon Pierce

Community Grid Labs, Indiana University
501 N. Morton St. Suite 224
Bloomington, IN 47404

{sblim, gcf, spallick, marpierc}@indiana.edu

Abstract— In this paper, we discuss reliable and secure file transfer middleware called NaradaBrokering. It is our goal to show that reliability features can be decoupled from the implementation of the service and protocol, and instead placed into the messaging substrate. This will allow us to provide file transfer quality of service comparable to GridFTP in other file transfer tools (such as normal FTP, SCP, HTTP uploads, and similar mechanisms).

Keywords: NaradaBrokering, Robust GridFTP, Reliable File Transfer, Web Service

1.0 Introduction

Reliable, secure high performance file transfer is one of the most important services offered by Grid environments. GridFTP [1] [6] is the one of the most common data transfer services for the Grid and is a key feature of Data Grids[10].. This protocol provides secure, efficient data movement in Grid environments by extending the standard FTP protocol. In addition to the standard FTP features, the GridFTP protocol supports various features offered by the Grid storage systems currently in use. This protocol includes the following features –

- Grid Security Infrastructures (GSI) and Kerberos support
- Support for reliable and restartable data transfer: By using this feature, GridFTP can restart transfers from point of failure when any failures occurred on network and/or server.
- Partial file transfer: It enables transfers of regions of a file.
- Parallel data transfer: It uses multiple TCP streams between two network endpoints to improve bandwidth.
- Third-party control of data transfer: It provides the ability to control transfers between storage servers from remote (third-party) server. This feature is especially handy for large distributed communities with large data sets.

However, even though GridFTP has good features of file recovery technologies, it has a restriction that the client needs to remain active at all the times until the transfer finishes. This in turn implies that we can not use the rich set of recovery features of GridFTP when the client state has been lost. In the event of client state loss, transfer has to restart from scratch. More fundamentally, GridFTP's many interested features are tied to its protocol and implementation. Providing these features to other file transfer services (such as those based on Web Services, for instance) requires reimplementation and re-engineering. These shortcomings may be addressed by inserting a reliable, high performance *messaging substrate* between the client and service. This addresses specific problems in GridFTP client lifetimes, but more generally will allow us to extend GridFTP-like features (listed above) to other services without extensive reimplementation.

In this paper we present our work which has addressed the client-active-at-all-times constraint. To achieve this we have made use of capabilities provided by the middleware, NaradaBrokering, developed at the Community Grids Lab at Indiana University. The remainder of this paper is organized as follows. In section 2 we present an overview of related work. In section 3 we present a brief overview of the NaradaBrokering system and the two services within NaradaBrokering that are most relevant to this work. In section 4 we provide details regarding our work. In section 5 we will compare NaradaBrokering with the reliable file transfer mechanism from Globus. Finally in section 6 we present our conclusions and future work.

2.0 Related Work

We are using many different file transfer mechanisms on daily bases. One of the most commonly used file transfer mechanism is File Transfer Protocol (FTP) [5]. This is the simplest way to exchange files between computers. FTP is an application protocol that uses the TCP/IP protocols. A more secure replacement for the common FTP, protocol is Secure Copy (SCP), which uses the Secure Shell (SSH) as the lower-level communication protocol. From the popularities of World Wide Web, we are also commonly using Hypertext Transfer Protocol (HTTP) as mechanism for transferring files. Even though some of file transfer mechanisms are quite reliable, these mechanisms do not provide guaranteed, reliable file transfer features like automatic recovery from failures.

Issues about reliable file transfer mechanism are more actively discussed and developed from the Grid community recently. One of the developments of reliable file transfer mechanism is included on the Globus project [2]. Globus provide a service that performs reliable file transfer by using the Reliable File Transfer (RFT) [3] [4] service. This service is developed with automatic failure recovery similar to NaradaBrokering. We will discuss more detail information about RFT in section 5.

3.0 NaradaBrokering and related services

NaradaBrokering [7] [8] is messaging middleware designed to run on a large network of cooperating broker nodes (we avoid the use of the term *server*s to distinguish it clearly from the application servers that would be among the sources/sinks to messages processed within the system). Communication within NaradaBrokering is asynchronous and the system can support large client configurations publishing messages at a very high rate. The system places no restrictions on the number, rate and size of messages issued by clients. NaradaBrokering imposes a cluster-based structure on the broker network. Clusters comprise strongly connected brokers with multiple links to brokers in other clusters, ensuring alternate communication routes during failures. This distributed cluster architecture allows NaradaBrokering to support large heterogeneous client configurations that scale to a very large size. NaradaBrokering provides support for a wide variety of event driven interactions – from P2P interactions to audio-video conferencing applications.

In NaradaBrokering entities can also specify constraints on the Quality-of-Service (QoS) related to the delivery of messages. Among these services is the reliable delivery service, which facilitates delivery of events to interested entities in the presence of node and link failures. Furthermore, entities are able to retrieve any events that were issued during an entity's absence (either due to failures or an intentional disconnect). The scheme can also ensure guaranteed exactly-once ordered delivery.

Another service, relevant to this paper, is NaradaBrokering's Fragmentation/Coalescing service. This service splits large files into manageable fragments and proceeds to publish individual fragments. Upon receipt at a consuming entity these fragments are stored into a temporary area. Once it has been determined (by the coalescing service) that all the fragments for a certain file have received these fragments are coalesced into one large file and a notification is issued to the consuming entity regarding the successful receipt of the large file.

The fragmentation/reliable delivery service combination can be used to facilitate transfer of large files reliably. Access to these capabilities is available to entities through the use of QoS constraints that can be specified. This facilitates exploiting these capabilities with systems such as GridFTP.

We emphasize here that NaradaBrokering software is a message routing system which provides QoS capabilities to any messages it sends. The NaradaBrokering system may be the messaging middle layer between many different applications, such as Audio/Video [11]. The QoS features provided by the NaradaBrokering system are independent of the implementation details of the endpoint applications that use it for messaging. Thus applications do not need to implement (for example) reliable messaging. They just use NaradaBrokering for communication and acquire reliability through NaradaBrokering.

Furthermore, NaradaBrokering provides capabilities for communicating through a wide variety of firewalls and authenticating proxies while supporting different authenticating-challenge-response schemes such as Basic, Digest and NTLM (a proprietary Microsoft authenticating scheme).

Support for the Web Service Reliable Messaging Framework (WS-RM) is currently being incorporated into NaradaBrokering. NaradaBrokering is quite resilient to failures since it is based on a distributed broker network and can sustain losses of one or more broker nodes.

4.0 Enhancing GridFTP

Here we describe how enhanced GridFTP uses features available in the NaradaBrokering reliable substrate. As we mentioned earlier GridFTP already incorporates a number of reliability features. However, all of those features are only valid for the GridFTP application and also only when the client is active at all times. It is our goal to show that these reliability features can be decoupled from the implementation of the service and protocol, and instead placed into the messaging substrate. This will allow us to provide file transfer quality of service comparable to GridFTP in other file transfer tools (such as normal FTP, SCP, HTTP uploads, and similar mechanisms).

Figure 1 is present the basic architecture of integration between GridFTP and NaradaBrokering. We can have two possible implementation approaches, proxy approach and router approach, for integration between GridFTP and NaradaBrokering. Even though the proxy approach is the more preferred method, this approach is harder to implement. We will discuss more detailed implementation for both approaches. Both implementations will use same security and any other underlying mechanisms that GridFTP provides.

The proxy approach is the preferred method since the GridFTP client code and user application do not have to change. All existing GridFTP code and user application can be used in our architecture without

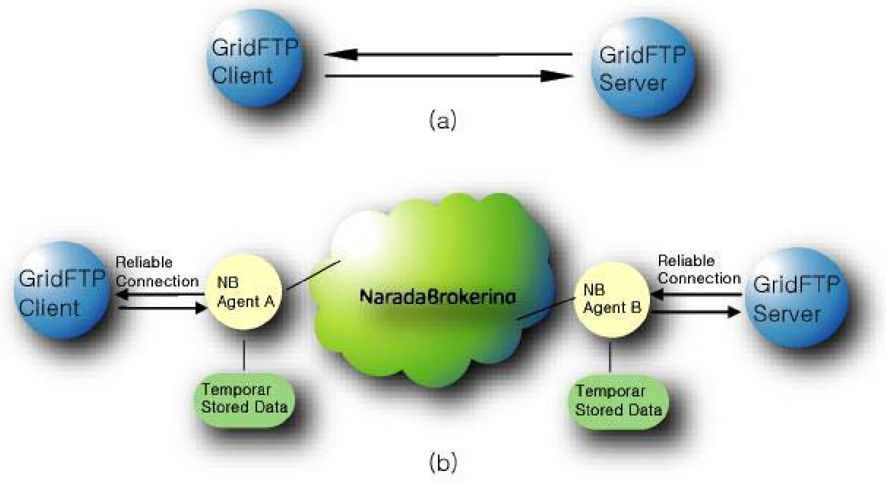

Figure 1 (a) Traditional GridFTP. (b) GridFTP with NaradaBrokering.

any changes once this method is implemented. Key to this approach is the remote GridFTP server is simulated by the NaradaBrokering *Agent A*. GridFTP client simply contacts the NaradaBrokering *Agent A* and uses it as if it is a regular GridFTP. NaradaBrokering *Agent A* then forwards all requests through the broker cloud to an available GridFTP server. Disadvantage of this approach is it is harder to implement and time consuming process since we have to create GridFTP server from the scratch.

Our eventual goal is to create a proxy GridFTP server on NaradaBrokering *Agent A*, but for initial testing we chose the "router" approach as simpler. This approach uses NaradaBrokering *Agent A* as simple router to transfer requests to the remote server. The advantage of this approach is it is easier than the proxy approach to implement since it uses NaradaBrokering *Agent A* as simple router. On the other hand we have to change the user application, even though change is minor (client code have to use *NBGridFTPClient* class instead of *GridFTPClient* class). And the router approach also requires some minor extensions to FTP/GridFTP client codes so that the client can tell NaradaBrokering *Agent A* that it want to use the GridFTP server. The client and server communicate solely with the agents on the edge of the broker cloud. For the GridFTP client stand point of view NaradaBrokering *Agent A* is a server and NaradaBrokering *Agent B* is a client for GridFTP server point of view.

The most characteristic change to GridFTP client code is we disabled automatic hostname checking of messages by the client and some extensions to allow hostnames and ports to be changed by the agent to match the remote server. This action was necessary because we are most likely will run NaradaBrokering *Agent A* and GridFTP server on different machines. By default GridFTP client will check for the match between GridFTP server hostname and a server hostname that actually GridFTP client is connected (in our case it is NaradaBrokering *Agent A*). Since we are using NaradaBrokering agents as simple router, all the security checks will done on the GridFTP server side. This means that all the Grid credentials will simply forwarded to GridFTP server. Then the GridFTP server will process requests and return results back to the client.

GridFTP requires two socket connections: a control channel that transfers commands ("put", "get") and a data channel that is used for data transfer. Currently, we have completed development of the uploading functionality of GridFTP with NaradBrokering using simple router approach. Connection between the

GridFTP client and NaradaBrokering *Agent A;* and NaradaBrokering *Agent B* and GridFTP server are connected with a high-speed, reliable, possibly local, connection. This connection is needed because if connection between Grid FTP client and the NaradaBrokering *Agent A* is lost, we cannot recover from this failure. Recovering from this failure is out of scope (GridFTP designed in this way). All the data will be first transferred and stored into the temporary local space of NaradaBrokering *Agent A*. This temporary data will be used when any failure is occurred inside of NaradaBrokering. Once all the data is stored locally in the NaradaBrokering *Agent A*, even if connection between GridFTP client and NaradaBrokering *Agent A* is lost, transferring to the server is guaranteed by NaradaBrokering. This feature is not on the current GridFTP system. In the current GridFTP system, if a client fails, the client has to begin uploading again from the start. NB Agent B also store data into the temporary local space. This temporary data will be used when any failure is occurred to the GridFTP server.

5.0 Comparison with Reliable File Transfer

Similar to NaradaBrokering, the Globus provide a service that performs reliable file transferring by the RFT service. In this section we will discuss more detail about the RFT and how it is differ from our NaradaBrokering.

RFT is developed with automatic failure recovery while overcoming the limitation of its predecessor technology, GridFTP. The RFT service can automatically deal with problems like dropped connections and temporary network outage by performing a retry until the problem is resolved or else an unrecoverable failure condition is met. Since the RFT service is built on the top of existing GridFTP client libraries, most of the automatic recovery features like restart support and remote problems of the RFT service are inherited by its predecessor. The RFT service, also, does not lose performance of GridFTP.

As we mentioned earlier, recovery feature of GridFTP has a strict restriction that the client needs to remain active at all the times until the transfer finishes. The RFT service resolves this restriction of GridFTP by introducing a non-user-based service. When a user submits a request for transferring, the transfer state is stored in a persistent manner. This state will be used to recover transfer from the last marker recorded for that transfer when failure occurs including the client state failure.

The RFT service itself has significant features to make reliable data transfer. However, the RFT service is not portable to any other systems. Once again our main goal of decoupling reliable features from the implementation is to make a portable system that can be deployed into any file transfer mechanisms and make that mechanism reliable by using NaradaBrokering as a middleware.

6.0 Conclusions and Future Work

We discussed reliable transfer mechanism in NaradaBrokering using GridFTP as an example. In this paper we discussed deploying NaradaBrokering in GridFTP. However, NaradaBrokering can be used as a reliable middleware of any other file transfer tools (such as normal FTP, SCP, HTTP uploads, and similar mechanisms). It is our goal to show that these reliability features can be decoupled from the implementation of the service and protocol, and instead placed into the messaging substrate.

For future work, we intend to do performance tests and exploit NaradaBrokering's distributed broker network capabilities to implement support for multiple underlying parallel socket streams. The brokering system is also by design a many-to-many messaging system, so we may exploit this to support simultaneous delivery of files to multiple endpoints. Finally, we will develop more examples of using other file transfer mechanisms that will mimic RTF-like features without reimplementation.

7.0 Reference

[1] GridFTP: Universal Data Transfer for the Grid http://www.globus.org/datagrid/gridftp.html

[2] The Globus Project http://www.globus.org/

[3] Ravi K Madduri, *Reliable File Transfer in Grid Environments*, Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02), 2002.

[4] Reliable File Transfer Service http://www-unix.mcs.anl.gov/~madduri/RFT.html

[5] RFC 765 – File Transfer Protocol specification http://www.faqs.org/rfcs/rfc765.html

[6] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, *GridFTP: Protocol Extensions to FTP for the Grid,* Argonne National Laboratory, April 2002.

[7] The NaradaBrokering System http://www.naradabrokering.org

[8] Shrideep Pallickara and Geoffrey Fox. NaradaBrokering: *A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids.* Proceedings of ACM/IFIP/ USENIX International Middleware Conference. 2003.

[9] G. Fox, S. Lim, S. Pallickara and M. Pierce. *Message-Based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services.* (To appear) Journal of Future Generation Computer Systems.

[10] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke, *Data Management and Transfer in High Performance Computational Grid Environments* Parallel Computing Journal, Vol. 28 (5), May 2002, pp. 749-771.

[11] G. C. Fox, W. Wu, A. Uyar and H. Bulut *Design and Implementation of Audio/Video Collaboration System Based on Publish/subscribe Event Middleware* Proceedings of CTS04 San Diego January 2004

PDPTA’04

Paper ID #: PDP2157

Title: **Web Service Robust GridFTP**

Sang Lim, Geoffrey Fox, Shrideep Pallickara*, and Marlon Pierce
Indiana University, Bloomington, Indiana, USA