GWD-R
GridFTP WG
http://www-isd.fnal.gov/gridftp-wg/

Igor Mandrichenko, FNAL (editor)

July 3, 2003
Revised July 11, 2003

GridFTP Protocol Improvements

(draft, 7/11/2003)

Status of this Document

This document is a Global Grid Forum Draft summarizing the analysis of GridFTP v1.0 protocol and experience of using the protocol in building distributed grid applications.

Abstract

GridFTP protocol has become popular data movement tool used to build distributed grid-oriented applications. GridFTP protocol extends FTP protocol defined by RFC959 [rfc959] and other IETF documents by adding certain features designed to improve performance of data movement over wide area network, to allow the application to take advantage of “long fat” communication channels, to help build distributed data handling applications.

Several groups have developed independent implementations of GridFTP v1.0 [gftp] protocol for different types of applications. This document summarizes the experience gained by these groups and their proposals for possible protocol improvements. The goal of these improvements is to develop more robust, reliable and scalable protocol for bulk file-oriented data transfer over wide and local area networks.

Contents

GridFTP Points of Improvements

The following are the issues identified by GridFTP Working Group as possible points of GridFTP protocol improvements.

Unidirectional Data Transfer in Extended Block Mode

GridFTP v1.0 requires that in extended block mode data is sent in the same direction as data channel connection is established. In other words, GridFTP server can send data only in active mode and receive data only in passive mode. This makes it impossible to use extended block mode in presence if firewall, etc.

There are several possible solutions of this problem:

- Pre-negotiation of number of data channels to be used for the transfer
- Modification of the protocol possibly introducing new transfer mode along with existing Stream, Block and Extended Block modes.

Ordering of PASV/SPAS and STOR/RETR Commands

As defined in RFC959, in passive mode server must reply to PASV command with the address of the data socket *before* it receives STOR or RETR command, and thus before it receives the name of the file to be transferred. In case of distributed server, this is not always possible. This makes it very difficult to implement passive mode in distributed FTP server in scalable way.

There are many proposed solutions for this problem:

1. Delayed passive option for PASV command which would allow to defer answer to the PASV command until STOR/RETR is received and include data socket address into (unused) answer to STOR/RETR
2. Introduction of PRET command as discussed in GridFTP v1.0 document which would carry attributes of the file which is about to be transferred and issuing PASV after PRET
3. Introduction of new pair of commands, GET and PUT which would essentially combine functionalities of (1) and (2) into single command/reply, preserve semantics of STOR/RETR and eliminate the need for PASV

Possible Disconnection of Idle Control Channel Socket by Some Firewalls

This problem is inherited from RFC959 FTP protocol. Some firewall software drops idle TCP connections. In some applications, such as disk cache in front of tape storage, data existing in name space is not always immediately accessible. In these cases, after issuing the STOR/RETR command, the client must wait for relatively long time before data transfer can even start. This makes the control channel socket connection idle for long time, and the firewall can drop it. The same may apply to data channel as well.

Proposed solution for control channel seems to be easier than for data channel. Performance data proposed in GridFTP v1.0 draft can be periodically sent over the control channel to keep it alive. As for data channel, some sort of keep-alive noise could be sent in the direction opposite to data transfer.

Control of Contents and Frequency of Feedback From the Server

Currently GridFTP server sends performance markers at fixed 5 second interval and restart markers come at a predefined block size. There should be some way to allow the interval between restart and performance markers to be set. Also it may be useful to allow this to be extensible so that other transfer event data could be returned as well, for example if the end host was a mass storage system and it were staging a file, it might send back ETA or % done markers.

Possible solutions would be to use FEAT/OPTS mechanism or introduction of new command (tentatively TREV for or TRansfer EVent).

Unreliable End-of-File Communication in Stream Mode

As specified by the RFC959, during data upload, the server is supposed to treat end of data socket as end of file. This makes it impossible for the server to distinguish between normal end of file and abnormal client shutdown in the middle of data transfer.

Possible solution for this problem is to introduce EOF command, which would be sent by the client to confirm that the entire file was sent successfully over the data socket.

Data Integrity Verification

In order to protect data from transmission errors, some data integrity verification mechanism should be introduced on the level of FTP protocol, in addition to TCP packet checksums. Some sort of CRC or another form of digital signature should be calculated either over each block of data in block-oriented transfer modes such as Block and Extended Block or over whole file in Stream mode.

Structured Directory Listings

RFC959 does not specify format of directory listing sent in response to LIST command. Usually it is well suited for human reading, but not for computer processing. Extensions to FTP Internet Draft [ftpext] proposes new MLST and MLSD commands, which would have easy to parse output. This proposal should be adopted by GridFTP protocol.

IPV6 Support

IPV6 uses significantly different addressing schema than IPV4. RFC 2428 introduces EPRT and EPSV commands suitable for IPV6. Endorsement of the extensions proposed by RFC2428 [rfc2428] should be considered.

Packed Transfers

In cases when it is necessary to transfer large number of small files, time spent on transfer initiation over the control channel and establishing data channel may create significant inefficiency in overall transfer performance. Partially this problem may be solved by reusing of data channels without opening and closing them for each file.

Another solution to consider is to pack multiple files into single file, transfer this file and then unpack it on the receiving end. ESTO and ERET commands already provisioned by GridFTP protocol seem to be useful in organizing such packed transfers.

Flexible Striping

There should be a way to dynamically and flexibly control striping algorithms.

References

gftp W.Allcock, et al, GridFTP: Protocol Extensions to FTP for the Grid, Global Grid Forum Draft, http://forge.gridforum.org/projects/ggf-editor/

rfc959 J.Postel, J.Reinolds, File Transfer Protocol (FTP) http://www.ietf.org/rfc/rfc0959

rfc2428 M.Allman, S.Ostermann, C.Metz, FTP Extensions for IPv6 and NATs http://www.ietf.org/rfc/rfc2428

ftpext Robert Elz, Paul Hethmon, Extensions to FTP, IETF Draft http://www.ietf.org/internet-drafts/draft-ietf-ftpext-mlst-16.txt