File Transfer in the Grid

The ARDA Group

Editor: B. Koblitz

Abstract: We present a quick overview on the properties and performance of RootD, GridFTP and RFIO from the point of view of a grid-based analysis environment.

Introduction

In this paper we give an overview of three file transfer protocols which are considered for deployment in the LHC Computing Grid: The RootD file transfer daemon of the Root analysis environment [ROOT], the GridFTP file transfer application from the Globus toolkit [GLOB] and the RFIO file transfer program[1].

For all three protocols a small client program was written which simulates several clients by using the Linux `fork()` system call. Using these clients the stability and performance of the servers were tested when handling several concurrent transfers. All three file transfer protocols also allow reading parts of the files and the test clients made use of this capability by reading parts of the remote file into a buffer before writing the data out to a local file, except for the RootD protocol, which offers remote-access only to Root data files and for which our client-implementation used the FTP like direct file transfer API.

Security and General Behaviour

The RFIO protocol was designed for the transfer of files in a trusted environment with a minimum of overhead. Thus it is the only transfer protocol which currently does not use a secure authentication (this is currently added to the protocol) and therefore lays open the overhead of authentication when comparing with the two other protocols. However, RFIO offers a minimum of authentication by sending the user's name as well as user-id and group-id to the server which then checks access privileges. Both GridFTP and RootD offer authentication with strong encryption through SSL and SSH (RootD also allows other other authentication mechanisms but SSH is the most convenient to set up for a single site). None of the protocols does encrypted data transfers by default, however GridFTP offers this as an option.

The desired behaviour of a reliable file-transfer protocol should be to allow as many simultaneous connection as is possible without endangering the stability and responsiveness of the server, as well as to provide useful feedback to the client. Table 1 summarizes the results of the file transfer protocols using several concurrent file requests by the provided clients[2]. The number of connections

1 For RootD the CVS version from May 12th 2004 including several patches for RootD from G. Ganis which were published in version 4.00/06a was used, GridFTP from Globus 3.2 and RFIO version 1.6.1.

2 `rfcp` in the case of RFIO, `globus-url-copy` in the case of GridFTP and a custom client written using the RootD-API.

	grid-ftp	rootd	rfio
Number of clients accepted by default	40/s (xinetd)	10 (default of sshd for concurrent authentication processes)	no direct limit, only time-outs
Abandoned daemons	no	yes	yes
Empty files if no connection	yes	-	no

Table 1: General behaviour of the 3 file transfer protocols.

is limited for GridFTP by the xinetd which is used to start the GridFTP servers when clients connect. The default for this is 40 connections per second, after which further connections are refused. For RootD this limit is the number of 10 concurrent SSH authentication sessions provided by the sshd. Note that this does not limit the total number of transfers by RootD, only the number of concurrently connecting clients. RFIO finally does not seem to have a hard limit. Instead rfcp commands fail with time-outs. While the feed-back of GridFTP to the user is quite good, RootD does have difficulties figuring out the cause for the refused connection. It will consider SSH authentication to be not-working and will consequently flood the user with the results of other failed authentication attempts via different authentication methods[3]. This feed-back is only given on stderr. There is no way for a program to understand the underlying problems because the relevant calls in the RootD API do not report any errors on return.

Another behaviour tested was what happens if client programs were prematurely terminated: In this case RootD and RFIO could leave abandoned server processes. In the case of RootD this prevented further connections to the server. GridFTP has the peculiarity that when connections failed empty files were created on the client side. Hanging client processes have been observed for RootD (again rendering the server useless) and for GridFTP.

Concerning the API the three solutions differ drastically. While RootD only allows for remote access to root-files via the streaming mechanisms, plus a possibility to initiate a file transfer, RFIO provides a full POSIX-inspired access mechanism to the remote files via adapted open, read/write and close commands using a file number as handler. GridFTP finally comes with a comprehensive API which offers monitoring of the transfers as well as it's own multithreading model. Programming for this API is difficult and entails much longer programs (a factor of 3 in case of out client program compared to the RFIO solution).

3 This behaviour was the reason for the patches by G. Ganis, unfortunately his patches did not entirely solve this problem.

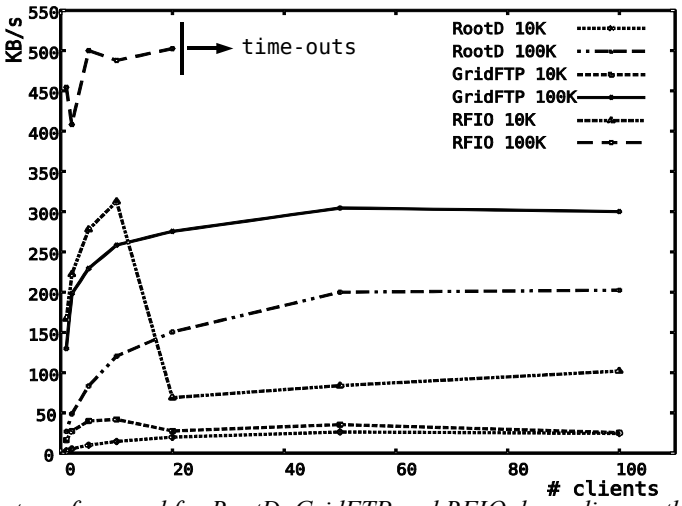

Figure 1: File transfer speed for RootD, GridFTP and RFIO depending on the number of clients concurrently transferring files. For RFIO time-outs prevent a reliable transfer for more than 20 clients when transferring 100KB files.

Performance of Large File Transfers

All three file transfer applications provide similar performance when transferring large files in their standard configurations. GridFTP can optionally use multiple streams for data transfer, which can increase the file transfer speed over WAN connections considerably [GSIF]. In our test set-up using a dual PIII server at 800MHz and with 1GB of memory connected to a P4 2.8GHz client with 512MB of memory via a 10MBit Ethernet connection, the transfer rate for large (1GB) files was in the order of 570KB/s and differed little for the three file transfer applications.

Many Concurrent File Transfers

While single large file-transfers are considered to be the usual use-case for file-transfer in the Grid as a production environment, the situation as seen in current user analysis environments is often different [ABH]: Users will read only small parts of many files, for example if they are interested in a small subset of high energy physics events which are spread over many files. In this situation, the file access speed and the capability of the server to handle many client requests becomes important. Also clients erroneously starting several transfers should not crash the system. In the following, we look into the stability and performance of the file transfer applications using several concurrent transfers.

In order to test the performance of the three file transfer applications, client programs were written which forked into several concurrent processes reading a remote file into a buffer and writing the data out to a local file (in case of RootD only a file transfer was initiated). The tests were done with relatively small files of 10KB and 100 KB size so that the access speed dominates the performance. The result as the transferred KB per second depending on the number of concurrently connecting clients is shown in Figure 1 and represents the performance of the server accessing many files. It is evident that RFIO has a problem with many concurrent request. For files of a 100KB size it time-outs for more than 20 concurrent clients. For smaller files its performance drastically

decreases at this point. Its very high performance for low numbers of connecting clients can be explained by the small overhead of authentication. This overhead is highest for RootD, due to its usage of SSH. All protocols have in common that their transfer capability increases first with growing numbers of clients and then start to drop again. This underlines the usefulness of multiple streams for data transfers. Interestingly, the much larger implementation of GridFTP offers a much higher performance than RootD.

Conclusions

The performance of all three investigated file transfer protocols RootD, RFIO and GridFTP is similar for large files, while it differs by an order of magnitude for small files (10KB and 100KB) where the file accessing speed is dominant and thus the overhead due to authentication is high. All three solutions still have some problems with handling serving several concurrent requests and especially the reporting of errors to the application in a way so that it can react in a sensible way.

Bibliography

ROOT: http://root.cern.ch
GLOB: http://www.globus.org
GSIF: http://www.ihep.ac.cn/~chep01/paper/10-012.pdf
ABH: http://www.slac.stanford.edu/~ahb/xrootd