Benchmarking third-party-transfer protocols with the FTS

Rizart Dona

CERN Summer Student Programme 2018

Supervised by Dr. Simone Campana & Dr. Oliver Keeble

1.Introduction

The aim of this project is to benchmark third-party-transfer protocols by using the FTS[1], through a software toolkit built in python. The protocols that are benchmarked are GridFTP[2], HTTP[3] and XRootD[4] and the testbed consists of three endpoints in the Worldwide LHC Computing Grid[5]. GridFTP is the most commonly used protocol at CERN for data transfer, the community though is contemplating a transition from it to another protocol (prompted largely by the withdrawal of support for GridFTP), HTTP and XRootD are the candidate protocols to replace it. This study essentially makes a comparison between them in terms of performance for latency (per single file), for different file sizes.

Worldwide LHC Computing Grid

The Worldwide LHC Computing Grid (WLCG) is a global computing infrastructure whose mission is to provide computing resources to store, distribute and analyse the data generated by the Large Hadron Collider (LHC), making the data equally available to all partners, regardless of their physical location. WLCG is the world's largest computing grid, in this project three WLCG endpoints are employed for the purpose of benchmarking the protocols.

FTS

The GRID Data Transfer Service used at CERN is called File Transfer Service (FTS); it is a data movement service. The FTS aims to reliably copy one storage URL to another, it uses third party copy transfer to achieve this and in the case of failure it retries the transfer. It also schedules these copies along network channels to ensure that bandwidth is properly used. State in the FTS is held in a database, which ensures that the service can be restarted reliably. The FTS is used by the experiment frameworks (typically the end-users do not interact directly with it) which submit jobs to

[1] "fts.web.cern.ch | Shipping data around the world." https://fts.web.cern.ch/. Accessed 6 Sep. 2018.
[2] "GridFTP - Wikipedia." https://en.wikipedia.org/wiki/GridFTP. Accessed 6 Sep. 2018.
[3] "Hypertext Transfer Protocol - Wikipedia." https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. Accessed 6 Sep. 2018.
[4] "XRootD: Home Page." http://xrootd.org/. Accessed 6 Sep. 2018.
[5] "WLCG: Welcome to the Worldwide LHC Computing Grid." http://wlcg.web.cern.ch/. Accessed 6 Sep. 2018.

the FTS; a job is a set of pairs with source and destination file names. The FTS is the main technology that is used to execute the protocol experiments that are presented in this report.

Third Party Copy

A third party copy means that when doing a copy between two remote endpoints, the data is sent directly between the two participating storages. This comes in contrast to the non-third party copy where the data goes through the client. All three protocols (GridFTP, HTTP, XRootD) that are examined support third party copy but it always needs to be the same protocol for both endpoints (e.g. from GridFTP to GridFTP). In the context of this project only third-party-copy scenarios are examined.

The remaining of this report is structured as follows. In section 2 the methodology of the benchmarking is described as well as the parameters that the software toolkit accepts. In section 3 the experimental results are presented along with some plots that explain better the produced data. In section 4 some future work is discussed, and finally, in section 5 conclusions are mentioned.

The code of the toolkit can be found at those interrelated repositories:

- FTS Benchmark Toolkit (https://gitlab.cern.ch/rdona/fts-analysis)
- FTS Benchmark Visualization (https://gitlab.cern.ch/rdona/fts-analysis-viz)

2.Methodology

The benchmarking setup consists of three WLCG endpoints, one serves as the source endpoint and the other two as the destination endpoints. Each one of the destination endpoints is located at a different location for the purpose of different test cases. The source endpoint (*dpmhead-trunk.cern.ch*) is located inside CERN, the first destination endpoint (*lapp-se01.in2p3.fr*) is located at Annecy, France and the second destination endpoint (b2se.mel.coepp.org.au) is located at Melbourne, Australia. The Annecy and Melbourne endpoints are examined in the context of short and long round trip transfer (rtt) time respectively. All three endpoints (source and destinations) are used with the Disk Pool Manager[6] (DPM) as storage technology.

[6] "DPM - Disk Pool Manager | LcgDM - Data Management Servers - CERN." http://lcgdm.web.cern.ch/dpm. Accessed 2 Oct. 2018.

For each pair of source - destination (e.g. CERN - Annecy), the toolkit is used to submit jobs through the FTS. In this study, each job contains ten file transfers and 100 jobs are submitted, the file sizes are 1MB, 1GB and 3GB. The retry option of the FTS is not used at all during those executions. The toolkit also supports the FTS options for checksum verification as well as for overwriting of files.

The next json snippet illustrates an example configuration file that is used to run the experiments.

```
{
    "source_url": "dpmhead-trunk.cern.ch/dpm/cern.ch/home/dteam/rdona/",
    "dest_url": "b2se.mel.coepp.org.au/dpm/mel.coepp.org.au/home/dteam/rdona/",
    "num_of_jobs": 1,
    "num_of_files":[10],
    "filesizes":[1, 1000, 3000],
    "protocols": ["gsiftp", "root", "https"],
    "checksum": "none",
    "overwrite": false
}
```

What this json configuration essentially determines is that the number of jobs that are going to be executed is the cartesian product of the options for each field (num_of_jobs $\times$ num_of_files $\times$ filesizes $\times$ protocols). Each combination will be executed and the statistics will be extracted. The results that are presented in the next section are produced from keeping the checksum flag “none” (no checksum verification) and the overwrite flag “false” (the destination endpoint is always empty before the actual copy).

3.Experimental Results

In this section the experimental results are presented along with some plots. More specifically, for each test case (Annecy, Melbourne) the plots are:

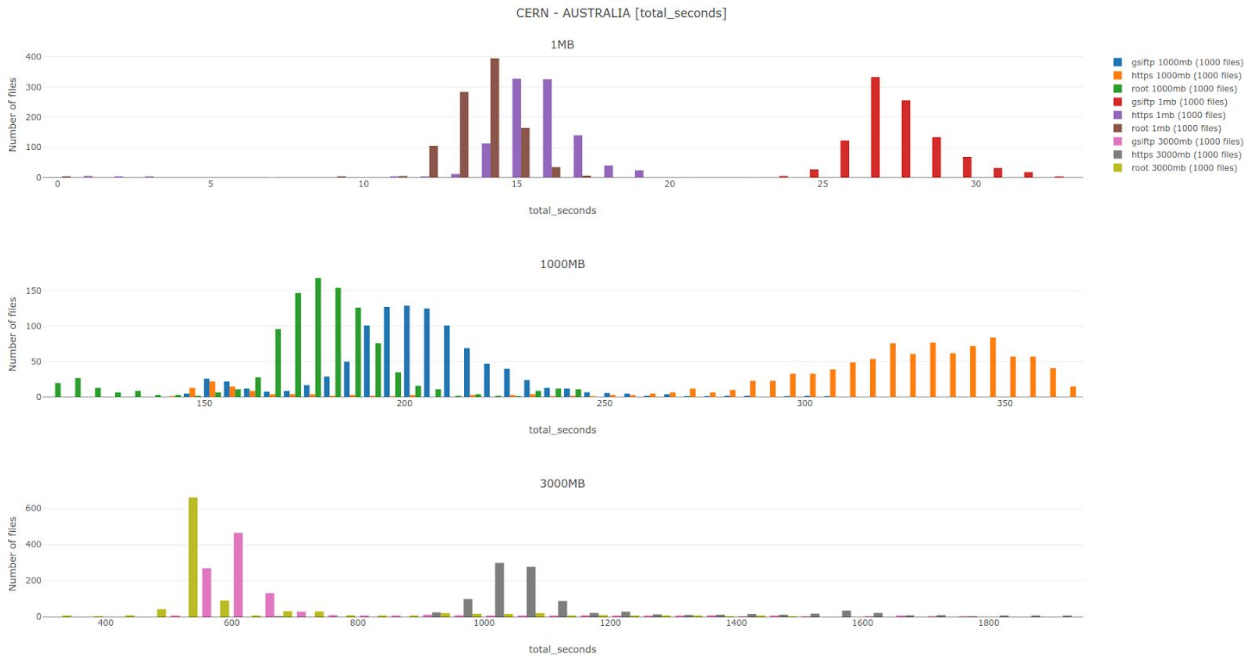
- Histograms of transferred files for total seconds, per size, for each protocol.
- Line plots of average total seconds, per size, with an error bar ($\pm$std), for each protocol.

Total seconds for each file is defined as the time it took for a single file to be transferred completely from the source to the destination. Std refers to the standard deviation[7] which is used to produce an error bar for the line plots.

[7] "Standard deviation - Wikipedia." https://en.wikipedia.org/wiki/Standard_deviation. Accessed 3 Oct. 2018.

Melbourne Test Case

The next plot corresponds to the histograms of transferred files for total seconds, per size, for each protocol.



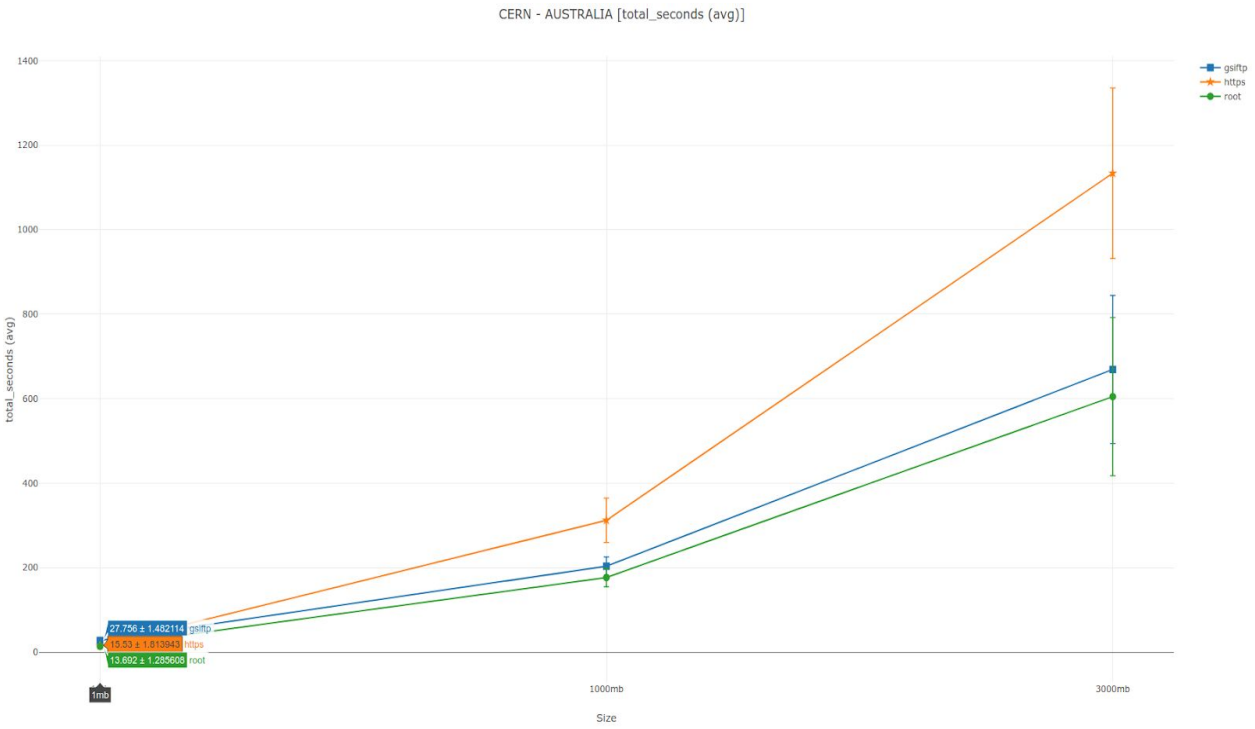
As one can see, all size-protocol histograms of transferred files follow more or less something that resembles a normal distribution[8]. It is expected that some files are transferred a lot faster than the majority and some others are transferred a lot slower, most files though tend to gather around a small area of the time scale. For each combination that is shown, 1000 files are transferred.

When the average of the total seconds it took for the files to be transferred is taken from the aforementioned histograms a line plot can be extracted for each protocol per size. The charts/plots that follow represent those averages.

[8] "Normal distribution - Wikipedia." https://en.wikipedia.org/wiki/Normal_distribution. Accessed 4 Oct. 2018.

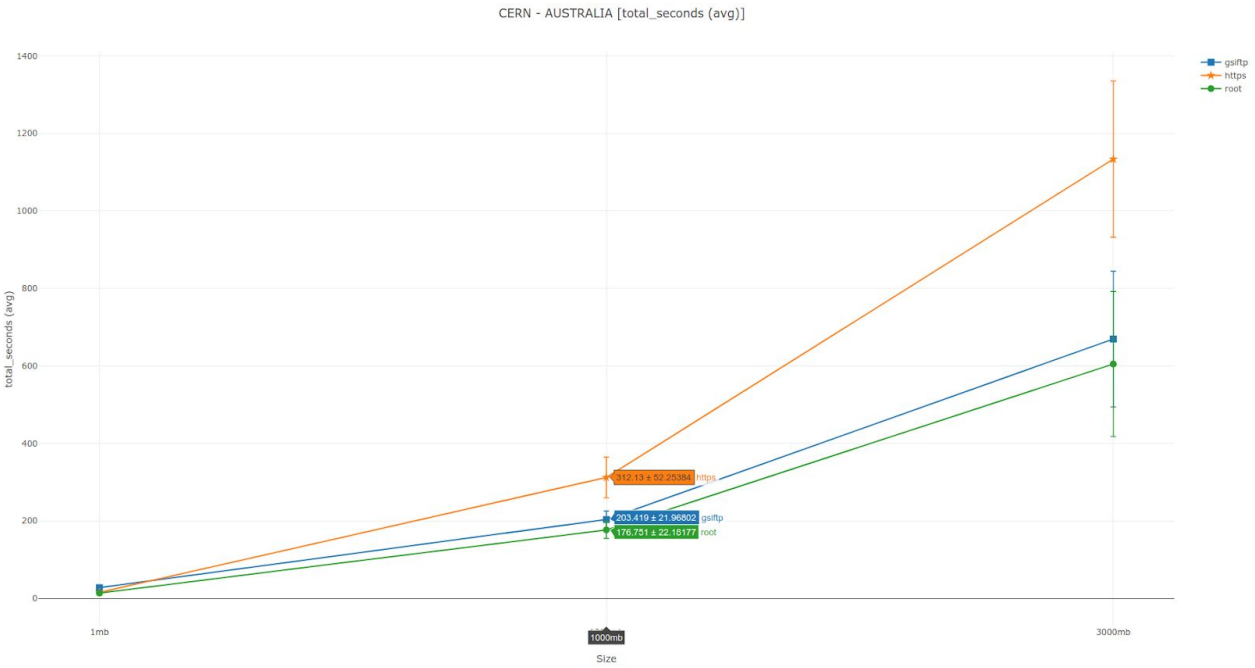
1MB Files

Protocol	Avg Total seconds	std	Number of files
GridFTP	27.75	1.48	1000
XRootD	13.69	1.28	1000
HTTP	15.53	1.81	1000



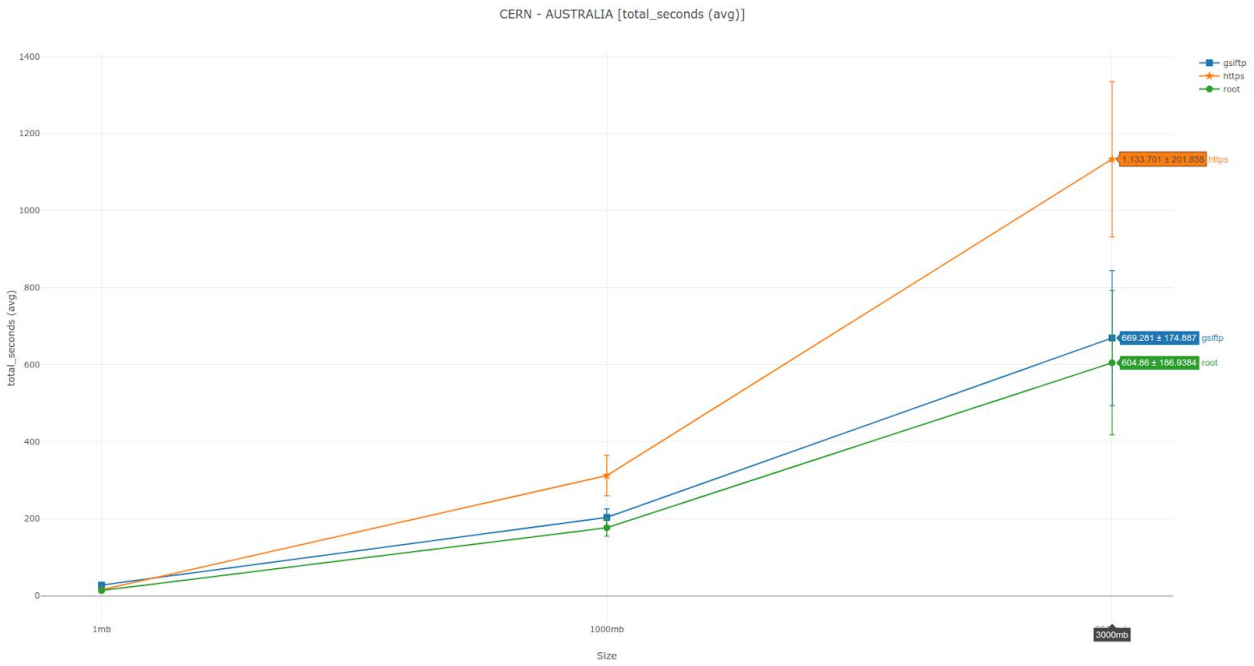
1GB Files

Protocol	Avg Total seconds	std	Number of files
GridFTP	203.41	21.96	1000
XRootD	176.75	22.18	1000
HTTP	312.13	52.25	1000



3GB Files

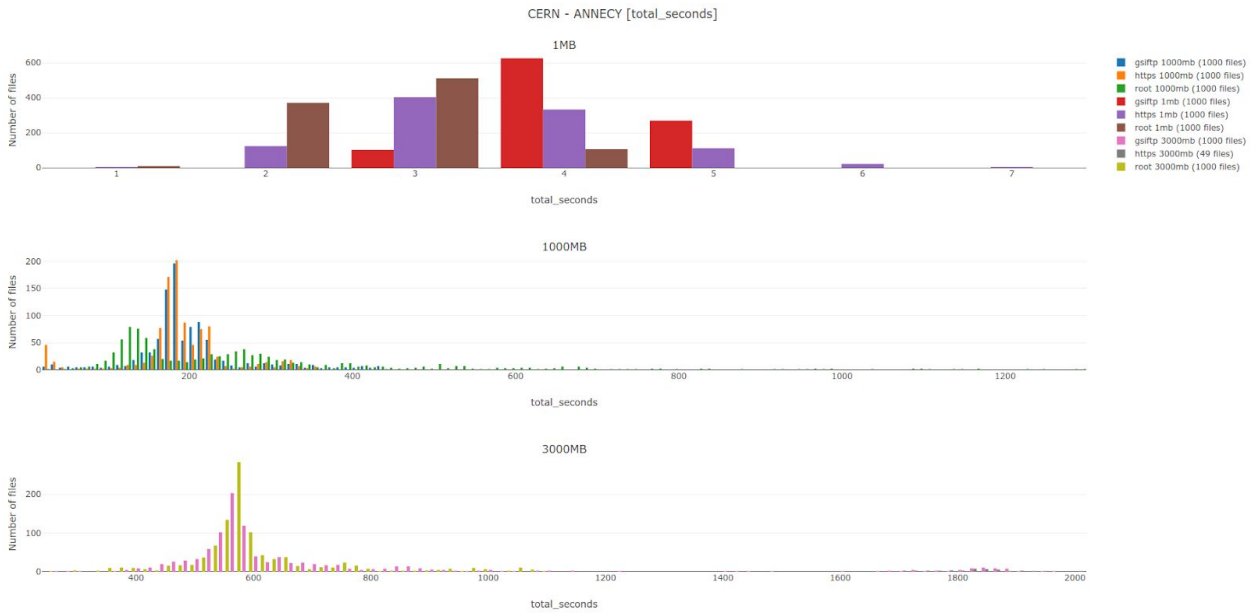
Protocol	Avg Total seconds	std	Number of files
GridFTP	669.28	174.88	1000
XRootD	604.86	186.93	1000
HTTP	1133.70	201.85	1000



Those results show that XRootD and GridFTP perform very close one to another for the larger file cases (1GB and 3GB), HTTP on the other hand falls way back in those. This bad performance of HTTP could be explained due to the fact that the destination endpoint is configured to encrypt all data traffic, the encryption process certainly slows down the transfer. GridFTP is slower than the other two protocols in the 1MB case, mainly because of the overhead the user has to pay in order to establish the connection. Concerning those results, one could argue that XRootD outperforms both other protocols for the Melbourne case (i.e. for the long rtt case).

Annecy Test Case

The next plot corresponds to the histograms of transferred files for total seconds, per size, for each protocol.

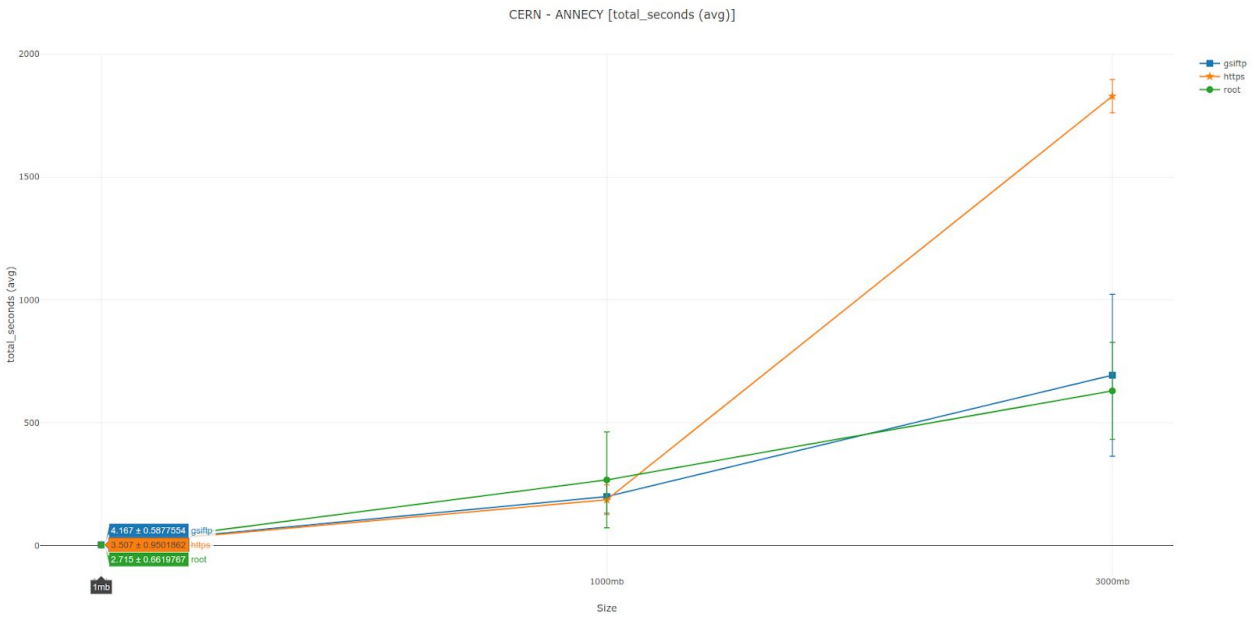


Here, most size-protocol histograms of transferred files again follow more or less something that is close to a normal distribution. However, one can notice that for the XRootD - 1GB case as well as for the HTTP - 3GB case the data points are more scattered.

As in the Melbourne case, the charts/plots that follow represent the respective averages.

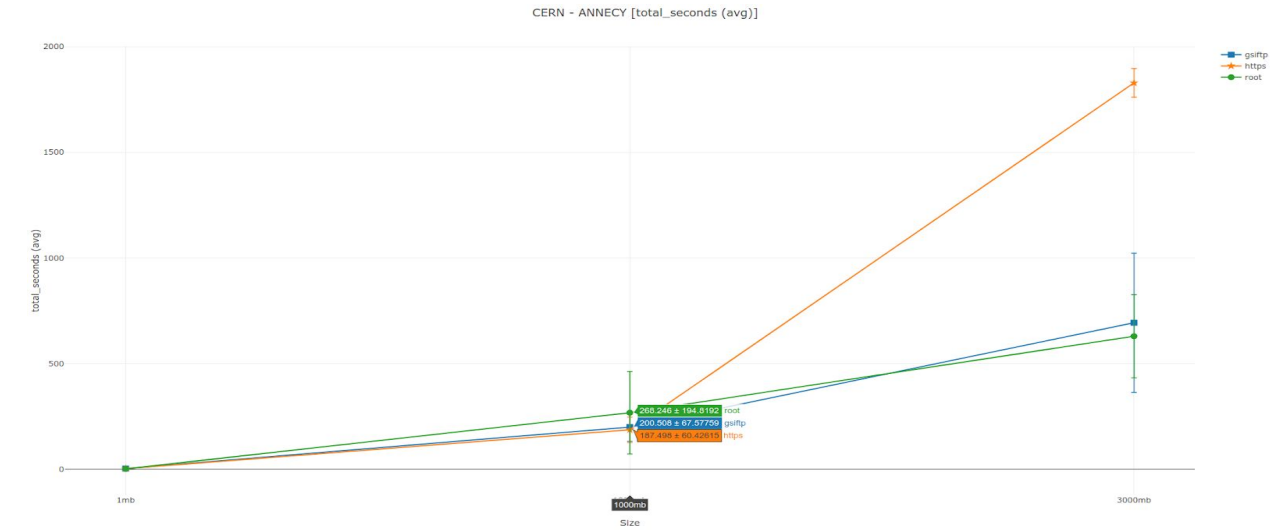
1MB Files

Protocol	Avg Total seconds	std	Number of files
GridFTP	4.16	0.58	1000
XRootD	2.71	0.66	1000
HTTP	3.50	0.95	1000



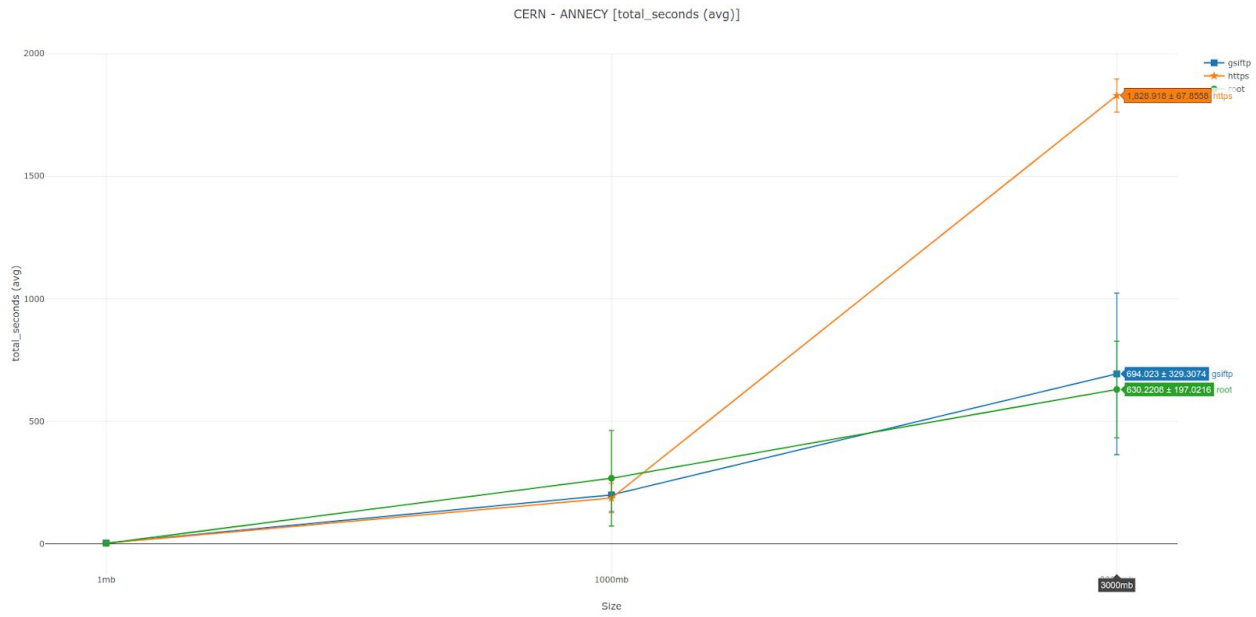
1GB Files

Protocol	Avg Total seconds	std	Number of files
GridFTP	200.50	67.57	1000
XRootD	268.24	194.81	1000
HTTP	187.49	60.42	1000



3GB Files

Protocol	Avg Total seconds	std	Number of files
GridFTP	694.02	329.30	1000
XRootD	630.22	197.02	1000
HTTP	1828.91	67.85	49



Those results show that for the 1MB case all three protocols perform more or less the same. For the 1GB case one can see that HTTP is way better than both other protocols in terms of average time and std. In the 3GB case, XRootD outperforms GridFTP slightly in terms of average time but it greatly outperforms it in terms of std. As one can notice, in the last case HTTP has only 49 data points (transferred files), so no trustworthy observation can be made for this behavior.

Failed Jobs

The results that are presented in the previous two test cases were gathered in a period of about two weeks. The transferred files consist of successful transfers that occured in different jobs and in different datetimes. Many jobs failed during those experiments and many files that are presented are part of jobs that just didn't fail completely (that means that in a single job some files could fail and some others could finish). This is particularly the case in the HTTP - 3GB case where only 49 file transfers succeeded out of 1000+ tries.

Those failed file transfers can be explained in two different dimensions. The first one has to do with the infrastructure, the CERN endpoint is testbed (not used in production) and some components are not working flawlessly in such demanding scenarios. The second dimension has to do with XRootD and with HTTP, the functionality of the third party copy in those protocols is experimental and some of the issues are already reported as software bugs that need fixing.

4.Future Work

In this section some possible future work is presented. Part of that work has to do with the software toolkit and some other aspects of it are concerned with further enhancement of the study.

One useful addition to the toolkit would be the functionality to visualize failed files/jobs in order to make it easier in the future to analyze what went wrong with the transfers. The toolkit already provides the option to store log files of failed file transfers but those logs are very verbose and need a lot of human effort to read them and interpret them. A visualization plugin would certainly help the analyst to explore better the data and observe patterns of failure.

Further enhancement of the study would consist of more useful plots from the produced data. The toolkit provides the functionality of visualizing any field of interest (for example throughput). This means that it is possible to explore more dimensions that are essential for a more complete study.

One last point would be the extension of the options for the storage technology that was used to benchmark the protocols. All results that were presented used DPM at the endpoints, a useful addition would be to explore how those protocols behave in other scenarios where other storage systems are used.

5.Conclusions

Some conclusions can be mentioned concerning this study. For the Melbourne case (long rtt) the XRootD protocol looks the most promising in terms of latency. Certainly GridFTP is more stable and has been the standard for a long time now in the community but what this report shows is that there is space for further improvement by using alternative technologies. For the Annecy case, the results can be described as inconclusive. There is no obvious pattern that emerges from the data and of course the HTTP - 3GB case is not representative.

One final conclusion would be that there is a need for further investigation. This study does not take into account the state of the endpoints at the time of the experiments (i.e. total traffic, overload). Those results represent an instance of the endpoints that were used, more experiments should be performed in order to determine with greater certainty if those patterns are consistent in other scenarios.