Characterizing Performance and Fairness of Big Data Transfer Protocols on Long-haul Networks

Se-young Yu, Nevil Brownlee, and Aniket Mahanti
Department of Computer Science
University of Auckland
Auckland, New Zealand
{se-young.yu, n.brownlee, a.mahanti}@auckland.ac.nz

***Abstract*—This paper presents a characterization study of big data transfer protocols on a long-haul network. We analyzed the performance and fairness of three well-known open-source protocols, namely, GridFTP, FDT, and UDT. Using a real-world 10 Gb/s network link between New Zealand and Sweden, we studied data transfer rates (in terms of goodput) and fairness (in terms of impact on round trip time) of the protocols. We performed extensive experiments using single and multiple data flows to comprehend how these protocols behave in real-world situations. We found that GridFTP has the fastest data transfer rates when using a single flow. UDT suffered from poor performance due to implementation issues. A small buffer size limited FDT's performance, however, this drawback can be overcome by using multiple flows in lieu of fairness.**

I. Introduction

Large scientific installations such as the Large Hadron Collider and the Square Kilometer Array produce very large amounts of raw data every day. These data (often referred to as *big data*) are processed locally and distributed to researchers all over the world for storage and analysis.

There are two categories of big data transfer protocols: (a) *TCP-based protocols* [1], [2] that depend on the existing TCP congestion avoidance algorithms, and (b) *UDP-based protocols* [3], [4] that utilize their own congestion control algorithm. While implementing a protocol's own congestion control may be more efficient (e.g., to increase throughput in long-haul networks), however, such an algorithm needs to also consider bandwidth fairness.

We surveyed commonly used big data transfer protocols used in *eScience* research projects and chose the following protocols: HPN-SSH, GridFTP, FDT, UDT, and Tsunami. We chose these protocols because they are open-source, have wide range of users in the *eScience* research community, and can be readily run on Linux workstations. We ran a comprehensive suite of experiments using our chosen set of big data transfer protocols in our 10 Gb/s local [5] and national [6] testbeds. During these experiments, we found HPN-SSH and Tsunami did not function sufficiently well. Thus, we focus on the following three protocols for rest of the paper: *GridFTP*, *FDT*, and *UDT*.

GridFTP [1] is an free software implementation, which provides extensions to FTP. It provides enhancements such as parallel data transfer, data striping, and TCP socket buffer optimization. *FDT* [2] is another free software implementation that uses TCP as the transport protocol. In addition to parallel data transfer and socket buffer optimization, it also provides multiple I/O threads and platform independent implementation. *UDT* [3] is an UDP-based connection-oriented data transfer protocol with its own congestion control algorithm, called *Decreasing Increases AIMD (DAIMD)*. UDT [3] aims to provide TCP-friendly congestion control on top of UDP, using DAIMD and rate control to achieve high throughput over long-haul networks.

In this paper, we present a characterization study of big data transfer protocols on a long-haul international network. We established a 10 Gb/s network link between New Zealand and Sweden. We used this testbed to perform experiments on the performance and fairness of GridFTP, FDT, and UDT. Our results show that GridFTP is well-suited in long-haul networks to provide fast transfers. This is due to its implementation that makes it stable as well as its ability to set a large TCP socket buffer size. To the best of our knowledge, this is the first work to study big data transfer protocols on a high-speed international network. Our results can be used by scientists to choose an appropriate big data transfer protocol that suits their research goals.

The rest of the paper is organized as follows. Section II discusses related work. Section III discusses our experimental setup and data collection methodology. Performance and fairness analysis of the big data transfer protocols are discussed in Section IV and Section V. Section VI concludes the paper.

II. Related Work

There has been limited work on characterizing performance and fairness of open-source big data transfer application-layer protocols in long-haul networks.

Ha *et al.* [7] compared fairness and link utilization of several high-speed TCP variants such as BIC, CUBIC, FAST, HSTCP, H-TCP, and STCP using the `dummynet`[1] network emulation tool with varying levels of background traffic in a local testbed network. They found that all the variants showed less fairness to TCP as the delay increased.

Cottrell *et al.* [8] compared throughput, stability, fairness, and CPU utilization of several protocols such as HSTCP, Scalable TCP, and Fast-TCP using a 1 Gb/s academic and research network at the Stanford Linear Accelerator Centre. They found that scalable TCP, BIC, and H-TCP were most efficient while

[1]http://info.iet.unipi.it/ luigi/dummynet/

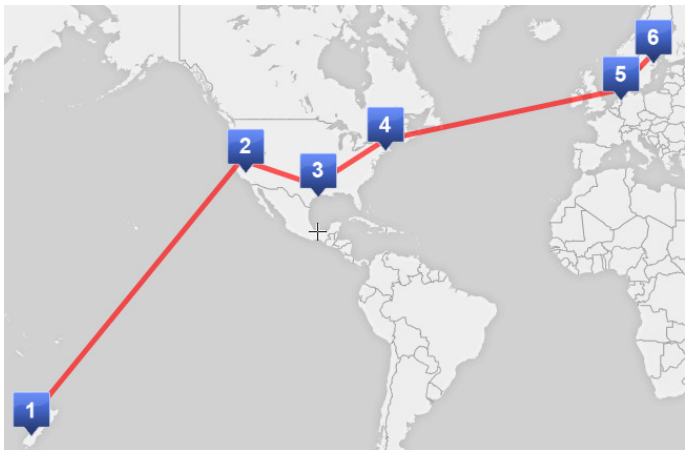

Fig. 1. Path between the experiment machines. Each flag shows different hops of the network path. 1 is Queenstown, NZ, 2 is LA, USA, 3 is Houston, USA, 4 is New York, USA, 5 is Amsterdam, Netherlands and 6 is Stockholm, Sweden.

TCP Reno, HSTCP-LP, and HSTCP did not utilize network capacity effectively. They used both TCP and UDP-based protocols, however, their network capacity was limited to 1 Gb/s.

Suresh *et al.* [9] evaluated throughput, fairness, and CPU usage of GridFTP, GridCopy and UDT. They found that UDT performed well compared to the rest of the tested protocols. GridFTP also performed better, except when transferring smaller files. While this study compared the performance of GridFTP with UDT, their testbed was restricted to using a 2 Gb/s campus network.

Tierney *et al.* [10] compared performance of RDMA over converged Ethernet, TCP, and UDP in ESnet's 10 and 40 Gb/s link with 47 ms round trip time (RTT). Since, this work focused on comparing transport-layer protocols using a dedicated interface for RDMA, we cannot compare the results directly with our work. We use TCP or UDP over Ethernet and instead characterize performance and fairness of different application-layer protocols. Furthermore, our experiments are performed in very high RTT network, which is typical for transfers between New Zealand and Europe.

Yu *et al.* [5], [11] compared big data transfer protocols in a 10 Gb/s local testbed network, and in [6] compared the big data transfer protocols in a national 10 Gb/s network. Our work extends these works by characterizing performance of the protocols in a 10 Gb/s long-haul international network.

Previous studies have used different metrics for measuring fairness. In this work, we measure RTT changes in the link during the data transfer as a measure of fairness. RTT changes may provide pointers on potential impact on the network.

III. Experimental Setup

Research and Education Advanced Network New Zealand (REANNZ)[2] hosted the the 14^{th} Annual Global LambdaGrid Workshop[3] in Queenstown, New Zealand. The conference organizers established a 100 Gb/s connection between Queenstown and Los Angeles, USA. We were able to use this network to conduct big data transfer experiments using a 10 Gb/s path between Queenstown and Stockholm, Sweden, as shown in Figure 1.

TABLE I
Summary of big data transfer protocols

Protocol	Version used	Transport protocol	Congestion control	Multiple data flows
GridFTP	6.0	TCP	TCP CUBIC	Supported
FDT	0.19.2	TCP	TCP CUBIC	Supported
UDT	4.7	UDP	DAIMD	Not Supported

`Traceroute` revealed that there were 13 hops in the network path, and the average RTT for the path was 323 ms during the experiment, without any experimental traffic injected. The Queenstown-side of network was operated by REANNZ (From 1 to 2 in Figure 1). NORDUnet[4] operates the Stockholm site (From 5 to 6 in Figure 1), and there are Internet2[5] (From 2 to 4 in Figure 1) and GEANT-operated[6] links (From 4 to 5 in Figure 1) in between them. To avoid other network traffic affecting our experiments, we conducted our data transfers between 12 a.m. and 4 a.m. We also noticed the link capacity was lightly utilized by other users. We did not notice any major path changes during the experiments.

We set up two machines – sender and receiver – each with a 10 Gb/s Ethernet interface and Linux operating system. We then installed GridFTP, FDT and UDT on both these machines. We chose these three protocols based on their performance in our internal testbed [5]. When the Globus Toolkit was updated to version 6, GridFTP was updated to support UDT. This update enabled us to perform memory-to-memory transfers. Multiple flows are not supported by UDT. Table I provides a summary of the protocols evaluated in our experiments.

The sender machine's role is to initiate a connection and send a 30 GB data file, or streamed data from `/dev/zero` to avoid disk read overheads, ten times using GridFTP, FDT and UDT. We used one through eight data flows for GridFTP and FDT. The receiver machine accepts this connection and writes the transferred data to its file system unless it is from `/dev/zero`, otherwise the data is sent to to `/dev/null` to avoid disk write overheads.

Host TCP tuning for each machine was done as per instructions in Energy Science Network's host tuning guide[7]. We used CUBIC TCP for our congestion control algorithm. Jumbo frames were enabled by default. We did not increase the TCP sender's buffer size for FDT because it is implemented in Java. We found that changing the Java Virtual Machine's maximum memory size, and that of FDT's TCP socket buffer made FDT unstable. For GridFTP, we set our TCP sender's buffer size to

[2]http://www.reannz.co.nz/

[3]http://www.glif.is/meetings/2014/

[4]https://www.nordu.net/

[5]http://www.internet2.edu/

[6]http://www.geant.net/

[7]http://fasterdata.es.net/host-tuning/linux/

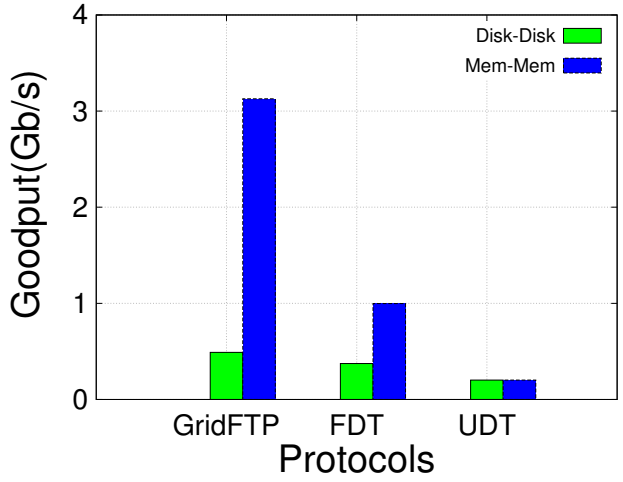

Fig. 2. Goodput measured for protocols using a single flow

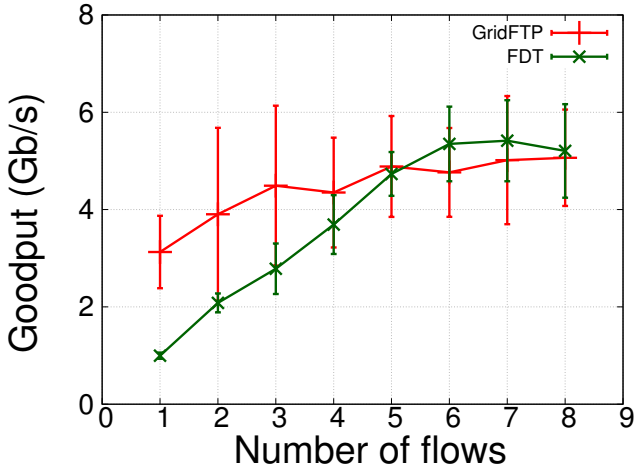

Fig. 3. Goodput measured for each protocol with 1-8 multiple flows, error bars show standard deviation

an arbitrary size of 180 MB.

IV. Performance Evaluation using a Single Flow

For all our experimental file transfers we recorded packet traces using `tcpdump`. We first present results from our analysis of a single data flow followed by discussion of experiments involving multiple flows.

Figure 2 shows the performance of each file transfer using a single flow. GridFTP performed better with file transfer, indicating that it is most efficient at reading and writing via the `EXT4` file system, compared to the other two protocols. Although these protocols have different transfer rates (i.e. measured goodput), their transfer rates are all less than 0.5 Gb/s. It was evident that file system I/O performance was limiting the file transfer rate. Hence, we decided to remove that bottleneck by reading data from `/dev/zero` and writing it to `/dev/null`. After removing the bottleneck, GridFTP improved more than 600% and FDT improved more than 260%. UDT showed no significant difference when the file system I/O limitation was removed.

In our previous study, using a smaller national testbed [6], all protocols achieved significantly higher throughput. With disk-to-disk transfer, GridFTP, UDT and FDT achieved 1.3 Gb/s, and with memory to memory transfer, GridFTP achieved 5 Gb/s and FDT achieved 4.5 Gb/s. The main causes of reduced performance in the international testbed are large RTT and small TCP socket buffer sizes. Compared to our national testbed, where RTT was 10 ms, this testbed has an RTT 30 times longer. This causes CUBIC TCP to take longer to increase its congestion window size, as well as for UDT to increase its sending rate after packet losses.

V. Performance Evaluation using Multiple Flows

We also measured the multiple flow performance of GridFTP and FDT. For each test, we used memory-to-memory transfers to start and terminate multiple flows at the same time. We did not use UDT because it does not support multiple data flows. We first report on data transfer results followed by fairness results.

A. Data Transfer Rate

Figure 3 shows goodput for each protocol using one to eight parallel data flows. Compared to our previous study [6], there was a significant increase in throughput as the number of flow increased for both GridFTP and FDT.

The goodput for both protocols increases until it reaches a maximum (5 Gb/s for GridFTP and 5.5 Gb/s for FDT). The goodput of FDT increases more rapidly compared to that of GridFTP, and the effect of using multiple flows is greater. This is due to the limitation of TCP socket buffer size.

When multiple flows are used, recovery time for their aggregate throughput is reduced. When multiple flows increase their congestion window size at the same time at similar rates, the aggregate congestion window increase rate is much higher than that of a single flow. Even though there is a higher chance of losing packets for the aggregated flow, the recovery time to reach its maximum congestion window size is reduced. Along with CUBIC TCP, which steadily increases congestion window size when it is near a previously determined maximum, its shorter recovery time allows both protocols to increase their goodput with multiple flows in the long-haul network.

This also happened in our national network experiment [6], which had shorter RTT, however the recovery time there was small compared to that of the longer path international network. The benefit of having shorter recovery time by using multiple flows diminishes with small RTTs, because the recovery time for a single flow is almost as small as the recovery time for the multiple flows.

In a network with shorter RTT (e.g. our national network), the benefit of using multiple flows is reduced. With its smaller Bandwidth-Delay Product (BDP) link, a smaller buffer size allows more throughput. With a small packet error rate, the effect of faster recovery gets diminished, and single flow with

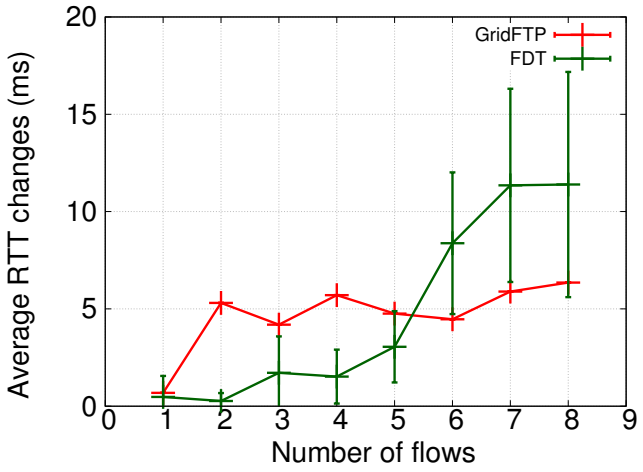

Fig. 4. RTT changes measured for each protocol with 1-8 multiple flows, errorbar for standard deviation

moderate buffer size will be as efficient as the multiple flows because the recovery time for any flow is relatively short.

B. Impact on RTT

We measured RTT values while using GridFTP and FDT by measuring baseline RTTs before running the experiments and then during the experiments using ICMP packets with the `ping` command. We calculated the average time difference between the baseline RTTs and the increased RTTs during the experiments for each protocol with different number of flows, to investigate the effect of congestion caused by the protocols building up queues in routers along the path.

Figure 4 shows average increase in RTT measured for each protocol with growing number of multiple flows. As the number of flows increases, the average RTT change also increases. The average RTT change for two to five flows for GridFTP is higher, although it has high variability, so it has more impact on RTT than FDT. GridFTP is able to produce more aggregated traffic compared to FDT, and as a result, its impact on the network path is greater. Since, both protocols use the same underlying congestion control algorithm, an increase in goodput for either protocol leads to an increase in average RTT. Overall, the average RTT change is higher with multiple flows because more traffic is injected by the multiple flows.

We found that there are different baseline RTTs for each flow, one on top of another. We identified with `traceroute` that there is a load-balancer within the path that causes flows to be forwarded into different links for a part of their route. This produces the slight difference in RTT for each flow.

VI. Concluding Remarks

We characterized the performance and fairness of three big data transfer protocols, namely, GridFTP, FDT, and UDT, on a long-haul international network. We found that GridFTP has the fastest data transfer rate when using a single flow. UDT has an implementation issue, which limited its performance. FDT also has an issue with small buffer size limiting its performance, however this drawback can be overcome by using multiple flows in exchange for fairness.

To have better performance, faster file systems and larger TCP socket buffers in both operating system and application are required. With our long-haul link, running multiple flows for a high-speed transfer were beneficial, especially for FDT (where there was a practical limit on the TCP socket buffer size). It also reduced recovery time after packet loss events.

We did not find much difference between GridFTP and FDT, but GridFTP was more stable in terms of handling file system and TCP socket buffer. FDT is easier to set up because it is implemented as a Java application, which does not require a high level of system permission.

References

[1] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, "GridFTP: Protocol extensions to FTP for the Grid," *Global Grid ForumGFD-RP*, vol. 20, pp. 1–21, Apr. 2003.

[2] I. Legrand, H. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, C. Dobre, A. Muraru, A. Costan, M. Dediu, and C. Stratan, "MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems," *40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures*, vol. 180, no. 12, pp. 2472–2498, Dec. 2009.

[3] Y. Gu and R. L. Grossman, "UDT: UDP-based Data Transfer for High-speed Wide Area Networks," *Comput. Netw.*, vol. 51, no. 7, pp. 1777–1799, May 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2006.11.009

[4] M. Meiss, *Tsunami: A High-Speed Rate-Controlled Protocol for File Transfer*, 2009. [Online]. Available: www.evl.uic.edu/eric/atp/TSUNAMI.pdf/

[5] Se-young Yu, N. Brownlee, and A. Mahanti, "Comparative performance analysis of high-speed transfer protocols for big data," *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pp. 292–295, 2013.

[6] ——, "Performance and Fairness Issues in Big Data Transfers," in *Proceedings of the 2014 CoNEXT on Student Workshop*. Sydney, Australia: ACM, 2014, pp. 9–11.

[7] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, "A step toward realistic performance evaluation of high-speed TCP variants," *Elsevier Computer Networks (COMNET) Journal, Special issue on PFLDNet*, Feb. 2006.

[8] R. Les Cottrell, S. Ansari, P. Khandpur, R. Gupta, R. Hughes-Jones, M. Chen, L. McIntosh, and F. Leers, "Characterization and evaluation of TCP and UDP-based transport on real networks," *Annales Des Tèlècommunications*, vol. 61, no. 1-2, pp. 5–20, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1007/BF03219966

[9] J. Suresh, A. Srinivasan, and A. Damodaram, "Performance Analysis of Various High Speed Data Transfer Protocols for Streaming Data in Long Fat Networks," in *Proc. International Conference on ITC*, Kochi, Kerala, India., Mar. 2010, pp. 234 –237.

[10] B. Tierney, E. Kissel, M. Swany, and E. Pouyoul, "Efficient data transfer protocols for big data," in *E-Science (e-Science), 2012 IEEE 8th International Conference on*, Oct. 2012, pp. 1–9.

[11] Se-young Yu, N. Brownlee, and A. Mahanti, "Comparative Analysis of Transfer Protocols For Big Data," in *31st International Symposium on Computer Performance, Modeling, Measurements and Evaluation 2013: Student Poster Abstracts*, Vienna, Austria, 2013, pp. 5–6.