Comparative Performance Analysis of High-speed Transfer Protocols for Big Data

Se-young Yu, Nevil Brownlee, and Aniket Mahanti
Department of Computer Science
University of Auckland, New Zealand

***Abstract*—Researchers working in diverse fields such as astronomy, experimental physics, genomics, and meteorology have to frequently deal with analyzing voluminous amounts of complex data. Such data is often referred to as big data. These researchers work in teams and have to transfer this data over long distances. Efficiently transferring big data over long distances requires the use of appropriate transfer protocols. Several TCP-based and UDP-based protocols have been proposed in the literature, however, a comparative analysis of such protocols is lacking in the literature. This paper presents a comparative performance analysis of four well-known high-speed data transfer protocols for long fat networks, namely, GridFTP, FDT, UDT, and Tsunami. We performed extensive experiments to measure the effectiveness of each protocol in terms of its throughput for various round-trip times, and against increasing levels of congestion inducing TCP or UDP background traffic on a 10 Gb/s network. Our results show that without much tuning, TCP based protocols are able to achieve throughputs of more than 2 Gb/s. In presence of background traffic, UDP protocols perform better.**

I. Introduction

Data-intensive research involving the Large Hadron Collider (LHC), Square kilometer Array (SKA), and International Thermonuclear Experimental Reactor (ITER) produce large volumes of data over relatively short periods. These data need to be transferred via high-speed networks, typically 10 Gb/s, to other locations to be analyzed by teams of researchers collaborating around the world. TCP has inherent limitations regarding high-speed links due to its preservative congestion control algorithms.

When there is a packet loss TCP reduces its window size, thus yielding some of the bandwidth to competing flows. This Additive Increase Multiplicative Decrease (AIMD) behaviour can prevent TCP from using the full bandwidth of an LFN [1]. Furthermore, this behaviour can cause poor link utilization during data transfer, and lead to unfair allocation of available bandwidth between concurrent flows.

To overcome this limitation and utilize available bandwidth during high-speed data transfer with large bandwidth delay product (BDP) links, a number of protocols have been proposed [2], [3], [4], [5], [6]. There are a number of TCP variants such as TCP CUBIC [6] which modify TCP's congestion control algorithm to increase throughput over LFNs. Some protocols use UDP (e.g., UDT, Tsunami) [3], [4] with their own congestion control, overcoming TCP limitations by increasing throughput more rapidly from the start or by decreasing their throughput less when a packet loss occurs. Others such as GridFTP [2] and FDT [5] use multiple TCP sessions with data striping to achieve a larger window size, thus increasing the throughput.

There has been some work on analyzing transfer protocols for transferring big data using analytical and simulation techniques [7], [8], however, an experimental evaluation is lacking. In this paper, we use experiments performed on a 10 Gb/s testbed network to perform a comparative analysis of four well-known high-speed transfer protocols, namely, GridFTP, FDT, UDT, and Tsunami. Our goal is to understand the behaviour of each of these protocols over a congested long-haul link as well as under varying level of background traffic. Furthermore, we want to compare TCP-based and UDP-based protocols and see which protocols perform better under the stated conditions.

Our results show that while TCP-based protocols are more efficient in transporting over short distances (low round trip times), UDP-based protocols may be well suited for longer distances (high round trip times). UDP-based protocols are able to reach their maximum throughput much faster than TCP-based protocols. In the presence of background traffic, we found rapid decrement in the throughput of data transfers for TCP-based protocols.

The rest of the paper is organized as follows. Section II describes the four analyzed transfer protocols. Section III discusses related work. Section IV presents our data collection and analysis methodology. Section V presents results of our analysis. Section VI discusses the implications of the presented results. Section VII concludes the paper.

II. Background

High-speed data transfer protocols for LFN can be classified into two categories; TCP-based protocols [2], [5] and UDP-based protocols [3], [4]. TCP-based protocols make use of TCP tuning and parallel connection to achieve higher throughput while UDP-based mechanisms use their own congestion control algorithms to avoid the possible inefficiency of slow start and AIMD. Table I summarizes the key features of four protocols evaluated in this paper.

UDT [3] is an UDP-based connection-oriented data transfer protocol with Decreasing Increases AIMD (DAIMD). *GridFTP* [2] is an extension to FTP that includes enhancements such as parallel data transfer, data striping and better access to the stored data. *FDT* [5] uses TCP at the transport layer with concurrent threads and connections during the transfer. *Tsunami*

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

TABLE I: High-Speed Transfer Protocols for Long Fat Networks

Protocol	Year first developed	Latest Version	Transport-layer protocol	Congestion control	Multiple connections	Example usage
UDT	2004	4.7	UDP	DAIMD	Supported	Transfer data between CERN and MANLAN
GridFTP	2001	5.2.3	TCP	TCP AIMD	Supported	Remote access to simulation data
FDT	2006	0.18.0	TCP	TCP AIMD	Supported	Hybrid TCP
Tsunami	2002	1.1	UDP	Rate Control	Not Supported	Radio astronomy data transfer

[4] is another UDP-based high-speed data transfer protocol with rate control and adjustable error threshold.

III. Related Work

High-speed data transfer protocols have been evaluated in theory and in simulated environments previously, however they have not been analyzed in an experimental setting. An analytical and simulation-based study on these protocols may not reflect realistic performance of each protocol. Using a testbed, we can observe the protocols themselves rather than their models.

Suresh *et al.* [9] evaluated throughput, fairness and CPU usage of GridFTP, GridCopy and UDT with a 2 Gb/s network. They found that UDT performed well compared to the other protcols. GridFTP also performed well, except when transferring smaller files. Weston *et al.* [10] transferred astronomy data from New Zealand to Germany and Finland using two UDP-based protocols, UDT and Tsunami, via the New Zealand Research and Education network. They found that UDT showed better performance and fairness than Tsunami, but also identified fairness issues in Tsunami. This study, however, did not evaluate any TCP-based protocols. Yue *et al.* [11] compared four UDP-based protocols, RBUDP, Tsunami, PA-UDP and UDT in terms of throughput, round trip times, loss rate, fairness and CPU utilization using a 1 Gb/s link. They found that PA-UDP got optimal performance. UDT was most convenient to use due to its parameterless setup.

Our work complements previous work by performing an experimental analysis of four well-known protocols for transferring big data over LFNs including TCP-based and UDP-based protocols. We also analyze individual protocol behaviour under varying levels of congestion inducing TCP and UDP background traffic.

IV. Methodology

A. Testbed setup

We built a testbed network in our lab with four machines to measure behaviour of a selection of high-speed data transfer protocols. Two machines act as receiver and sender of the transfer. There are two switches, each is connected to sender and receiver and the switches are connected each other with Directed Attach Copper cables allowing 10 Gb/s transfer rate. Based on our initial experiments, we decided to measure performance of *GridFTP with TCP*, *GridFTP with UDT*, *Tsunami*, and *FDT*.

Since we did not have access to a hardware traffic generator, we chose to use an existing application program to generate (congestion-inducing) background traffic from sender to receiver. We used `nuttcp` to generate TCP and UDP traffic. To observe their behaviour with various RTTs, we decided to use `netem`[1] to emulate different delays in the testbed. We tuned the test machine's TCP parameters by following ESnet's Linux Tuning guide.[2] To keep our comparisons fair, we did not use concurrent connections for any of the protocols, although GridFTP with TCP and UDT, and FDT support concurrent connections for better link utilization. Tsunami does not support multiple simultaneous connections, and GridFTP with UDT did not work well with concurrent connections in our testbed.

B. Factors that influence transfer rate

1) Round trip times: High-speed data transfer protocols behave differently with varying RTTs. We analyzed the performance of the protocols in our testbed with RTT values in the range 0 ms (local hosts) to 200 ms (US Midwest).

2) Background traffic: We used `nuttcp` to generate TCP and UDP background traffic for a scenario involves data transfer between New Zealand and Western Australia where Murchison Radio-Astronomy Observatory located. An RTT of 75 ms is chosen to represent the link between the sites.

To induce congestion, a number of concurrent TCP flows were run alongside each protocol to observe their behaviour when there is a significant amount of TCP traffic. Since TCP congestion control decides the transfer rate at a given time, we chose to increase the number of background TCP flows rather than increasing the background TCP data rate.

For UDP, we generated $10,000 - 40,000$ packets of $1,250$ byte datagrams to induce congestion on the link with an emulated 75 ms RTT. The packet size and number of packets per second were selected according to the level of congestion likely to be induced. For $10,000$ packets per second, we hardly saw packets dropped by congestion while with $40,000$ packets per second we observed about 33% of UDP datagrams being dropped due to congestion. With TCP background traffic, we were investigating congestion events induced by TCP AIMD. With UDP background traffic, we were interested in observing congestion events induced with constant rate UDP.

V. Performance Evaluation

We measured throughput of each protocol when they were used to transfer data in our testbed network with varying RTTs as well as varying levels of TCP and UDP background

[1] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
[2] http://fasterdata.es.net/host-tuning/linux/

traffic. We also measured the time for each protocol to achieve its initial throughput, and the CPU resource they used. The results were gathered by repeating 10 sets of experiments, after removing outliers due to malfunctioning of `nuttcp` or the data transfer protocols. We use statistical measures such as mean, 15^{th} percentile, and 85^{th} percentile to present our results.

A. Impact of increasing distance

Figure 1 shows the throughput (in Gb/s) when there was no background traffic, and therefore no serious congestion. Overall, performance of FDT and GridFTP with TCP was better than GridFTP with UDP or Tsunami in this environment. Although the transfer rate decreases more rapidly as RTT increases, FDT and GridFTP with TCP showed better performance over GridFTP with UDT and Tsunami, at least for RTTs of 200 ms or less.

For increasing RTT, TCP-based protocols performed much better than UDP-based protocols. Performance of FDT for less than 100 ms RTT were the best among all four protocols. The highest throughput of FDT was 2.34 Gb/s with a 1 ms RTT. However, the throughput of FDT decreased faster compared to GridFTP with TCP for RTTs greater than 100 ms. For RTTs greater than 100 ms, GridFTP with TCP performed the best.

UDP-based protocols did not perform well for most of the RTTs we tested. GridFTP with UDT performed poorly for most of the RTTs. Although its author claims that UDT is more efficient for increasing RTT [3], the rate of decreased throughput for increasing RTT from 0 ms to 100 ms was the fastest among all protocols tested. At RTT 200 ms, UDT was the slowest among all the protocols.

Throughput of Tsunami did not decrease much as RTT increased, however its throughput was the lowest among all four protocols for RTTs less than 200 ms. Tsunami showed throughput of 0.55 Gb/s across all the RTTs we tested.

B. Impact of background traffic

We emulated a network path between New Zealand and Western Australia in our testbed network by using an RTT of 75 ms. Then, we generated different levels of TCP background traffic and UDP background traffic. To generate this cross-traffic, we created an `nuttcp` process that generates multiple TCP flows or multiple UDP packets per second so as to generate increasing volumes of background traffic.

Figure 2 shows the throughput for each protocol over a path with 75 ms RTT and increasing volumes of TCP cross-traffic. The throughput of TCP-based protocols became unreliable with increasing volumes of TCP background traffic. Even with a small amount of TCP background traffic, FDT becomes unusable due to the induced congestion. The throughput of GridFTP with TCP decreased rapidly as number of TCP flows increases. On the other hand, GridFTP with UDT and Tsunami showed stable performance over background TCP flows.

FDT showed its highest performance when there was no background traffic, but as soon as we added some background traffic, the application collapsed and become unusable. GridFTP with TCP suffered a progressive decrease in throughput with increasing TCP background traffic. Moreover, it became unstable and its overall throughput decreased compared to the UDP-based protocols.

The throughput of GridFTP with UDT shows good performance with increasing TCP background traffic. After having 3 TCP flows at the background, it starts to outperform TCP-based protocols, and the throughput did not vary much.

Tsunami's performance was slowest with increasing TCP background traffic, but it seemed to be very stable against background traffic. Increasing background traffic did not affect its throughput significantly.

Figure 3 shows the throughput for each protocol over a path with 75 ms RTT and increasing volumes of UDP background traffic. With UDP background traffic, the throughput of GridFTP with TCP and FDT decreased rapidly after some level of congestion was induced by increasing UDP background traffic. GridFTP with UDT was also affected by tha traffic, but the decrease in throughput was small compared to the TCP-based transfer protocols. Tsunami was less affected by the UDP background traffic during the experiments.

Our results suggest that using GridFTP with UDT may work best for transferring big data over moderate distances. If the network is not likely to be congested during the transfer, using GridFTP with TCP is clearly faster. If we can use a dedicated path between two hosts where there is no possible congestion, FDT will be the best choice for data transfer.

VI. Discussion

1) Experimental Challenges: To have more control over the many factors in the experiments while still approximating real-world situations, we used decided to create a testbed rather than doing a simulation. On the other hand, we are well aware that we are testing the implementation of protocols, not protocols themselves.

The focus of this study is not on the optimization of throughput of each protocol using careful protocol tuning. Rather, we wanted to observe their behaviour with various environmental limitations that they are likely to encounter, including different RTTs and levels of background traffic, with limited transmit queue sizes. The stability in paths with background traffic, and the need to not disrupt too much other traffic on those paths, are important to demonstrate that the protocols can be used in real-world situations. During our experiments, we found that `nuttcp` become unstable when it is used to create too many tcp sessions in a long RTT link. This may have particularly affected the average throughput of the protocols where they are used with more than four TCP background traffic flows. We tried repeating experiments to reduce uncertainty in our results, but we were not able to stabilize `nuttcp` behaviour or that of other traffic generators. This could be improved by using a dedicated traffic generator, which we will explore in future work.

With no background traffic, we could observe a slightly lower throughput of GridFTP with UDT at short RTT region.

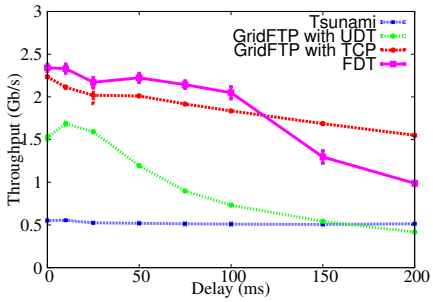

Fig. 1: Throughput of the protocols with varying RTT, with no background traffic

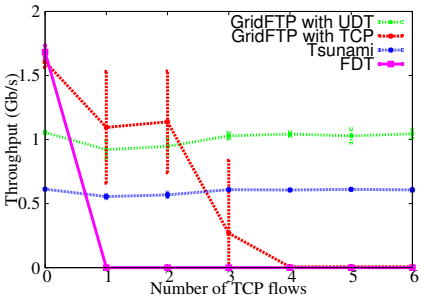

Fig. 2: Throughput of the protocols with TCP background flows with RTT 75 ms

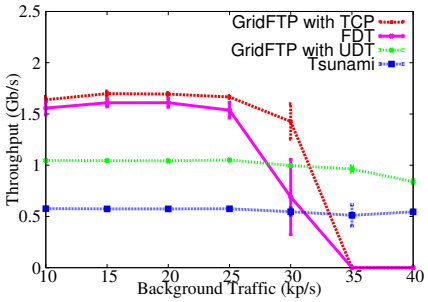

Fig. 3: Throughput of the protocols with UDP background traffic, RTT 75 ms

We suspect this effect is due to its implementation rather than the protocol itself.

2) Result Implications: From our results, using TCP-based protocols for transferring big data would be beneficial especially when the links used have relatively short RTTs and where there is no significant background traffic. On the other hand, with significant background traffic, TCP-based protocols may not perform well due to the lack of stability. Using GridFTP with UDT may give better performance with smaller RTT links. Tsunami is only recommended for a high-RTT link, and when there is significant background traffic on the link.

VII. Conclusions

We presented a comparative performance analysis of four well-known transfer protocols for big data, namely, GridFTP, FDT, UDT, and Tsunami. Our results show that using GridFTP with TCP or FDT gives higher throughput over a 10 Gb/s network link. Unless there is a significant background traffic inducing congestion, using GridFTP with TCP may be the best choice for data transfer between distant hosts. On the other hand, with significant level of congestion on the link, the TCP-based protocols show a tendency to be unstable, hence using GridFTP with UDT may give more stable throughput. We recommd using Tsunami only if there is significant background traffic and the RTT between hosts is high.

In contrast, using a UDP-based protocol over links with high background traffic is preferable, since it can provide more reliable and efficient data transfer. The TCP-based protocols were not stable, or efficient enough to compete against high levels of TCP background traffic. Also, they took longer to increase their initial throughput at the start of the transfer compared to UDP based protocols.

VIII. Future work

We plan to extend our testbed to test transfer protocols over long Internet paths, such as between Auckland and Sydney. Using this extended testbed, we will be able to understand how the high-speed data transfer protocols behave on a longer-haul link where there is realistic congestion causing variable round-trip times. We are also looking into using software defined networking concepts to improve performance of high-speed data transfer. Using OpenFlow [14], we believe we can bypass firewall and other obstructions between sender and receiver.

References

[1] V. Jacobson, R. Braden, and D. Borman, *TCP Extensions for High Performance*, ser. Request for Comments. IETF, May 1992, no. 1323, published: RFC 1323 (Proposed Standard). [Online]. Available: http://www.ietf.org/rfc/rfc1323.txt

[2] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, "GridFTP: protocol extensions to FTP for the grid," *Global Grid Forum GFD-RP*, vol. 20, 2003.

[3] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks*, vol. 51, no. 7, pp. 1777 – 1799, 2007.

[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] 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.

[6] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[7] F. Baccelli, G. Carofiglio, and M. Piancino, "Stochastic analysis of scalable tcp," in *Proc. IEEE International Conference on INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 19–27.

[8] D. M. Lopez-Pacheco and C. Pham, "Enabling large data transfers on dynamic, very high-speed network infrastructures," in *Proc. International Conference on ICN/ICONS/MCL*, 2006, pp. 40–40.

[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] S. Weston, T. Natusch, and S. Gulyaev, "Radio astronomy data transfer using KAREN network," in *Proc. General Assembly and Scientific Symposium on IEEE URSI*, Istanbul, Turkey, Aug. 2011, pp. 1 –4.

[11] Z. Yue, Y. Ren, and J. Li, "Performance evaluation of UDP-based high-speed transport protocols," in *Proc. IEEE International Conference on ICSESS*, Beijing, China, Jul. 2011, pp. 69 –73.

[12] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, and I. Foster, "Gridftp pipelining," in *Proc. TeraGrid Conference*, Wisconsin, USA, Jun. 2007.

[13] A. Rajendran, P. Mhashilkar, H. Kim, D. Dykstra, G. Garzoglio, and I. Raicu, "Optimizing large data transfers over 100Gbps wide area networks," in *Proc. IEEE/ACM International Symposium (CCGrid)*, Delft, Netherlands, Nov. 2012.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69–74, Mar. 2008.