Net-Preflight Check:Using File Transfers to Measure Network Performance before Large Data Transfers

Bashir Mohammed
Scientific Data Management(SDM)
Lawrence Berkeley National Lab
Berkeley,CA, USA
email:bmohammed@lbl.gov

Mariam Kiran
Energy Sciences Network(ESnet)
Lawrence Berkeley National Lab
Berkeley,CA, USA
email:mkiran@es.net

Bjoern Enders
Natl. Energy Research Sci.Comp.Center(NERSC)
Lawrence Berkeley National Lab
Berkeley, CA, USA
email:benders@lbl.gov

***Abstract*—During bulk data transfer for exascale scientific applications, measuring the available throughput is very useful for route selection in high-speed networks, Quality of service verification, and traffic engineering. Recent years have seen a surge in available throughput estimation tools, especially in Research and Education (R&E) Networks. Some tools have been proposed and evaluated in simulation and over a limited number of Internet paths. However, there is still significant uncertainty in the performance and flexibility of these tools at large. Furthermore, some existing tools' primary concern is the lack of network performance history or a memory that stores previous configurations and network measurements. This paper introduces Net-PreFlight, a simple end-to-end, lightweight tool for measuring available throughput, traceroute, and maintains memory, compared to existing tools like Iperf. Our tool focuses on throughput measurements, flexibility, a retentive memory, security, and performance. We conduct experiments between multiple Data Transfer Node (DTN) setups in isolated and public network setups to measure how throughput measurements fare in the two domains. In all scenarios, Net-Preflight produces comparable results as established tools and hence positions itself as a complementary tool for situations where the deployment of Iperf or perfSONAR is not possible. In addition, Net-Preflight features retentive memory to easily compare past and present measurements. Our analysis reveals that using socket and file transfer protocols performs well in initial measurements and also indicates that parallel TCP streams are equivalent to using a large maximum segment size on a single connection in the absence of congestion. Here, we lay the foundation to build a new monitoring system for DTN bulk transfers that target end-users who require optimum network performance.**

***Keywords*—Throughput measurement, Data Transfer, TCP, Network performance monitoring**

I. Introduction

Recent years have seen a strong interest in techniques for estimating available throughput and bandwidth along an Internet network path. The path diversity in R&E or overlay networks creates a need for estimating the available throughput over these paths as a method for choosing the best time and network route before initiating a bulk data transfer [1]. However, in an overlay or traditional network, one can assume the cooperation of both the sender and the receiver, which is necessary for most probing techniques. Available throughput measurement during bulk data transfer for exascale scientific applications is very useful for route selection in overlay networks, Quality of service verification and traffic engineering [2]. A few tools have been proposed and evaluated in simulation and over a limited number of internet paths, but there is still great uncertainty in the performance of these tools over isolated and public network, as well as R&E Networks at large. For instance, Iperf is a popular network monitoring tool for active measurements of the maximum achievable bandwidth on IP networks, and it supports tuning of various parameters related to timing, buffers and protocols, such as TCP and UDP. For each test, Iperf3 reports the bandwidth, loss, and other parameters. However, a major limitation with iperf3 is lack of flexibility because you have to install it at both ends of the measurement (Server-Client), and also lack of retentive memory and performance history which stores previous configurations and network measurements.

Another popular existing tool is the performance Service-Oriented Network monitoring Architecture (perfSONAR) [3], which is a network measurement toolkit designed to provide federated coverage of paths and help to establish end-to-end usage expectations. There are thousands of perfSONAR instances deployed worldwide and many of which are available for open testing of key measures of network performance. It is an essential tool that ensures scientists can rely on networks to get their data from end-to-end as quickly as possible. However, even though perfSONAR provides a uniform interface that allows for the scheduling of measurements, storage of data in uniform formats, as well as scalable methods to retrieve data and generate visualization, it has a similar limitation with Iperf3 because you need to install a perfSONAR instance or node, at both server and client-end, before you can run tests for accurate measurements. It also does not have retentive memory capability, which means users or network engineers are unable to see previous configurations and network performances to compare with current measurements [3].

To bridge this gap, this paper introduces Net-Preflight, a tool intended to indicate the state of the network between two nodes in an overlay, prior to performing large scale data transfers. We summarize our contributions as follows:

- We develop and present Net-Preflight, a custom, simple end-to-end, light-weight tool for measuring available throughput on a well provisioned network.
- We compare and verify Net-Preflight with Iperf focusing on throughput measurement accuracy, security, retentive

 ISBN: 978-1-61208-895-2

memory and performance issues, using different file transfers and utilizing isolated networks as well as the public internet network path for DTN-to-DTN testing.

Please note: the Net-Preflight tool is designed to help end-users sending bulk transfers and need a quick network health check to ensure quality of service. It is not designed to replace Iperf or Perfsonar tools, but rather network checks especially when root access at receiver end is not available.

In the last decade, the US Department of Energy experimental and observational user facilities have seen a huge increase in the amount of data transferred across its science network creating workflows that cross facility boundaries [4]. This has increased the complexity of users successfully scheduling big data transfers. Hence, this has necessitated the demand for a tool that is generalizable, light weight, flexible and guarantees quality-of-services. In addition, a tool that shows a prior end-to-end throughput measurement that is attainable by an application using systems located at different sites alongside delivering other metrics relevant to the bulk data transfers and saving the results in a retentive memory.

The rest of the paper is organized as follows: Section 2 presents our proposed approach. Section 3 presents the description of Net-Preflight alongside it's implementation and use cases. Section 4 describes the experimental evaluation and discussion of results. Section 5 presents some past related work and finally, Section 6 presents conclusion and future work.

II. Our Approach

We choose an experimental approach for our comparison using a dedicated isolated cloud network testbed and a public internet network respectively. We conducted several experiments using different configurations like single, parallel and concurrent bulk data transfers between two endpoints. Building on the method reported in [1]. The objective is to measure the end-to-end bulk transfer throughput metrics. We upload large files with several file sizes to the destination node using the Secure File Transfer Protocol (SFTP) because it has the ability to leverage a secure connection to transfer files and traverse the file system on both the local and remote system. We then initiate concurrent downloads of the large files over and over again and record the total duration for the files to be downloaded from the destination node to the source node. With this time and known file sizes or bytes downloaded, we calculate a relative aggregated throughput graph over time. We then repeat the same set of experiments using different TCP variants, then take the measurements and observe the metrics and performance under different conditions. We then provide a throughput comparison measurement of some widely known TCP congestion control algorithms variants with different data transfer rates namely: TCP Reno [5], Hamilton TCP [6], Binary Increase Congestion control (BIC) [7] and Cubic [8]. We exposed our simulated network to a range of congestion conditions and compared multiple TCP congestion control algorithms in order to investigate the behavior of the alternate congestion control algorithms under little to modest congestion, which should reveal any differences in user experience when large files are sent over between sources and receivers with high-speed network interfaces. Different algorithms respond differently to network loads, but are based on the same principle to avoid congestion. So, we investigate the different congestion control algorithms that are included as loadable modules in the Linux kernel as reported in [9].

A. TCP Throughput Measurement and Socket buffersize using SFTP

We perform TCP throughput measurement utilizing and varying the socket buffersize and comparing it with default settings. We consider a unidirectional TCP data transfer from the source socket S_n to destination node D_n. TCP uses window-based flow control, such that source is allowed to have up to a certain number of transmitted unacknowledged bytes, called window size W_s. Then, W_c is the sender's congestion window and W_r is the receive window advertised by the source node S_n, and B_s is the size of the send socket buffer at source node S_n. The destination window W_r is amount of available receive socket buffer memory at source socket S_n, and is limited by source socket buffersize B_r.

Bottleneck Conditions: A link is said to be a bottleneck when the current bandwidth is less than the available bandwidth. Also, a link could be non-congested when its packet loss rate due to congestion is practically zero or when the current link capacity or bandwidth equals the available bandwidth. For simplicity, we used traffic shaping [10] bandwidth management technique to reduce the bandwidth of the link at both endpoints. We did not consider packet loss during our measurements hence it is out of scope. Equation (1) and (2) are used to calculate the measured throughput, where, T_p = Measured Throughput (mbps), B = Bytes downloaded, ΔT = t_n - t_s, t_n = current time, t_s = time started.

$$T_p = \frac{B}{\Delta T} = \frac{B}{t_n - t_s} \tag{1}$$

$$T_p(bps) = \frac{\text{WinSize(bits)}}{\text{Latency(sec)}} = \frac{\text{Bandwidth(bps)} * \text{RTT}}{\text{Latency(sec)}} \tag{2}$$

III. Net-Preflight Tool Description and Use Cases

In this section, we present Net-Preflight Tool Description and explored different use cases both on isolated networks and the public internet network path for DTN-to-DTN testing. Please note that the source code and implementation of Net-Preflight are available at [11].

A. Net-Preflight Tool Definition and End-to-End requirements

Net-PreFlight is a simple end-to-end, light-weight tool for measuring available throughput, traceroute. Compared to existing tools like Iperf, It saves all the previous results and configuration in its memory. It doesn't require root access at the destination node, which is one of the main contributions of our tool and makes it Unique. In addition, It mainly focuses on accurate throughput measurement, flexibility, a retentive memory, security, and performance-related issues.

 ISBN: 978-1-61208-895-2

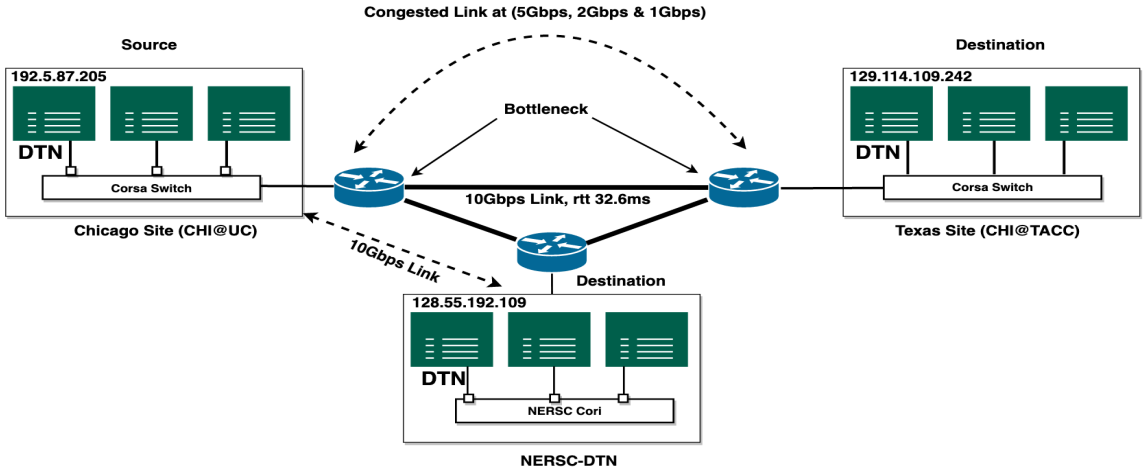


Fig. 1. Throughput measurement for large data transfer over isolated Network (CHI@UC to CHI@TACC) and public Network (NERSC DTN - CHI@UC DTN).

The main requirement of Net-Preflight is that you need to have a source and a destination IP, and both should be able to ping each other. Ideally, Net-Preflight was designed such that Root access is not required in both locations, but in this work, we have assumed we have files with different sizes saved in the destination node. To use the tool, you log in to your source node and run the command by specifying the required arguments, such as the destination IP address, the key file path, the target file, file size, and the number of iterations, respectively as stated in [11]. This operation initiates the download of the data from a destination to source over and over again as quickly as possible, then records how long it takes for the files to be downloaded alongside computing the difference. Using the time and known file size, a relative bandwidth graph is plotted over time, and the throughput is measured with respect to time.

B. Use Cases

1) Isolated Network (Network Setup between (CHI@UC to CHI@TACC): We present our first use case scenario, which is an isolated network over Chameleon testbed [12]. Chameleon is a large-scale, deeply re-configurable experimental platform, which is built to support Computer Sciences systems research with a wide variety of capabilities for researching systems, networking, distributed and cluster computing across multiple sites, such as the University of Chicago (CHI@UC) and Texas Computing Center (CHI@TACC), connected by a shared 100Gbps network. Figure 1 shows the network topology deployed on the isolated network over a dedicated 10Gbps bandwidth link. The goal of the experiment is download data from destination to source over and over again as quickly as possible, then record how long it takes for the files to be downloaded and compute the difference. Using the time and known file size, a relative bandwidth graph is plotted over time and the throughput is measured with respect to time.

2) Public Internet (Network Setup between (NERSC DTN - CHI@UC DTN): We present our second use case scenario experiment, which is a transfer from a DTN at the National Energy Research Scientific Computing Center (NERSC) to one of the nodes on the isolated network CHI@UC DTN. NERSC is a high performance computing (supercomputer) user facility operated by Lawrence Berkeley National Laboratory for the United States Department of Energy Office of Science. The DTN are NERSC servers dedicated to performing transfers between NERSC data storage resources such as HPSS and the NERSC Global File System (GFS), and storage resources at other sites. These nodes are being managed (and monitored for performance) as part of a collaborative effort between Energy Sciences Network (ESnet) and NERSC to enable high performance data movement over the high-bandwidth 100Gb ESnet wide-area network (WAN). All DTNs have two 100-gigabit ethernet links for outgoing connections and two 10-gigabit ethernet links to transfer to NERSC internal resources (HPSS). We repeat the same set of experiments similar to the isolated network scenario.

IV. EXPERIMENTAL EVALUATION

We conducted five experiments with respect to buffersize, file size, TCP congestion algorithm variant, limiting bandwidth and window size respectively.

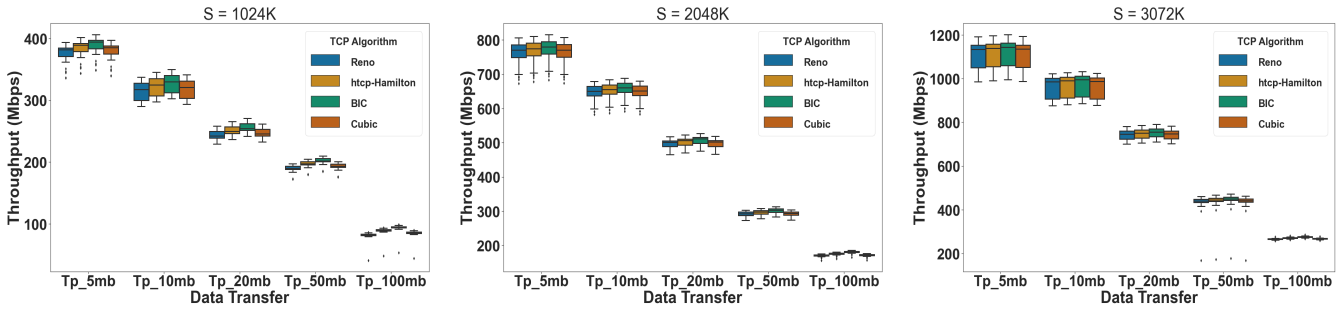
A. ***Experiment 1*** - *Measurement w.r.t Buffersize and Filesize at 10Gbps link Capacity (Isolated network):*

This experiment was setup with the aim of evaluating the performance of the tool when data is transferred using different file sizes and buffersize. It was set-up on an isolated network using a 10Gbps link capacity between source and destination node. We performed the experiment varying the file transfer sizes between 5MB - 100MB along side using different buffersizes respectively.

B. ***Experiment 2*** - *Measurement w.r.t different TCP congestion algorithm with 10Gbps link Capacity(Isolated network):*

In this experiment, our aim was to evaluate the performance of the tool using different TCP congestion algorithm along side

 ISBN: 978-1-61208-895-2



(a) (b) (c)

Fig. 2. Comparison of TCP congestion control algorithm comparison.

utilizing different file transfer sizes and buffersize. We started with Cubic, which is the current default TCP algorithm in most Linux operating systems, followed by Reno, Hamilton and BIC. The experiment was setup on an isolated network using a 10Gbps link capacity as shown in Figure 2.

C. ***Experiment 3*** - *Aggregated Throughput Measurement limiting bandwidth w.r.t congested vs Non-congested links (Isolated Network):*

In this experiment, we used the 10Gbps link capacity only. Several file transfers were tested at different congestion link capacity but keeping the buffersize constant. The aim was to introduce a bottleneck on the 10Gbps dedicated network link and observe the effect. The traffic at both source at destination node was limited using traffic shaping as reported in [10]. We start with the default link capacity of 10Gbps and then limit down to 5Gbps, 2Gbps and 1Gbps respectively as shown in Figure 3a.

D. ***Experiment 4*** - *Tool Comparison w.r.t Window size, Iperf vs Net-Preflight:*

Here, we tested measurements at window sizes from 8K - 1024k as shown in Figure 3b. We observed that there is a significant difference in network performance depending on the window size, with 128K window size being the most performed. This might be as result of the underlying background traffic condition, which resulted to a drop in throughput at 512K, 1024K and 2048K respectively.

E. ***Experiment 5*** - *Large file Aggregated Throughput Measurement w.r.t Concurrent transfers via Public Network from NERSC DTN - CHI@UC DTN:*

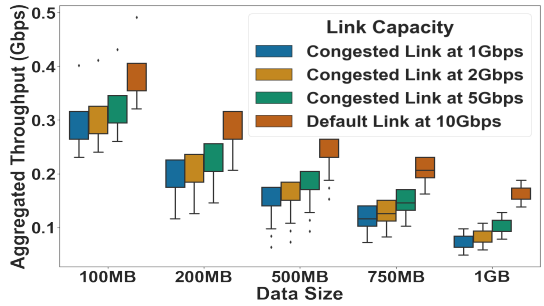
In this experiment, the aim was to evaluate the performance using huge concurrent file transfers with multiple streams (1, 2, 4, 6, and 8-Streams) with respect to 100MB, 200MB, 500MB 1GB and 2GB respectively (as shown in Figure 5). We show aggregated throughput keeping the buffersize constant and observing the effect of concurrent transfers on the aggregated throughput across a public network from NERSC DTN to CHI@UC DTN.

F. *Discussion of Results*

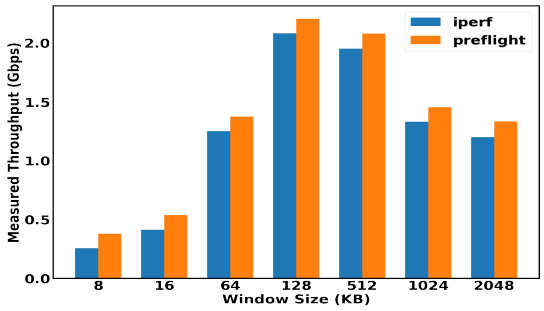
We collected throughput measurement for different data transfers sizes performed some experiments to observe the effect of different TCP congestion control variants on the throughput measurement. Figure 2 shows the comparison of various TCP congestion control algorithms. We conducted different experiments between buffersize S=512K and S=3072K while varying the data transfer files sizes between 5MB to 100MB, alongside comparing four TCP congestion control algorithm namely Reno, Hamilton, BIC, and Cubic, respectively. We observed that while keeping the buffersize constant at S=512K and S=1024K, a decrease in throughput was measured with increasing file size where BIC came out on top followed by Hamilton, BIC and Cubic. But as the buffersize increases between S=2048K and S=3072K, it was observed that it has less effect on the throughput measurement, which indicates that as we max out the buffersize, changes were not noticed between the TCP congestion control algorithms and hence becomes insensitive. However, on comparing all the four scenarios side-to-side, it was observed that an increase in the data transfer sizes results in a decrease in measured throughput while the opposite is true for the buffersize. But in all cases, BIC's performed best, followed by Hamilton and Cubic, then Reno.

Figure 3a shows a comparison of the congested link vs the non-congested link. We conducted this experiment in an isolated network between source node at Chicago and destination node and Texas. The default link capacity is 10Gbps, and several bulk file transfers were tested at different congestion link capacity but keeping the buffersize constant. The aim was to introduce some bottleneck on the network and observe the effect on the measure aggregated throughput. In order to introduce a bottleneck in the setup, the traffic at both source at destination node was limited using traffic shaping as reported in [10]. We started with the default link capacity of 10Gbps and then reduced it down to 5Gbps, 2Gbps, and 1Gbps, respectively. We observed that decreased in the link capacity result to a decrease in measure aggregated Throughput. A clear difference was observed in the measured throughput when the data transfer file was increased at different link capacities. The default link capacity of 10Gbps showed the highest throughput

 ISBN: 978-1-61208-895-2



(a)



(b)

Fig. 3. (a)Testing Congestion Link vs non-congested links (b) Window Size Iperf vs Net-Preflight comparison.

while the 1Gbp congested link capacity showed the least aggregated throughput, which shows a good performance of Net-Preflight measurement tool.

Figure 3b shows the comparison between iperf vs Net-Preflight at different window sizes respectively. We further compared the performance of the throughput measurement between Iperf and Net-preflight as shown in Figure 4, where we kept the file transfer sizes between 20MB to 100MB. We observed that in both scenarios, while we keep the data transfer sizes constant, an increase in buffersize results to an increase in throughput. Overall Net-Preflight provided measurements largely in agreement with established tools. Figure 5 shows results of the experiments using large data transfers using concurrent and multiple streams of transfers. In this experiment, the aim was to evaluate the performance using huge concurrent file transfers of 100MB, 200MB, 500MB 1GB and 2GB, respectively. We performed the experiments over 30 iterations. We show aggregated throughput keeping the buffersize constant and observing the effect of concurrent transfers on the aggregated throughput. This time we started with a large buffersize S = 2048K. We kept the buffersize constant and varied the file transfer size starting from 100MB to 2GB. We then compare different streams of transfers from 1-stream to 8 streams respectively. It was observed that as we increase the file sizes the aggregated throughput increases, however a change is only noticed between the 1-stream and 2-streams. But, between 4-stream, 6-stream and 8-streams a significant change was not noticed. In addition, we also observed that an increase in buffersize from 2048k to 4096k corresponds to an increase in total aggregated throughput. *Why does Net-Preflight perform similar to Iperf?* Our results show a throughput measurement that is comparable to Iperf, which requires further exploration, since Net-Preflight also includes storage I/O in addtion to network characteristics. Upon further investigation, it is found that because Chameleon architecture has been optimized for I/O writes, hence we do not see it affecting the Net-Preflight results. Further analysis on a wide range of network infrastructure will be conducted in future.

V. Related Work

Past research has explored tools for network measurement, monitoring, available throughput estimation and bandwidth measurement [13]–[15], as well as analysis of TCP throughput and socket buffer auto-sizing for high-performance data transfers [16], [17]. The authors developed a simple analytic characterization of the steady state throughput as a function of loss rate and Round Trip Time(RTT) for a bulk transfers [2]. For instance the authors in [16] proposed a technique called SOBAS, their strategy was based on automatic socket buffer sizing at the application layer. Their results show that SOBAS provides consistently a significant throughput increase compared to TCP transfers that use the maximum possible socket buffersize. In a similar scenario, while the authors in [18] analyzes the Incast problem, the authors in [19] examined the effects of using parallel TCP flows to improve end-to-end network performance for distributed data intensive applications. While the authors in [20] describes the analysis of TCP/IP socket buffer length in Local Area Network (LAN) and Wide Area Network (WAN), authors in [21] presented a MPT-GRE software, which is a two multipath communication systems based on different technologies. Following from various traditional approaches, a lot of existing network monitoring tools lack the capability of retaining all previous measurements, flexibility and security in terms of having root access to both ends. Hence this paper propose Net-Preflight, a tool intended to bridge the gap by indicating the state of the network between two nodes in an overlay, prior to initiating large data transfers.

VI. Conclusion and Future Work

This paper presents Net-Preflight, a tool intended to indicate the state of the network between two nodes in an overlay, prior to performing large scale data transfers. It was compared with existing tools using different metrics and utilizing isolated networks, like Chameleon testbed and public internet transfers. The comparison focused on throughput accuracy, flexibility and performance history, which stores previous configurations and network measurements. Net-Preflight addresses questions of how concurrent stream connections can improve aggregate TCP throughput measurement. It also addresses the question of how to select the maximum number of sockets necessary to maximize TCP throughput while simultaneously avoiding congestion. Our results show that the use of parallel TCP streams is equivalent to using a large maximum segment size on a single connection. In the future, we will continue to

 ISBN: 978-1-61208-895-2

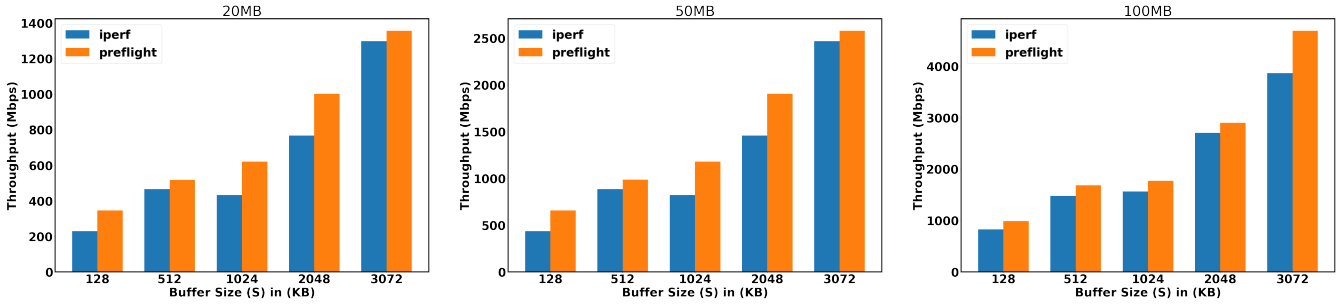


Fig. 4. Iperf and Net-Preflight data transfer comparison.

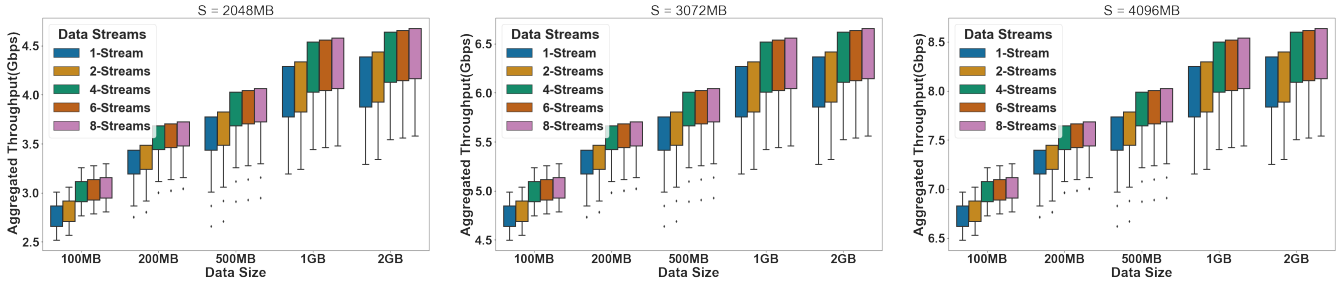


Fig. 5. Large file concurrent transfers.

improve the performance of the tool and explore the possibility of applying learning techniques to predict the future end-to-end throughput and latency that is attainable by the tool.

REFERENCES

[1] M. Allman, "Measuring end-to-end bulk transfer capacity," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 139–143, 2001.

[2] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 303–314, 1998.

[3] J. Zurawski *et al.*, "perfsonar: On-board diagnostics for big data," in *1st Workshop on Big Data and Science*, pp. 1–6, Citeseer, 2013.

[4] B. Enders, D. Bard, C. Snavely, L. Gerhardt, J. Lee, B. Totzke, K. Antypas, S. Byna, R. Cheema, S. Cholia, M. Day, A. Gaur, A. Greiner, T. Groves, M. Kiran, Q. Koziol, K. Rowland, C. Samuel, A. Selvarajan, A. Sim, D. Skinner, R. Thomas, and G. Torok, "Cross-facility science with the superfacility project at lbnl," in *2020 IEEE/ACM 2nd Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (XLOOP)*, pp. 1–7, 2020.

[5] Y. Nishida, "The newreno modification to tcp's fast recovery algorithm," *Standards Track, PP*, pp. 1–16, 2012.

[6] D. Leith and R. Shorten, "H-tcp: Tcp for high-speed and long-distance networks," in *Proceedings of PFLDnet*, vol. 2004, pp. 1–6, 2004.

[7] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (bic) for fast long-distance networks," in *IEEE INFOCOM 2004*, vol. 4, pp. 2514–2524, IEEE, 2004.

[8] S. Ha and I. Rhee, "Taming the elephants: New tcp slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011.

[9] A. Esterhuizen and A. Krzesinski, "Tcp congestion control comparison," *SATNAC, September*, pp. 1–6, 2012.

[10] P. Kanuparthy and C. Dovrolis, "Shaperprobe: end-to-end detection of isp traffic shaping using active methods," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 473–482, 2011.

[11] B. Mohammed, M. Kiran, and B. Enders, "Net-Preflight Github repository: https://github.com/esnet/netpreflight-performancetest." Accessed: 2021-09-26.

[12] K. Keahey *et al.*, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*, USENIX Association, July 2020.

[13] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 39–44, 2003.

[14] A. Tirumala, L. Cottrell, and T. Dunigan, "Measuring end-to-end bandwidth with iperf using web100," in *In Web100, Proc. of Passive and Active Measurement Workshop*, Citeseer, 2003.

[15] D. Kaur, B. Mohammed, and M. Kiran, "Netgraf: A collaborative network monitoring stack for network experimental testbeds," *arXiv preprint arXiv:2105.10326*, 2021.

[16] R. S. Prasad, M. Jain, and C. Dovrolis, "Socket buffer auto-sizing for high-performance data transfers," *Journal of GRID computing*, vol. 1, no. 4, pp. 361–376, 2003.

[17] M. Jain, R. S. Prasad, and C. Dovrolis, "The tcp bandwidth-delay product revisited: network buffering, cross traffic, and socket buffer auto-sizing," tech. rep., Georgia Institute of Technology, 2003.

[18] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan, "Measurement and analysis of tcp throughput collapse in cluster-based storage systems.," in *FAST*, vol. 8, pp. 1–14, 2008.

[19] T. J. Hacker, B. D. Athey, and B. Noble, "The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network," in *Proceedings 16th International Parallel and Distributed Processing Symposium*, pp. 10–pp, IEEE, 2002.

[20] L. Mazalan, S. S. S. Hamdan, N. Masudi, H. Hashim, R. Abd Rahman, N. M. Tahir, N. M. Zaini, R. Rosli, and H. A. Omar, "Throughput analysis of lan and wan network based on socket buffer length using jperf," in *2013 IEEE International Conference on Control System, Computing and Engineering*, pp. 621–625, IEEE, 2013.

[21] S. Szilágyi, F. Fejes, and R. Katona, "Throughput performance comparison of mpt-gre and mptcp in the fast ethernet ipv4/ipv6 environment," *Journal of Telecommunications and Information Technology*, 2018.

 ISBN: 978-1-61208-895-2