Multiple Use-Case Support in a Parallel Data Stream Layer for WANs

Nooshin Eghbal
Dept. of Computing Science
University of Alberta

Paul Lu
Dept. of Computing Science
University of Alberta

Abstract

A variety of use-cases and workloads for wide-area networks (WAN) can benefit from the use of parallel data streams to maximize utilization and throughput.[1] For bulk-data transfers, high bandwidth-delay-product (BDP) networks with even small amounts of packet loss (e.g., due to congestion) can suffer reduced TCP/IP throughput. Some commonly used tools, such as `rsync` and `git`, are layered on standard-in/-out streams. Other protocols, including Virtual Network Computing (VNC) and distributed file systems, use port forwarding.

With these motivations, we design, implement, and evaluate the open-source Parallel Data Streams (PDS) user-level tool. PDS makes trade-offs (e.g., TCP fairness) for specific workloads, when performance is the key goal. The main contribution of the PDS tool is the additional functionality and support for more protocols and workloads, beyond the well-known GridFTP tool. We also quantify and contribute a performance evaluation of PDS, using a combination of emulated and real WANs. We establish that PDS achieves comparable performance to GridFTP on comparable workloads (e.g., file transfer), but additional functionality via other tools (e.g., `rsync`, Network File System). For example, PDS can transfer a 14 GB file on a WAN between Alberta and Quebec (maximum 1 Gbps; over 3,100 km) at 861 Mbps, using `rsync` and 8 parallel TCP cleartext streams.

1 Introduction

Many researchers around the world collaborate by transferring large data sets. For example, the Square Kilometre Array (SKA) radio telescope project will generate data at around 400 Tbps, which needs to be sent to a supercomputer center 1,000 km away for analysis [13]. Similarly, the Large Hadron Collider generates terabytes-to-petabytes of data for each experiment that have to be transferred around the world for analysis.

Today, cloud computing sometimes requires dozens of gigabytes-to-terabytes to be transfered between local and cloud storage. Shared networks are common and the contention from other traffic can have a large impact on performance. When allowed by service-level agreements (SLA), and for short periods of time, it is useful to have a tool that can (unfairly and) aggressively utilize the available bandwidth. Furthermore, beyond Virtual Private Networks (VPNs) and similar technologies, it is useful to have a non-priviledged, user-level tool that supports large-data transfers, multiple other tools, and multiple workloads. Therefore, we developed the open-source Parallel Data Streams (PDS) tool.

An alpha version of PDS is available at `https://github.com/paullu-ualberta/paralleldatastream`

Wide-area networks (WANs) are one example of high bandwidth-delay-product (BDP) networks. In WANs, large geographic distances (e.g., thousands of kilometers) and many middleboxes (e.g., cumulative latency, possibility of queuing delays, and congestion), contribute to large round trip times (RTT). For example, the NSF TeraGrid network has 30 Gbps links between clusters at six sites, with 60 ms RTT [6]. The Canadian CANARIE research network has 51 ms of RTT between the University of Alberta and Sherbrooke, Quebec (Section 2.1). Non-research networks and paths that span commercial Internet service providers (ISPs) can have even higher RTTs, in our experience (e.g., Edmonton to Halifax varied between 71 and 146 ms during one *ad hoc* experiment). With large RTTs, even a small packet-loss rate can impact protocols like Transmission Control Protocol (TCP) and make the high-bandwidth utilization a challenging task.

Consequently, GridFTP [6] is a widely used, parallel TCP-connection file transfer tool, particularly in computational science. To avoid underutilizing research net-

[1]This paper Copyright 2017. Under submission.

works like TeraGrid and CANARIE, users accept the use of parallel TCP connection tools which are unfair by design. Ultimately, fairness and sharing can be achieved by other mechanisms, such as SLAs and resource scheduling (an area of future work for PDS). For the rest of this paper, we set aside the policy aspects of parallel data streams, to focus on the design, implementation, and evaluation of the PDS tool as a mechanism to support more use-cases than GridFTP.

Specifically, general client-server applications have request-response messages interleaved with bulk-data transfer. For example, common tools such as `rsync` (i.e., file synchronization tool) and the Network File Systems (NFS) (a DFS) are not suitable for the passive file transfers performed by GridFTP. The contribution of our new PDS tool is that it is designed to support multiple tools, workloads, and use-cases. The use of parallel TCP data streams in PDS is inspired by GridFTP, and we empirically show that PDS's performance is comparable for file transfers. Also, using both emulated and real WANs, we show that PDS can support general protocols like the Network File System (NFS) for specific use-cases with high performance.

Without modifying the underlying transport protocols, using parallel TCP connections is an effective and well-known method to improve bandwidth utilization [6]. The intuition behind this method is that when a packet gets lost in one of these parallel streams, the congestion window of that stream will be reduced but there are still other streams with large congestion windows that can keep the aggregated throughput of the link high. Parallel data streams are the basis for both GridFTP and PDS.

2 Use-Case: rsync over PDS

`rsync` is an example of an tool that requires request-response messages for metadata and control (beyond just the FTP protocol). `rysnc` is often used to synchronize a hierarchy of files across a WAN, where the files might be large, but the file deltas between mirror copies can vary in size between small to the entire file. For example, logs, sensor data, map-reduce pipelines, and incremental checkpoints can have append-write access patterns which are easily synchronized using file deltas. `rsync` can compute and transfer only the file deltas, potentially saving a large amount of the bandwidth required to synchronize the contents of two file. GridFTP does not support `rsync` nor file-delta-only use-cases. GridFTP does support a mode where a file is only transferred if it is different, but a file delta of even a single byte will require the whole file to be transferred.

In Figure 1, if we have two copies of a file at nodes S (Source) and D (Destination) and we modify the one located at S, we can update D's copy by running `rsync`.

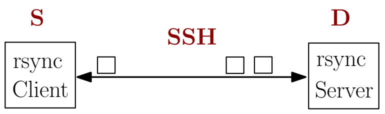


Figure 1: Current Use-Case: `rsync` over SSH (single TCP stream)

Then S and D negotiate with each other until S knows which parts of the file must be sent to D (delta). During this process, the data sent between S and D is not in the form of files, it is just a stream of data packets and messages. Therefore, a file transmission tool like GridFTP cannot be used.

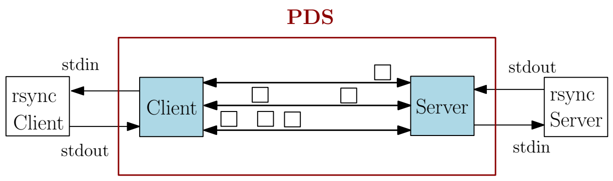


Figure 2: High-Performance Use-Case: `rsync` over PDS

Commonly, `rsync` runs over Secure Shell (SSH) with one TCP stream, which has low throughput on lossy, high-BDP networks (Figure 1). Figure 2 shows a general view of how a program like `rsync` runs over PDS. PDS has a Client and Server. To run `rsync` over PDS, we set the `rsh` option of the `rsync` command to the PDS Client executable file with proper command-line arguments explained below.

PDS's command-line arguments will (eventually) be a drop-in replacement for SSH in all applications with an `rsync`-like `rsh` option. Currently, there are three command-line arguments for the PDS Client program: 1) number of parallel streams, 2) block size and 3) connection type. PDS divides data into blocks in order to stripe them over parallel streams. The block size argument is for the maximum size of these blocks. The connection type argument can be either TCP (for maximum speed, but cleartext) or SSH but we only bring the results of parallel TCP streams in this paper.

For example, a client-side `rsync` command to synchronize `file1` to server `PDS-Server-IP`, where the name of PDS Client executable file is `PDS_client`, the number of TCP streams is 10, and the block size is 10000 bytes, is:

```
rsync --rsh="./PDS_client 10 10000 TCP"
file1 PDS-Server-IP:file2
```

2.1 Evaluation: rsync

Both GridFTP and `rsync` over PDS can be used to transfer entire files efficiently. And, if the file deltas are small, `rsync` over PDS would have a greater performance advantage over GridFTP, but there is no widely accepted benchmark for file-delta transfers. Therefore, the goal of this section is to establish (1) the comparable performance of GridFTP and `rsync` over PDS for whole file transfers, (2) the substantial performance advantage of `rsync` over PDS versus `rsync` over single-stream SSH. In addition to downloading and trying PDS directly, we wish to provide evidence that performance is a reason to consider using `rsync` over PDS. We evaluated `rsync` over PDS on an emulated WAN with different configurations, and also a WAN between two virtual machine (VM) nodes located at Quebec and Alberta in Canada (Figure 4). We also used GridFTP and UDT [9]. UDT is a reliable UDP-based tool.

Emulated WAN: To emulate a network with high RTT and packet-loss rate on a local-area network (LAN), we used the Netem-tc Linux tool [4] on a cluster between two nodes with 2.25 GHz AMD Opterons, 8-cores in total, 32 GB of memory, and with 1 Gbps bandwidth. Table 1 contains the version of the tested tools on the cluster nodes.

	Linux kernel	GridFTP	OpenSSH	rsync	UDT
version	2.6.32	8.6	6.3p1-hpn14v2	3.1.1	4

Table 1: Version of different tools on the cluster nodes.

We studied the effect of three parameters on the throughput of data transmissions in our emulated WAN: 1) RTT, 2) packet-loss rate, 3) number of parallel streams. In Figure 3, we report the throughput results of varying each of these parameters between two nodes of the cluster while the other two parameters are fixed. We repeat each test 5 times and report the averages with the standard deviation (SD) (very small bars).

For all three parameters (RTT, loss rate, and number of streams), the throughput of PDS is comparable to GridFTP. The performance of single-stream SSH and UDT are significantly lower.

AB-QC WAN: We experiment with PDS on a research WAN between Alberta and Quebec (AB-QC) provided by CANARIE [1], Cybera [2] and RISQ [5], spanning over 3,100 km geographically. The typical RTT is 51 ms and bandwidth is limited to 1 Gbps by the LAN (and measured by netperf), despite the WAN supporting higher speeds. We plan to experiment with higher WAN bandwidths in the future.

The research WAN is shared, although with access limited to just the research community, so our experiments are both more representative of real scenarios and less controlled than on our emulated WAN. Unfortunately, we do not know the exact packet-loss rate experienced during our experiments. The software on the Quebec and Alberta nodes are the same versions as our cluster nodes (Table 1), except the Linux kernel version is 3.11.0. Furthermore, our experimental nodes in Quebec and Alberta are actually virtual machines (VM) running under OpenStack. In terms of CPU power, our VMs are relatively modest, which may have an impact on some performance numbers (more below).

Figure 4 presents the average throughput results over 10 runs of transferring a 14 GB file from Alberta to Quebec for different number of parallel streams.

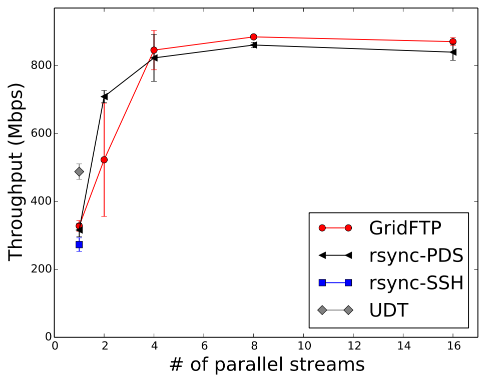


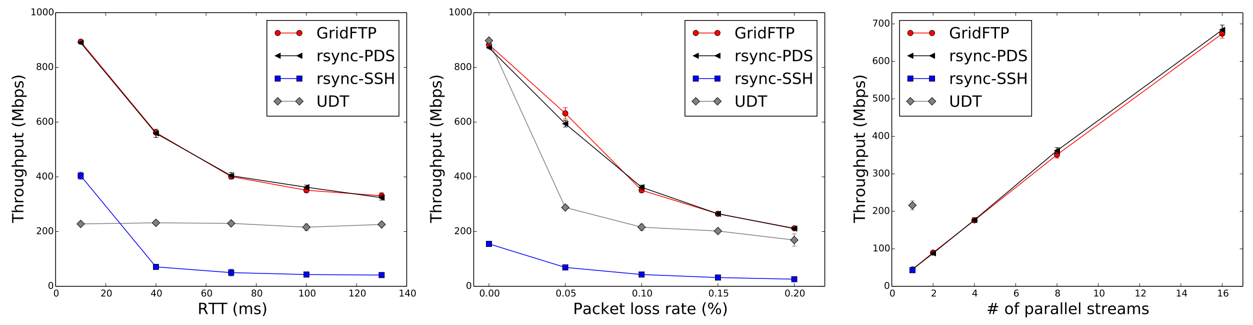
Figure 4: Throughput results (Mbps) for different number of parallel streams in **AB-QC** WAN.

In Figure 4, the throughput of both GridFTP and PDS decrease after a certain number of parallel streams (i.e., between 8 and 16 for GridFTP and PDS). Parallel streams and connections can cause contention among themselves. As with the emulated WAN, the throughput of `rsync` over PDS (i.e., 861 Mbps with 8 parallel TCP connections) is much higher than a single SSH stream (i.e., 274 Mbps).

Our main conclusion is that `rsync` over parallel PDS on a real WAN is comparable to GridFTP, and PDS shows a great improvement compared to a **single** SSH stream. And, we conclude that UDT has lower throughout than both GridFTP and PDS in most of these tests.

3 NFS over PDS

NFS was never designed for WANs and we are not advocating the general use of NFS over WANs. NFS is still an interesting (partly because of familiarity) workload to evaluate the performance benefits and limitations of PDS. Interestingly, with PDS, it becomes practical to



(a) 0.1% packet-loss rate and 8 streams. (b) 100 ms RTT and 8 streams. (c) 0.1% packet-loss rate and 100 ms RTT.

Figure 3: Throughput results (Mbps) in an **emulated** WAN with 1 Gbps bandwidth.

consider using NFS in a very specific use-case: moving whole files across WANs via a file system interface.

Port forwarding, as popularized by SSH, is a well-known technique to support network-based applications. Not all applications support the standard in/out layering model used by `rsync` and `git`. Existing scripts and applications must be modified to invoke the GridFTP application. By using `NFS` or a different DFS, the file access API is supported and any tool that uses, say, `open()` or `cp` works transparently. For future work, we intend to explore the combination of PDS and DFSes like Lustre [3, 14, 11]. However, we can currently report on the performance of `NFS` over PDS, through port forwarding.

We evaluated the performance of running `NFS` over PDS for both write and read operations. We used the standard `cp` command to transfer a 14 GB file from `NFS` client to server (write operation) and `NFS` server to client (read operation). Experiments using the standard Linux `dd` command were comparable to `cp` and are not discussed here.

Similar to `rsync` results, we studied the effect of the following three parameters on the throughput of `NFS` in our emulated WAN: 1) RTT, 2) packet-loss rate, 3) number of parallel streams. In Figure 5 we report the results of varying each of these parameters between two nodes of the cluster while the other two parameters are fixed for read operation. We repeat each test 5 times and report the averages with the standard deviation (SD). Also, the results of write operation are presented in Figure 6.

For both read and write operations, increasing RTT and packet-loss rate has a huge negative impact on the throughput of `NFS` over either TCP or PDS, as we expected. However, using PDS we can improve the performance significantly. For example, PDS achieves a throughput of 133.92 Mbps for the read operation using 8 parallel TCP connections when we set 0.1% packet-loss rate and 100 ms RTT in the network. Using the same settings, the throughput of NFS over single stream TCP is only 34.70 Mbps.

Comparing the results of read and write operations shows that PDS can achieve better performance for write operations. For example, PDS with 8 TCP streams is about 8.9 faster than TCP for write operation when the packet-loss rate is 0.1% and RTT is 100 ms. However, PDS can achieve only about 3.9 speed-up for read operation under the same network settings.

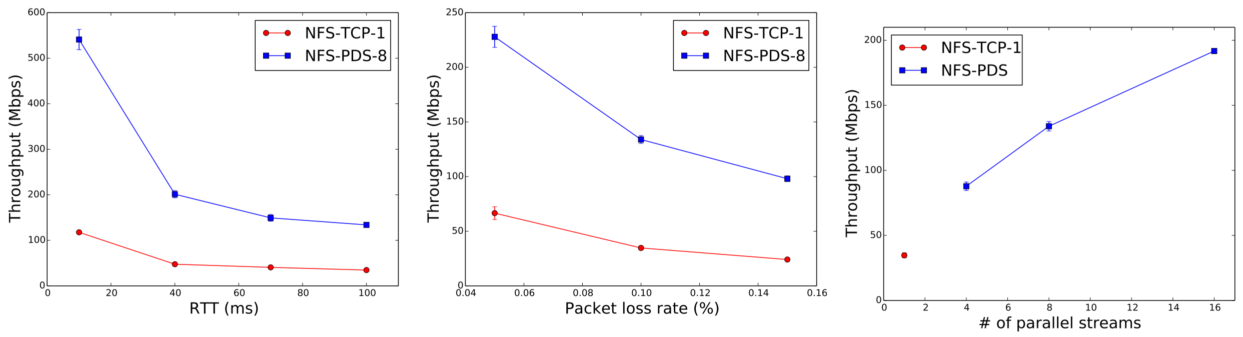
One possible reason for the higher write performance is that various pipelining and write-caching strategies are at play within the software stack and within `NFS` itself. In contrast, for reads, the request-response `NFS` message suffers from the impact of the RTT, despite the possibility of prefetching.

4 Related Work

The literature on high-performance, large-data transfers on WANs is extensive and mostly beyond the scope of our tool-based paper. However, we attempt a brief survey.

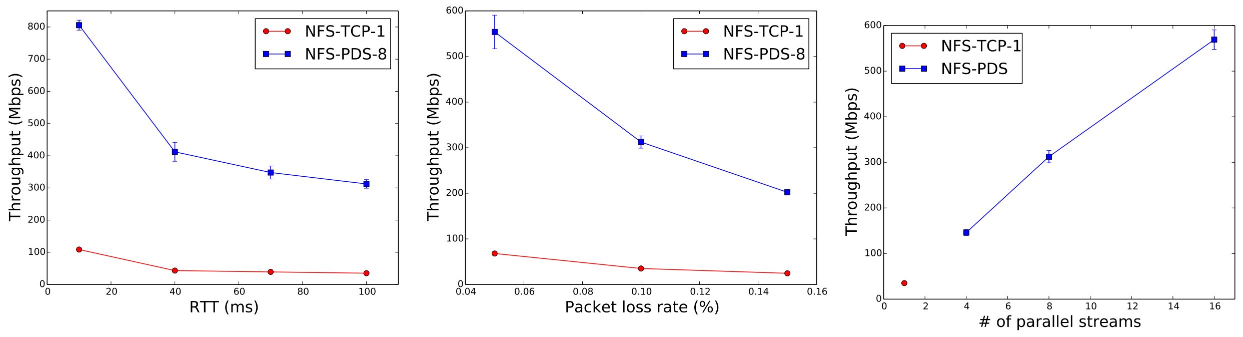
TCP-based Tools: GridFTP is the most well-known and widely used TCP-based tool [6]. It extends File Transfer Protocol (FTP) to provide a secure, reliable, and high throughput file transfer tool based on parallel TCP connections.

UDP-based Tools: The most important challenge of using UDP for large data transmissions in WANs is that it is not a reliable protocol. It means that there is no guarantee that the data is completely received at the destination. To solve this, some researchers work on implementing UDP-based tools such as UDT [9] and Reliable Blast UDP (RBUDP) [10] that support reliability at application layer. Our experiments (Section 2.1) show that UDT throughput is lower than the throughput of parallel TCP connections tools. Notably, for PDS, SSH, GridFTP, and UDT, the parameter space for command-line arguments and optimizations have not been explored yet.



(a) 0.1% packet-loss rate, 8 TCP streams for PDS, and 1 TCP stream. (b) 100 ms RTT, 8 TCP streams for PDS, and 1 TCP stream. (c) 0.1% packet-loss rate and 100 ms RTT.

Figure 5: Throughput results (Mbps) of **read** operation (using `cp`) in an emulated WAN with 1 Gbps bandwidth.



(a) 0.1% packet-loss rate, 8 TCP streams for PDS, and 1 TCP stream. (b) 100 ms RTT, 8 TCP streams for PDS, and 1 TCP stream. (c) 0.1% packet-loss rate and 100 ms RTT.

Figure 6: Throughput results (Mbps) of **write** operation (using `cp`) in an emulated WAN with 1 Gbps bandwidth.

Transport layer: High performance TCP variants: The other approach to solve TCP underutilization problem is modifying TCP itself. This approach has been addressed by many research papers and several high-performance TCP variants were proposed such as TCP NewReno, High-speed TCP, BIC-TCP, Compound TCP, CUBIC TCP [7], and TCP Proportional Rate Reduction [8]. Modifying TCP is effective and at the same time risky because of the possible compatibility problems with the middleboxes along the path.

5 Concluding Remarks and Future Work

As a tools paper, our main contribution is in describing the PDS platform for important existing workloads (e.g., data transfer) and for new workloads on WANs. But, our contributions also include a performance evaluation that establishes the appropriateness of PDS for a variety of use-cases and for a spectrum of RTTs and loss rates.

For example, via `rsync` over PDS, we can efficiently synchronize files that have read-mostly-write-seldomly or append-write access patterns without having to transfer the entire file. Applications such as map-reduce files and sensor logs have these properties. File transfer tools, such as GridFTP, are not as efficient for these use-cases.

Furthermore, with the new port forwarding feature of PDS, it is possible to support a wide variety of new tools, such as Virtual Network Computing (VNC) and connecting different clouds (e.g., OpenStack regions or cells across WANs; future work). Within our group, we have already demonstrated that VNC works across PDS (but not discussed here, for reasons of space). Also, specialized file-storage systems, like the Hadoop Distributed File System (HDFS) [12], are targeting streaming-data applications (producer-consumer-like write-only or read-only files per use, rarely concurrent read-write to the same file). Through port forwarding, it should be possible (in theory; future work) to direct HDFS to send the data over the PDS system.

With the ability of PDS to support standard-in/-out layering (rsync, git) and port forwarding, several new use-cases become possible, outside of simple file transfers and computational science.

References

[1] Canarie. `www.canarie.ca`. Accessed: 2015-10-29.

[2] Cybera. `www.cybera.ca`. Accessed: 2015-19-23.

[3] Lustre. `http://www.lustre.org`.

[4] Netem. `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`. Accessed: 2015-10-16.

[5] Risq. `http://www.risq.quebec`.

[6] ALLCOCK, W., BRESNAHAN, J., KETTIMUTHU, R., LINK, M., DUMITRESCU, C., RAICU, I., AND FOSTER, I. The Globus striped GridFTP framework and server. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing* (2005), IEEE Computer Society, p. 54.

[7] ALRSHAH, M. A., OTHMAN, M., ALI, B., AND HANAPI, Z. M. Comparative study of high-speed Linux TCP variants over high-BDP networks. *Journal of Network and Computer Applications 43* (2014), 66–75.

[8] DUKKIPATI, N., MATHIS, M., CHENG, Y., AND GHOBADI, M. Proportional rate reduction for TCP. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 155–170.

[9] GU, Y., AND GROSSMAN, R. UDTv4: Improvements in performance and usability. In *Networks for Grid Applications*. Springer, 2009, pp. 9–23.

[10] HE, E., LEIGH, J., YU, O., AND DEFANTI, T. A. Reliable blast UDP: Predictable high performance bulk data transfer. In *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on* (2002), IEEE, pp. 317–324.

[11] KOUTOUPIS, P. The Lustre distributed filesystem. *Linux Journal 2011*, 210 (2011), 3.

[12] SHVACHKO, K., KUANG, H., RADIA, S., AND CHANSLER, R. The Hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)* (2010), IEEE, pp. 1–10.

[13] TIERNEY, B., KISSEL, E., SWANY, M., AND POUYOUL, E. Efficient data transfer protocols for big data. In *E-Science (e-Science), 2012 IEEE 8th International Conference on* (2012), IEEE, pp. 1–9.

[14] WANG, F., ORAL, S., SHIPMAN, G., DROKIN, O., WANG, T., AND HUANG, I. Understanding Lustre filesystem internals. *Oak Ridge National Laboratory, National Center for Computational Sciences, Tech. Rep* (2009).