Comparative Performance Evaluation of High-performance Data Transfer Tools

Deepak Nadig*, Eun-Sung Jung†, Rajkumar Kettimuthu‡, Ian Foster‡§, Nageswara S.V. Rao¶, Byrav Ramamurthy*
*Dept. of Computer Science & Engineering, University of Nebraska-Lincoln, USA
†Dept. of Computer and Information Communication, Hongik University, South Korea
‡Data Science and Learning Division, Argonne National Laboratory, USA
§Dept. of Computer Science, The University of Chicago, USA
¶Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA

***Abstract*—Data transfer in wide-area networks has been long studied in different contexts, from data sharing among data centers to online access to scientific data. Many software tools and platforms have been developed to facilitate easy, reliable, fast, and secure data transfer over wide area networks, such as GridFTP, FDT, bbcp, mdtmFTP, and XDD. However, few studies have shown the full capabilities of existing data transfer tools from the perspective of whether such tools have fully adopted state-of-the-art techniques through meticulous comparative evaluations. In this paper, we evaluate the performance of the four high-performance data transfer tools (GridFTP, FDT, mdtmFTP, and XDD) in various environments. Our evaluation suggests that each tool has strengths and weaknesses. FDT and GridFTP perform consistently in diverse environments. XDD and mdtmFTP show improved performance in limited environments and datasets during our evaluation. Unlike other studies on data transfer tools, we also evaluate the predictability of the tools' performance, an important factor for scheduling different stages of science workflows. Performance predictability also helps in (auto)tuning the configurable parameters of the data transfer tool. We apply statistical learning techniques such as linear/polynomial regression, and k-nearest neighbors (kNN), to assess the performance predictability of each tool using its control parameters. Our results show that we can achieve good prediction performance for GridFTP and mdtmFTP using linear regression and kNN, respectively.**

I. Introduction

Data transfer in wide-area networks has been long studied in different contexts including data sharing among data centers managed by a single organization (e.g., synchronization among data centers [1, 2]), and online access to scientific data [3–6]. Many software tools and platforms have been developed to facilitate easy, reliable, fast, and secure data transfer over wide area networks, such as GridFTP [7], FDT [8], bbcp [9], mdtmFTP [10], and XDD [11].

Faster data transfers in e-Science will ultimately result in faster knowledge discovery. In this regard, high-performance data transfer has been studied in various aspects since the data transfer operation involves many system components along which the data path lies. The following areas have been studied towards the performance improvement of high-performance data transfers over wide area networks (WANs).

- Storage systems: How can we read/write a dataset in the fastest way possible?
- Networks: How can we transfer data to/from memory over WANs? What network parameters can be optimized?
- Protocols: What protocol optimizations yield the best transfer performance?
- Tool/platform: How can data transfer tools and platforms coordinate the above components to optimize performance by adapting to diverse use cases?

High-performance parallel file systems (PFS) such as Lustre and General Parallel File System (GPFS) by IBM are deployed in supercomputing facilities. The complex structure of these filesystems require sophisticated parameter tuning to utilize the storage system to its full capacity [12, 13]. Similar tuning is also required for the network protocols (e.g. TCP/IP protocol stack based on static/dynamic network configuration [14]) and the underlying networks to exploit multiple alternative paths. Several data transfer tools have modified the traditional FTP protocol to improve performance in the context of specific environments or data transfer workloads. For example, GridFTP builds on FTP to improve its performance, reliability and security. An important issue with data transfer tools/platforms is the orchestration of storage, network and protocols, and their holistic optimization [15]. Studies have also focused on the above problems to analyze the performance of both the isolated components [16] and the end-to-end systems [17]. However, few studies have shown the full capabilities of existing data transfer tools from the perspective of whether such tools have fully adopted state-of-the-art techniques through meticulous comparative evaluations.

The main contributions of this paper are as follows:

1) We present a comprehensive comparative performance evaluation of various existing data transfer tools.
2) For each data transfer tool, we outline all parameters that are critical to data transfer performance by classifying them into appropriate categories.
3) We developed a number of test datasets to accurately capture the real-world performance of the data transfer tools, and to analyze their strengths and weaknesses.
4) We perform extensive real-world data transfers on various geographically separated supercomputing facilities over the wide area network and discuss performance

978-1-5386-8134-3/18/$31.00 ©2018 IEEE

predictability based on the tools' control parameters.

The rest of this paper is as follows. Section II outlines the related work. Section III presents a brief description of various data transfer tools that are evaluated in our study. Section IV presents our experimental methodology and experimental results under various parameters critical to data transfer performance. We also discuss the experimental results and their implications for the design of high-performance data transfer tools. In Section V, we discuss performance predictability based on the tools' control parameters. We present a discussion on the strengths and weaknesses of the tools compared in Section VI. Finally, we conclude in Section VII.

II. Related Work

Many high-performance data transfer tools have been developed. GridFTP [7], FDT [8], BBCP [9], mdtmFTP [10], and XDD [11, 18] are representative examples. Advanced services such as Globus [19], PhEDEx [20], and LIGO Data Replicator [6] are currently provided. These tools and services are built upon many years of performance improvement endeavors. For example, in the case of GridFTP, these include the use of parallelism for high-speed data transfer [21], a layered software architecture to adapt to various working environments [22], and *pipelining* [23] and *popen* [24] to address the lots of small files (LOSF) problem.

Most previous studies of data transfer performance have focused on evaluating a single tool or feature [25]; few have involved comparative evaluations of multiple data transfer tools. Mattman et al. [26] classified data transfer tools along seven dimensions (scalability, reliability, easy of use, transfer rate, cost of operate, cost to implement, and industry adoption), but that study was performed more than a decade ago and considered mostly older tools. Zhang et al. recently evaluated the performance of the mdtmFTP data transfer tool [27]. Our work here is distinguished by its extensive comparative evaluation of four representative data transfer tools, namely FDT, GridFTP, mdtmFTP, and XDD.

III. Data Transfer Software Architecture

We first describe the major features of the data transfer tools from a performance perspective. Table I summarizes the features of the various tools in terms of storage, network, protocol, and their software architecture. For storage, there are three sub-features; (1) Direct I/O, (2) Parallel file system (PFS) multi-threading, and (3) PFS striping. Direct I/O indicates whether a tool uses the direct I/O function for file reads/writes (file reads and writes go directly between the tool and the storage system, bypassing the operating system read and write caches). PFS multi-threading and striping indicate whether a tool exploits multiple threads and file striping over multiple disks on PFS, respectively. Regarding the network, there are two sub-features: (1) multi-thread and (2) multi-socket, which indicate whether a tool deploys multi-threads for network I/O and multiple TCP sockets to achieve high performance, respectively. At the protocol level, we consider one sub-feature, namely, the support for lots of small files (LOSF) transfers. This indicates whether a tool has implemented an optimization to improve the performance for LOSF transfers. The software architecture indicates any distinguishable characteristics of a tool in terms of its design architecture.

A. Fast Data Transfer (FDT)

This tool uses a managed pool of TCP socket buffers to enable continuous streaming of a dataset between endpoints [8]. FDT ensures fast and efficient data transfers by using independent read/write threads, appropriate corresponding storage I/O and network buffer sizes. On the receiver endpoint, the received dataset (or files) are recreated asynchronously from the managed buffer pool. For large datasets, FDT can be used to stream the data continuously between the endpoints without requiring the need for restarting network transfers between files.

B. GridFTP

GridFTP [7] is a high-performance protocol optimized for secure and reliable data transfers across high-bandwidth WAN. The Globus implementation of GridFTP supports parallel data transfers using multiple TCP streams with support for striping and interleaving. It incorporates a number of features for secure, reliable and high-performance data transfer including third-party data transfer controls, automatic renegotiation of TCP buffers and window sizes, user authentication, data integrity/confidentiality controls, and support for checkpointing and connection restarts.

C. mdtmFTP

This tool uses dedicated I/O threads for both disk and network operations [10]. It uses a pipelined I/O-centric architecture and multi-core services provided by the multicore-aware data transfer middleware (MDTM) [28] for fast and efficient file transfers. The MDTM services are combined with other features like pipelining, batch-processing, managed buffer pools, zero-copy, and asynchronous I/O operations to optimize data transfer performance. Further, mdtmFTP employs a "virtual file" mechanism for handling LOSF transfers. A large virtual file is created by sequentially adding all the artifacts of the dataset along with a content index table. This virtual file is restored at the destination endpoint by using the metadata information contained in the content index table.

D. eXtreme DD Toolset (XDD)

This tool creates and uses a set of threads at both the source and the destination endpoints when a file transfer is initiated [11]. A number of *QThreads* [29] read from a thread-local buffer (the size of which is determined by the *request size* parameter) and source XDD process creates a *TargetThread* which initiates a corresponding connection with the destination XDD process. A file transfer is initiated by creating a set of source/destination paired processes. For accomplishing an end-to-end data transfer task, the XDD instance uses the same number of ports, the same I/O request sizes and queue depths at both endpoints. Thus, XDD requires a pair of "matched" instances for each file transfer.

TABLE I: Comparison of Software Architecture of Different Data Transfer Tools

	Storage			Network		Protocol	SW Arch.
	Direct I/O	PFS multi-thread	PFS striping	multi-thread	multi-socket	LOSF Approach	
GridFTP	Y	Y	Y	Y	Y	Concurrency, Pipelining	Layered
XDD	Y	Y	Y	Y	Y	NA	Not layered
FDT	N	Y	N	Y	Y	Managed buffer pool, streaming	Not layered
mdtmFTP	Y	Y	N	Y	Y	Virtual fileystem	Unknown

IV. Experimental Evaluation

We evaluate both local area network (LAN) and wide area network (WAN) disk-to-disk transfers over high-speed network connections. While LAN transfer performance was evaluated on the cloud computing infrastructure, WAN transfer performance evaluation on the other hand was conducted between different supercomputing center pairs with different round-trip-time (RTT) latencies. Dedicated data transfer nodes (DTNs), high-performance parallel file systems, and software components including storage I/O and network transport modules were provided by the supercomputing sites.

A. Experimental Setup

The LAN transfers were performed over the cloud computing resources provided by the Holland Computing Center (HCC) at University of Nebraska-Lincoln (UNL). The nodes were configured to use 4 cores, 16GB RAM, 160GB of Ceph storage and 10GbE network interfaces. The transfer performance for the sub-1 ms RTT case was evaluated on this setup. For WAN transfers, the data transfer nodes (DTNs) at the following supercomputing centers were employed: a) Gordon at San Diego Supercomputer Center (SDSC), b) Bridges at Pittsburgh Supercomputing Center (PSC), c) Research Computing Center (RCC) at University of Chicago, and d) HCC at UNL. We consider RTTs of approximately 15 ms, 30 ms, and 60 ms, corresponding to the supercomputing center pairs RCC-to-PSC, UNL-to-SDSC and SDSC-to-PSC respectively. Each DTN provided a 10GbE network connection for the different data transfer measurements. The disk-to-disk file transfer performance was measured over the GPFS/Lustre filesystems mounted over an Infiniband network at each compute center.

B. Transfer Tools

We evaluated the performance of the four high performance data transfer tools described in Section III namely: GridFTP, FDT, mdtmFTP, and XDD. Of these tools, GridFTP was set up for third-party transfers, while the other tools operated in client-server mode. The mdtmFTP tools were distributed as Docker [30] containers and RPM packages, and required superuser privileges to set up on the DTNs. Therefore, we present mdtmFTP performance measurements for the LAN scenario only and do not incorporate it in our WAN evaluations as none of the production DTNs provided this support. Also, we were unable to build an earlier source distribution of mdtmFTP due to the lack of support for its dependent libraries on the production DTNs. We do not include bbcp transfer performance in our discussions because bbcp performed poorly for both the large file and lots of small files (LOSF) cases. For example, bbcp was from $3\times$ to $8\times$ slower than other tools in our experiments—or, in the case of Datasets 5 and 6 (see Table II), did not complete at all.

C. Methodology

We evaluate and compare the transfer performance of the different tools by measuring the disk-to-disk transfer times for different datasets (abbreviated as *DS* in the figures). The datasets used in the tests are as shown in Table II. The total size of each dataset is fixed at 10GB, with file sizes varying from 10GB to 100KB. Thus, the presented transfer performance is representative of both large file transfers, as well as the LOSF transfers. We measure the transfer performance consecutively for each tool for a given dataset. Each dataset is transferred ten times to ensure that the measured transfer times are statistically significant and measurements are presented with a confidence interval estimate of 95%.

TABLE II: Transfer Datasets

Dataset#	**1**	**2**	**3**	**4**	**5**	**6**
File Size	10GB	1GB	100MB	10MB	1MB	100KB
#Files	1	10	100	1000	10000	100000

D. Performance Parameters

Different data transfer tools use different performance parameters. Here, we provide a description of the parameters and their values used in our experiments for both large file transfers and LOSF transfers.

1) *GridFTP*: The storage I/O block size was set to 4MB and the TCP buffer size was set to 2MB. Pipelining and Direct I/O were enabled for all transfers. We set concurrency (number of concurrent file transfers) to be 32 and parallelism (number of TCP streams per file) to be 4 for all datasets.
2) *FDT*: For all datasets, the I/O buffer size was set to 32M, the TCP SO_SND_BUFFER size to 2MB and blocking I/O was enabled. All dataset transfers used a total of 32 parallel network streams.
3) *mdtmFTP*: mdtmFTP was set up to use an I/O block size of 4MB on both client/server endpoints and the total number of parallel TCP streams were set to 32. Further, direct I/O was enabled, and the monitoring option was disabled on the server.
4) *XDD*: We configure XDD to use an I/O block size of 4MB, I/O queue depth of 32 and the total parallel network streams to 32, on both the source and the destination endpoints. The direct I/O option was also enabled on both endpoints.

E. Results

We present data transfer performance results for the different tools in Figure 1. In order to present widely varying transfer times observed across different datasets in a readable manner, we present normalized results for each dataset with respect to the lowest time taken by any tool. The lowest time taken is used as a baseline measure for evaluating the performance of other tools. As XDD was designed to move single large files across the WAN, it does not provide directory transfer capabilities. Thus, we only present XDD results for Dataset 1. Further, as noted in Section IV-B, mdtmFTP evaluations are presented for the LAN case only.

1) LAN case: 0.6 ms RTT: Figures 1a shows FDT, GridFTP, mdtmFTP, and XDD results for Dataset 1 (a single 10 GB file) in the 0.6 ms RTT (i.e., LAN) case. XDD has the best performance of the four tools, closely followed by GridFTP and mdtmFTP. All tools are within $1.25\times$ of the baseline.

Figure 1b presents results for Datasets 2 to 6 with GridFTP, FDT, and mdtmFTP. We see that GridFTP and mdtmFTP perform similarly in all cases, with mdtmFTP slightly faster than the other tools. GridFTP transfer performance lies within 10% of mdtmFP. FDT is the slowest in all cases, with a much larger transfer delay for the LOSF case (Dataset 6), where it is about 190% of the baseline (mdtmFTP in this case).

2) WAN case #1: 15 ms RTT: Figure 1c and 1d show results for the 15 ms environment, between the DTNs at UChicago-RCC and PSC-Bridges. For WAN transfer of large files, XDD performance is better than its counterparts, with GridFTP closely following the XDD performance. For Datasets 2 to 5, FDT performs comparably to GridFTP, but its performance deteriorates for the LOSF case.

3) WAN case #2: 30 ms RTT: Figures 1e and 1f show results for the 30 ms environment, between the DTNs at UNL-HCC and PSC-Bridges. For these transfers, we see reduced variations in performance for the large file transfer case among the tools. However, FDT performance for Datasets 2 to 6 decreases with a decrease in file sizes and a corresponding increase in the total number of files transferred between the endpoints.

4) WAN case #3: 60 ms RTT: Figures 1g and 1h show results for the 60 ms environment, between the DTNs at SDSC-Gordon and PSC-Bridges. Results are similar to those seen for the 30 ms environment.

V. Performance Predictability based on Tool Control Parameters

So far we have analyzed various data transfer tools in terms of the maximum performance that they can achieve. In this section, we analyze and compare data transfer tools with respect to the predictability of performance based on tools' control parameters. This is important both for users initiating data transfers and for the administrators of data transfer systems such as data transfer nodes (DTNs), so that they can get an accurate estimate of the transfer time and/or aggregate throughput.

TABLE III: Comparison cases

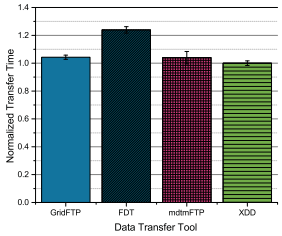
Case#	RTT	Dataset#	Tool
1	0.6ms	All	GridFTP, FDT, mdtmFTP
2	All	1	GridFTP, FDT, XDD
3	All	All	GridFTP, FDT

Based on the data points that we gathered for each data transfer tool, we assess the performance predictability of the tool by running several statistical learning techniques and measuring errors. Regarding the data points, we have some limitations and assumptions. Unfortunately, due to limited configuration parameters of the tools, we used all the data points for training only and measured errors. For fairness, we only compare the tools with the same number of data points and similar control parameters. Accordingly, there are three cases as shown in Table III.

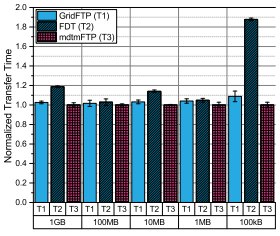
The number of predictors/features varies among data transfer tools since the parameters provided by each tool is different. External loads, e.g. network traffic generated by data transfers other than the one of interest, are assumed to be steady since the performance data are averaged over ten measurements. In general, the training error decreases as the flexibility of a model increases. Therefore, it is obvious that the more predictors are provided, the less training errors will be if all other conditions are same. We conduct the analytical comparison among tools based on the tools' control parameters.

Among many statistical and machine learning techniques, we use two basic techniques namely: linear regression and kNN (k-Nearest Neighbors) regression techniques. We used R [31] for running such algorithms and getting numerical results. We evaluate both the linear regression technique and the polynomial linear regression technique. In case of the polynomial linear regression technique, the degree of the polynomial linear regression is restricted to a minimum of 2 and a maximum of the one less than the number of different points. For example, if a predictor has only 2 distinct values, the degree of the predictor is restricted to 1. In case of kNN, the minimum error when k=2 or k=3 is chosen. We use Mean Absolute Percentage Error (MAPE), which is the average of absolute differences of predicted values and actual values, for error measurement metrics for all techniques. Lower error value indicates better predictability.

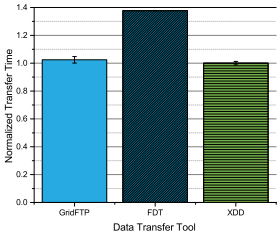
Due to the small number of data points, only the training error results are presented in Table IV. As described in Section IV-B, we could not evaluate all of the tools for each case due to the tools' platform and dependency limitations. Accordingly we evaluated the performance for three cases. Case 1 indicates the test condition with RTT = 0.6ms and all datasets, Case 2 indicates the test condition with all RTTs and Dataset 1, and Case 3 indicates the test condition with all RTTs and all datasets. In Case 1 and 3, linear regression techniques for GridFTP accounts for the best MAPE percent while in Case 2, polynomial linear regression technique works better for other tools than GridFTP. The results also suggest that kNN regression technique in general is a little worse than



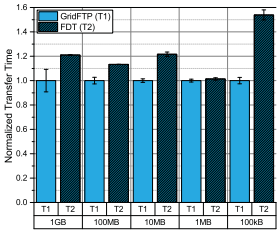
(a) 10GB File transfer performance (DS1), RTT=0.6 ms.



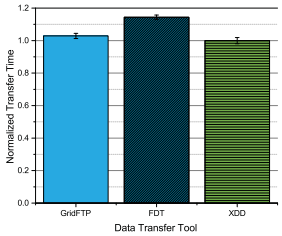
(b) File transfer performance (DS2 to 6), RTT=0.6 ms.



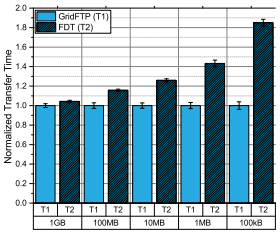
(c) 10GB File transfer performance (DS1) b/w UChicago-RCC and PSC-Bridges, RTT=15 ms.



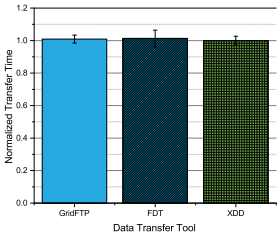
(d) File transfer performance (DS2 to 6) b/w UChicago-RCC and PSC-Bridges, RTT=15 ms.



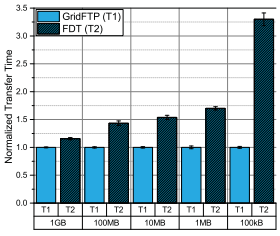
(e) 10GB File transfer performance (DS1) b/w UNL-HCC and PSC-Bridges, RTT=30 ms.



(f) File transfer performance (DS2 to 6) b/w UNL-HCC and PSC-Bridges, RTT=30 ms.



(g) 10GB File transfer performance (DS1) b/w SDSC-Gordon and PSC-Bridges, RTT=60 ms.



(h) File transfer performance (DS2 to 6) b/w SDSC-Gordon and PSC-Bridges, RTT=60 ms.

Fig. 1: Performance Evaluation of the data transfer tools.

TABLE IV: Predictability comparison results (MAPE%) for *linear regression/polynomial linear regression/kNN*.

#	RTT	Dataset#	GridFTP	FDT	mdtmFTP	XDD
1	0.6ms	All	4%/ 0.08%/ 21%	30%/ 27%/ 36%	13%/ 8%/ 16%	NA/ NA/ NA
2	All	1	19%/ 16%/ 142%	60%/ 0%/ 113%	NA/ NA/ NA	72%/ 13%/149%
3	All	All	56%/ 14%/ 100%	128%/ 88%/ 131%	NA/ NA/ NA	NA/ NA/ NA

linear regression technique. But the order of the kNN MAPE of data transfer tools has a similar tendency to the order of the linear regression techniques.

VI. Discussion

Of the data transfer tools compared in this paper, FDT required the least configuration. The Globus Connect Personal packaging of GridFTP [19, 32], although not evaluated here, is similar in this regard. With a Java-based implementation, FDT is highly portable and can be used in a wide range of environments. It is to be noted that all tools compared in this evaluation, with the exception of GridFTP, employed a client-server architecture for data transfers. GridFTP is the only tool in this mix to support third-party transfers, which is important for moving data between different computing center endpoints. Also, we note that XDD does not support recursive/directory file transfers: it can only be used with single-file transfers. XDD is optimized for very large single file transfers and performs well for this use case. In our tests, anonymous authentication was enabled for all GridFTP transfers. We note that the remaining tools offered limited/no support for authentication/confidentiality controls for end-to-end data transfers. The use of authentication with GridFTP will result in additional overheads and it would therefore be interesting to compare the performance of the tools by discounting the security latencies for GridFTP transfers. All GridFTP data transfer results presented here include the authentication costs in the performance evaluations.

To the best of our knowledge, this is a first evaluation of data transfer tools in terms of predictability. Performance predictability can play an important role in autotuning the data transfer tools for maximum performance. A large number of control parameters provided by a data transfer tool can result in additional predictors/features for use with statistical/machine learning techniques. In this paper, we demonstrated the use of techniques such as linear regression, k-nearest neighbors and polynomial regression for performance prediction. GridFTP and mdtmFTP have various parameters to control system and data transfer performance. This results in high predictability when linear regression or kNN are used.

VII. Conclusions and Future Work

We have presented a detailed comparison and evaluation of the capabilities and performance of four data transfer tools: FDT, GridFTP, mdtmFTP, and XDD. Our performance evaluation focuses on different types workloads, including one

that involves many small files–a relatively uncommon, but still important use case for scientific workloads.

Our evaluation suggests that each tool has strengths and weaknesses. FDT and GridFTP win in terms of ease of installation and use. XDD and mdtmFTP performed slightly better than GridFTP in the limited environments and datasets for which we were able to get them to work. GridFTP works in the widest range of environments and delivers performance that is always above 90% (and in many cases above 95%) of that best performance achieved by any tool.

A more comprehensive study would also evaluate data transfer tools along additional dimensions. In particular, reliability and usability are of vital importance for any data transfer tool that aims to be widely adopted in academia and industry. Here, the availability of the Globus transfer service as a highly usable and reliable GridFTP client, with both Web and REST interfaces, is an important differentiator for GridFTP, enabling large-scale deployment [32] and integration into applications [33].

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy under contract number DEAC02-06CH11357 and SDN-SF project, and the National Science Foundation, under grant numbers ACI-1440761, OAC-1541442. This work was completed using the Holland Computing Center of the University of Nebraska, which receives support from the Nebraska Research Initiative.

REFERENCES

[1] S. Jain, A. Kumar, S. Mandal *et al.*, "B4: Experience with a Globally-deployed Software Defined Wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013.

[2] C.-Y. Hong, S. Kandula, R. Mahajan *et al.*, "Achieving High Utilization with Software-driven WAN," in *ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. ACM, 2013, pp. 15–26.

[3] W. Hendrix, I. K. Tetteh, A. Agrawal *et al.*, "Community Dynamics and Analysis of Decadal Trends in Climate Data," in *2011 IEEE 11th Intl. Conf. on Data Mining Workshops*, Dec. 2011, pp. 9–14.

[4] J. Saez-Rodriguez, A. Goldsipe, J. Muhlich *et al.*, "Flexible informatics for linking experimental data to mathematical models via DataRail," *Bioinformatics*, vol. 24, no. 6, pp. 840–847, Mar. 2008.

[5] R. Latham, C. Daley, W.-k. Liao *et al.*, "A case study for scientific I/O: improving the FLASH astrophysics code," *Computational Science & Discovery*, vol. 5, no. 1, p. 015001, 2012.

[6] A. Chervenak, R. Schuler, C. Kesselman *et al.*, "Wide area data replication for scientific collaborations," in *The 6th IEEE/ACM Intl. Workshop on Grid Computing, 2005.*, Nov. 2005.

[7] W. Allcock, J. Bester, J. Bresnahan *et al.*, "GridFTP: Protocol extensions to FTP for the grid," *Global Grid Forum, GFD-RP*, vol. 20, pp. 1–21, 2003.

[8] "Fast Data Transfer," http://monalisa.cern.ch/FDT/.

[9] "BBCP," http://www.slac.stanford.edu/ abh/bbcp/.

[10] L. Zhang, W. Wu, P. DeMar, and E. Pouyoul, "mdtmFTP and its evaluation on ESNET SDN testbed," *Future Generation Computer Systems*, 2017.

[11] "XDD - The eXtreme dd toolset," https://github.com/bws/xdd.

[12] T. Jones, A. Koniges, and R. Yates, "Performance of the IBM general parallel file system," in *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, 2000, pp. 673–681.

[13] Z. Sebepou, K. Magoutis, M. Marazakis, and A. Bilas, "A Comparative Experimental Study of Parallel File Systems for Large-scale Data Processing," in *First USENIX Workshop on Large-Scale Computing*, ser. LASCO'08, Berkeley, CA, USA, 2008, pp. 5:1–5:10.

[14] D. Yun, C. Q. Wu, N. S. V. Rao *et al.*, "Profiling Optimization for Big Data Transfer over Dedicated Channels," in *2016 25th Intl. Conf. on Computer Communication and Networks (ICCCN)*, Aug. 2016, pp. 1–9.

[15] C. Q. Wu, D. Yun, N. Rao, Q. Liu, R. Kettimuthu, and E.-S. Jung, "Data transfer advisor with transport profiling optimization," in *2017 42nd Annual IEEE Conference on Local Computer Networks*, 2017.

[16] Y. Kim, S. Atchley, G. R. Valle, S. Lee, and G. M. Shipman, "Optimizing End-to-End Big Data Transfers over Terabits Network Infrastructure," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 188–201, Jan. 2017.

[17] E.-S. Jung, R. Kettimuthu, and V. Vishwanath, "Cluster-to-cluster data transfer with data compression over wide-area networks," *Journal of Parallel and Distributed Computing*, vol. 79–80, pp. 90–103, May 2015.

[18] B. W. Settlemyer, J. D. Dobson, S. W. Hodson, J. A. Kuehn, S. W. Poole, and T. M. Ruwart, "A Technique for Moving Large Data Sets over High-performance Long Distance Networks," in *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies*, ser. MSST '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–6.

[19] B. Allen, J. Bresnahan, L. Childers *et al.*, "Software as a service for data scientists," *Commun. ACM*, vol. 55, no. 2, pp. 81–88, 2012.

[20] "Physics Experiment Data Export," https://github.com/dmwm/PHEDEX.

[21] J. Lee, D. Gunter, B. Tierney, B. Allcock, J. Bester, J. Bresnahan, and S. Tuecke, "Applied techniques for high bandwidth data transfers across wide area networks," Ernest Orlando Lawrence Berkeley National Laboratory, Tech. Rep. LBNL–47183, 2001.

[22] W. Allcock, J. Bresnahan, K. Kettimuthu, and J. Link, "The globus extensible input/output system (XIO): a protocol independent IO system for the grid," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, Apr. 2005, p. 8 pp.

[23] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, and I. Foster, "GridFTP Pipelining," in *Proc. 2007 TeraGrid Conference*, Teragrid, 2007.

[24] R. Kettimuthu, S. Link, J. Bresnahan, M. Link, and I. Foster, "Globus XIO pipe open driver: enabling GridFTP to leverage standard Unix tools," in *2011 TeraGrid Conference: Extreme Digital Discovery*, ser. TG '11. New York, NY, USA: ACM, 2011, pp. 20:1–20:7.

[25] C. Cirstoiu, R. Voicu, and N. Tapus, "Framework for High-Performance Data Transfers Optimization in Large Distributed Systems," in *2008 International Symposium on Parallel and Distributed Computing*, Jul. 2008, pp. 385–392.

[26] C. A. Mattmann, S. Kelly, D. J. Crichton, J. S. Hughes, S. Hardman, P. Ramirez, and R. Joyner, "A classification and evaluation of data movement technologies for the delivery of highly voluminous scientific data products," 2006.

[27] L. Zhang, W. Wu, P. DeMar, and E. Pouyoul, "mdtmFTP and its evaluation on ESNET SDN testbed," *Future Generation Computer Systems*, Apr. 2017.

[28] "A Multicore-Aware Data Transfer Middleware (MDTM)," https://web.fnal.gov/project/mdtm.

[29] K. B. Wheeler, R. C. Murphy, and D. Thain, "Qthreads: An API for programming with millions of lightweight threads," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, April 2008, pp. 1–8.

[30] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014.

[31] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2018. [Online]. Available: https://www.R-project.org

[32] K. Chard, S. Tuecke, and I. Foster, "Efficient and secure transfer, synchronization, and sharing of big data," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 46–55, 2014.

[33] K. Chard, E. Dart, I. Foster, D. Shifflett, S. Tuecke, and J. Williams, "The Modern Research Data Portal: A design pattern for networked, data-intensive science," *PeerJ Computer Science*, vol. 4, p. e144, 2018.