Explaining Wide Area Data Transfer Performance

Zhengchun Liu
Argonne National Laboratory
Lemont, IL, USA
zhengchun.liu@anl.gov

Prasanna Balaprakash
Argonne National Laboratory
Lemont, IL, USA
pbalapra@anl.gov

Rajkumar Kettimuthu
Argonne National Laboratory
Lemont, IL, USA
kettimut@anl.gov

Ian Foster
Argonne National Lab and University of Chicago
Lemont, IL, USA
foster@anl.gov

ABSTRACT

Disk-to-disk wide-area file transfers involve many subsystems and tunable application parameters that pose significant challenges for bottleneck detection, system optimization, and performance prediction. Performance models can be used to address these challenges but have not proved generally usable because of a need for extensive online experiments to characterize subsystems. We show here how to overcome the need for such experiments by applying machine learning methods to historical data to estimate parameters for predictive models. Starting with log data for millions of Globus transfers involving billions of files and hundreds of petabytes, we engineer features for endpoint CPU load, network interface card load, and transfer characteristics; and we use these features in both linear and nonlinear models of transfer performance, We show that the resulting models have high explanatory power. For a representative set of 30,653 transfers over 30 heavily used source-destination pairs ("edges"), totaling 2,053 TB in 46.6 million files, we obtain median absolute percentage prediction errors (MdAPE) of 7.0% and 4.6% when using distinct linear and nonlinear models per edge, respectively; when using a single nonlinear model for all edges, we obtain an MdAPE of 7.8%. Our work broadens understanding of factors that influence file transfer rate by clarifying relationships between achieved transfer rates, transfer characteristics, and competing load. Our predictions can be used for distributed workflow scheduling and optimization, and our features can also be used for optimization and explanation.

1 INTRODUCTION

Many researchers have studied the performance of network architectures, storage systems, protocols, and tools for high-speed file transfer [15, 20, 25, 33, 36, 38]. Using a mix of experiment, modeling, and simulation, often in highly controlled environments, this work has produced a good understanding of how, in principle, to configure hardware and software systems in order to enable extremely high-speed transfers, which can achieve close to line rates on 10 Gbps and even 100 Gbps networks [11, 23].

Yet despite these results, the actual performance achieved by disk-to-disk transfers in practical settings is usually much lower than line rates. For example, a study of more than 3.9 million Globus transfers [8] involving more than 33 billion files and 223 PB over a seven-year period (2010-2016) shows an average transfer speed of only 11.5 MB/s. (On the other hand, 52% of all bytes moved over that period moved at >100 MB/s and 14% moved at >1 GB/s.)

With effort, we can often explain each low-performing transfer, which may result from (mis)configurations and/or interactions among storage devices, file systems, CPUs, operating systems, network interfaces, intermediate network devices, local and wide area networks, file transfer software, network protocols, and competing activities. But we have lacked an approach that could use easily obtainable information sources to explain and improve the performance of arbitrary transfers in arbitrary environments. We believe that lightweight models are required for this purpose and that the construction of such models will require a combination of data-driven analysis of large collections of historical data, the development and testing of expressive analytical models of various aspects of transfer performance, and new data sources. Here we report on steps toward this goal.

This paper makes four contributions. (1) We show how to use machine learning methods to develop data transfer performance models using only historical data. (2) We engineer features for use in these models, including features that characterize competing load at source and destination endpoints. (3) We identify features that have nonlinear impact on transfer performance, in particular those that capture competing load. (4) We demonstrate that model accuracy can be improved even further by using new data sources to obtain more complete knowledge of competing load.

The rest of the paper is organized as follows. In §2 we provide background on the Globus service that manages the transfers considered here. In §3 we introduce a simple three-feature analytical model that provides just an upper bound on performance, and we use this model to identify factors that impact maximum achievable transfer rate. In §4 we analyze additional factors that affect transfer rate, and we define the features that we use in the data-driven models we introduce in §5, where we describe and evaluate both linear and nonlinear regression models. Starting from a different model for each edge, we incorporate endpoint- and edge-specific features to develop one model for all edges. In §6 we review related

HPDC '17, June 26-30, 2017, Washington, DC, USA

978-1-4503-4699-3/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3078597.3078605

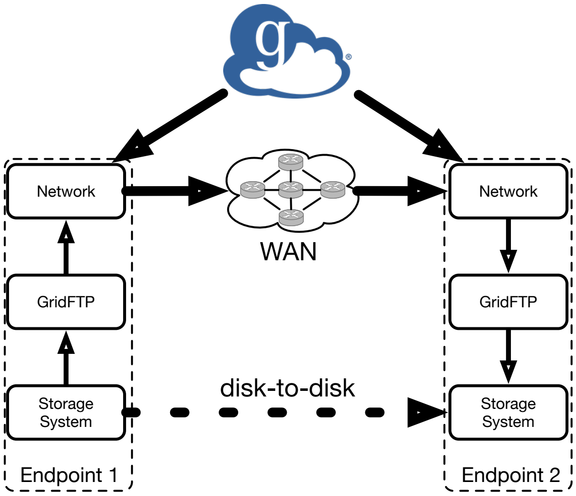


Figure 1: Structure of a Globus end-to-end file transfer from source (left) to destination (right), managed by cloud service.

work, in §7 discuss the broader applicability of our work, and in §8 summarize our conclusions and briefly discuss future work.

2 BACKGROUND ON THE GLOBUS SERVICE

The Globus transfer service is a cloud-hosted software-as-a-service implementation of the logic required to orchestrate file transfers between pairs of storage systems [2] (see Figure 1). A transfer request specifies, among other things, a source and destination; the files and/or directories to be transferred; and (optionally) whether to perform integrity checking (enabled by default) and/or to encrypt the data (disabled by default). Globus can transfer data with either the GridFTP or HTTP protocol; we focus here on GridFTP transfers, since HTTP support has been added only recently. GridFTP extends FTP with features required for speed, reliability, and security.

Globus has been running since late 2010, providing us with a considerable body of transfer performance data. In the work described here, we consider transfers through the end of 2015. These transfers involved ∼26K endpoints, each running Globus Connect software, and 46K unique edges (source–destination endpoint pairs for which at least one transfer has occurred). Figure 2 shows endpoints for which location data are available [8].

These data have limitations: we know relatively little about the endpoints and networks involved in many transfers and little or nothing about competing load. Nevertheless, we can learn some general features about transfer characteristics and performance, as we show in subsequent sections.

3 A SIMPLE ANALYTICAL MODEL

We introduce a simple analytical model for the maximum achievable end-to-end file transfer rate for a given source and destination. We validate this model using both experimental and historical data and draw conclusions about the model's accuracy.

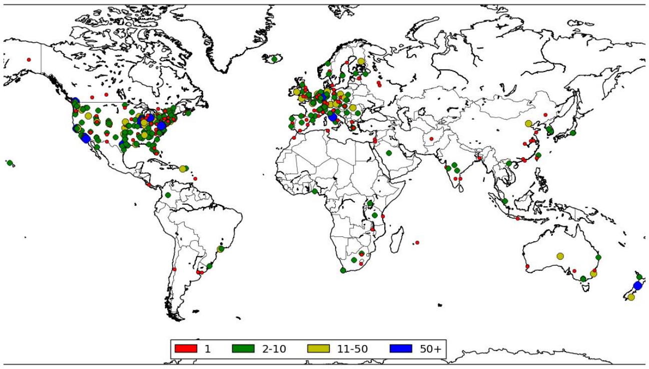


Figure 2: Globus endpoints, grouped by number of deployments in a single location [8]. (Some endpoints geolocate erroneously to the center of countries.)

3.1 Maximum achievable transfer rate

As shown in Figure 1, an end-to-end file transfer engages three subsystems: source endpoint, network, and destination endpoint. The maximum achievable transfer rate, R^{max}, cannot be more than the minimum of the maximum rates achievable by each subsystem:

$$R^{max} \leq min(DR^{max}, MM^{max}, DW^{max}), \tag{1}$$

where DR^{max} is the maximum achievable disk read rate on the source endpoint, MM^{max} is the maximum achievable memory-to-memory transfer rate from source to destination (including the network transfer), and DW^{max} is the maximum achievable disk write rate on the destination endpoint.

To test Equation 1, we conducted data transfer experiments between ESnet testbed nodes to determine R^{max}, DW^{max}, DR^{max}, and MM^{max} separately. The ESnet testbed comprises identical hardware deployed at three DOE labs in the United States (Argonne: ANL; Brookhaven: BNL; and Lawrence Berkeley: LBL) and at CERN in Geneva, Switzerland. Each system features a powerful Linux server configured as a data transfer node (DTN) [11], with an appropriately configured high-speed storage system and 10 Gb/s network link. We use transfers from */dev/zero* to disk and from disk to */dev/null* on each DTN to measure DW and DR separately; from */dev/zero* on source to */dev/null* on destination to measure MM; and from disk on source to disk on destination to measure R. We performed at least five repetitions of each experiment and selected the maximum observed values as R^{max}, DW^{max}, DR^{max}, and MM^{max}.

Table 1 gives our results. We see that all edges are consistent with Equation 1.

3.2 Extending the model to other endpoints

Of the 46K unique edges in the Globus log records studied here, 36,599 had been used for only a single transfer, 16,562 for ≥10 transfers, 2,496 for ≥100 transfers, and 182 for ≥1000 transfers. We focus in this work on the 2,496 edges with ≥100 transfers. For most of these endpoints, we cannot get the access that would be required to measure DR^{max}, DW^{max}, and MM^{max}, information that is also not measured by the GridFTP servers. Instead, we estimate these quantities, as we now describe.

Table 1: Experimentally determined R^{max}, DW^{max} (at destination), DR^{max} (at source), and MM^{max}, in Gb/s, on ESnet testbed, with minimum in each row in bold.

From	To	R^{max}	DW^{max}	DR^{max}	MM^{max}
ANL	BNL	**7.843**	**7.843**	9.302	9.412
	CERN	**6.250**	7.080	9.302	8.989
	LBL	**7.547**	7.767	9.302	9.302
BNL	ANL	**7.407**	7.619	9.302	9.524
	CERN	**6.780**	7.080	9.302	9.091
	LBL	**7.339**	7.767	9.302	9.412
CERN	ANL	**7.080**	7.619	8.696	8.989
	BNL	**7.143**	7.843	8.696	9.091
	LBL	**6.349**	7.767	8.696	8.791
LBL	ANL	**7.407**	7.619	9.302	9.412
	BNL	**7.143**	7.843	9.302	9.412
	CERN	**6.557**	7.080	9.302	8.889

We estimate the first two quantities from the historical data. For each endpoint, we set DR^{max} as the maximum rate observed among all transfers with that endpoint as source and DW^{max} as the maximum rate observed among all transfers with it as destination.

We use perfSONAR [16] to estimate MM^{max} for some edges. This network performance-monitoring infrastructure is deployed at thousands of sites worldwide, many of which are available for open testing of network performance. Many sites that run Globus Connect servers also have perfSONAR hosts with network performance measurement tools connected to the same network as the Globus Connect servers.

We grouped the 2,496 edges with 100 or more transfers by location so that nodes at the same site are treated as equivalent. This grouping resulted in 469 edges with ≥100 transfers. We were able to find perfSONAR hosts at the sites associated with 195 of these edges. Some perfSONAR hosts allow anyone on the research and education network to run third-party Iperf3 [17] tests. Of the 195 edges with perfSONAR hosts at both ends, 81 supported third-party tests. We ran third-party tests for a period of several weeks and collected hundreds of network performance measurements.

Four of the 81 edges on which we performed tests show Globus transfer performance significantly greater than MM^{max} as measured by perfSONAR. In two cases, this is because their perfSONAR and data transfer interfaces are different: the site has a single perfSONAR host with a 10 Gbps network interface card (NIC) but either 4 or 8 DTNs, each with a 10 Gbps NIC.

Of the remaining 77 edges, 38 show Globus transfer rates in the interval $[0.8\ R^{max}, 1.2R^{max}]$ when R^{max} is estimated by Equation 1. After accounting for the known load from other simultaneous Globus transfers (i.e., adding $\max(K^{sout}, K^{din})$: see §4.3), the observed rate for seven more edges also falls in this interval. Thus Equation 1 works reasonably well for a total of 45 edges. Of these, the performance of 11 is limited by disk read, 14 by network, and 20 by disk write.

For the remaining 32 edges, we see significantly lower rates than estimated by Equation 1. We thus examine the log data to see how throughput varies with load from other (competing) Globus transfers. We first calculate the load from competing Globus transfers on a transfer at an endpoint—the *relative endpoint external load*—as follows: We scale the rate of each competing transfer based on the fraction of the time that it overlaps with the transfer with which it competes, sum the scaled rate of all competing transfers (K^{sout} at source and K^{din} at destination), and compute the fraction of competing transfer rate. For example, for a transfer k from endpoint src_k to endpoint dst_k with throughput R_k, we calculate the relative external load of k at src_k and dst_k as $K^{sout}/(R_k + K^{sout})$ and $K^{din}/(R_k + K^{din})$, respectively. We then define the *relative external load* for a transfer as the greater of the relative endpoint external loads for the transfer at the source and destination.

Given this definition, we can then examine how the transfer rate varies with the relative external load. Figure 3 shows one set of results, plotting transfer rate vs. relative external load for each transfer over four edges in the ESnet testbed. As we might expect, the achieved transfer rate declines with the external Globus load, showing that Equation 1 is not sufficient as a model for the end-to-end transfer rate achieved by real transfers. Other features must also be considered.

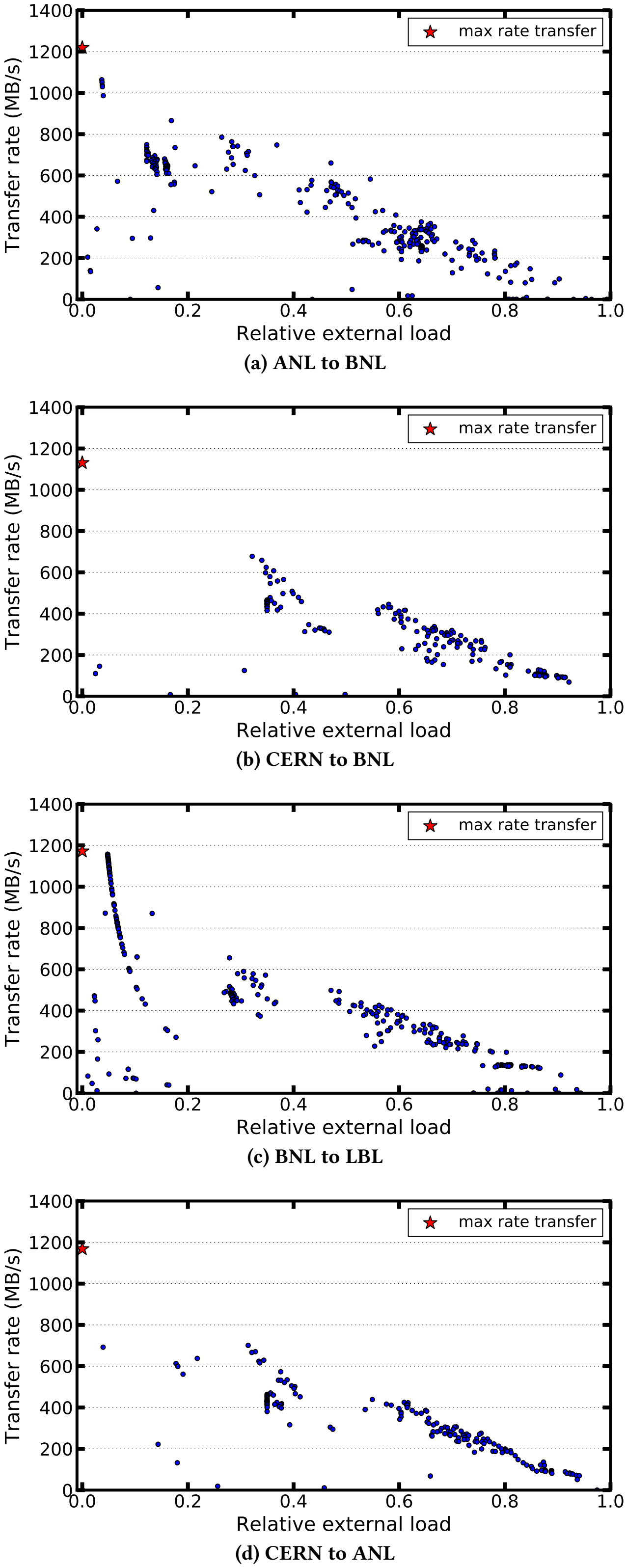
4 TRANSFER FEATURES

Feature engineering is a general term for methods that combine variables to get around the unreasonably large number of variables that are often available in machine learning, while still describing the data with sufficient accuracy. Such methods are often the key to understanding the complex relationships between independent and dependent variables and to developing successful data-driven models such as those constructed by machine learning algorithms. Feature engineering typically uses domain knowledge to create features that make data-driven models work [14].

Starting with measured data, feature engineering seeks to build derived values that are informative and non-redundant. These *features* can facilitate subsequent learning and generalization, and may also enable better human interpretations. We describe here how we generated various features from Globus transfer logs, and we study the utility of these features through extensive experiments. We use these features in §5 to build a data-driven model of achievable Globus transfer performance.

Our starting point for this work is Globus log data, which provide, for each transfer, start time (*Ts*), completion time (*Te*), total bytes transferred, number of files (*Nf*), number of directories (*Nd*), values for Globus tunable parameters, source endpoint, and destination endpoint. (The log also tells us the number of faults associated with a transfer, N_{flt}. Since this number is not known in advance, however, we use it for explanation—see Figures 9 and 12—but not prediction.)

We then construct new features by reproducing resource load conditions on endpoints during each transfer. We join these new features with the log data for training and testing, giving us three groups of features: tunable parameters specified by users, transfer characteristics such as number and size of files, and load measurements that quantify competition for transfer resources. For convenience, we list in Table 2 some notation used in this article. The various terms are introduced in the following.



(a) ANL to BNL

(b) CERN to BNL

(c) BNL to LBL

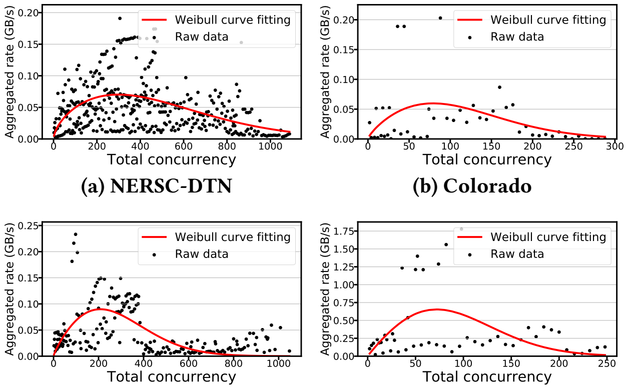
(d) CERN to ANL

Figure 3: Transfer rate vs. relative external load: ESnet.

4.1 Tunable parameters

The Globus GridFTP implementation includes user-configurable features that can be used to optimize transfer performance [1]. Two that are commonly used are concurrency (*C*) and parallelism (*P*). Concurrency involves starting *C* independent GridFTP processes at the source and destination endpoints. Each of the resulting *C* process pairs can then work on the transfer of a separate file, thus providing for concurrency at the file system I/O, CPU core, and network levels. In general, concurrency is good for multi-file transfers, since it can drive more filesystem processes, CPU cores, and even endpoint servers, in addition to opening more TCP streams. Parallelism is a network-level optimization, in which data blocks for a single file between a process pair are distributed over *P* TCP streams. Large files over high-latency links can benefit from higher parallelism, for reasons noted in §6.

While *C* and *P* have significant influence on transfer rate, accurately and efficiently tuning these parameters in a dynamically changing network environment is challenging [4]. Furthermore, the performance achieved by a transfer depends also on the concurrency and parallelism associated with other transfers at the same endpoints. For example, as shown in Figure 4, aggregate transfer throughput first increases but eventually declines as total concurrency across all transfers increases.



(a) NERSC-DTN

(b) Colorado

(c) JLAB

(d) UCAR

Figure 4: Aggregate incoming transfer rate vs. total concurrency (i.e., instantaneous number of GridFTP server instances) at four endpoints, with Weibull curve [37] fitted.

4.2 Transfer characteristics

The total number of bytes in a transfer and the average file size have a significant impact on the transfer rate. Because of startup costs, a transfer with a relatively small total size achieves a lower rate than does a larger transfer. A transfer with many files incurs more coordination overhead, and a dataset with many directories may incur more overhead because of lock contention on parallel filesystems.

To study these effects, we choose one edge with many transfers, namely, JLAB to NERSC. We first group transfers by total size to form 20 groups. Then we determine the average file size for each transfer, and within each group we create two subgroups

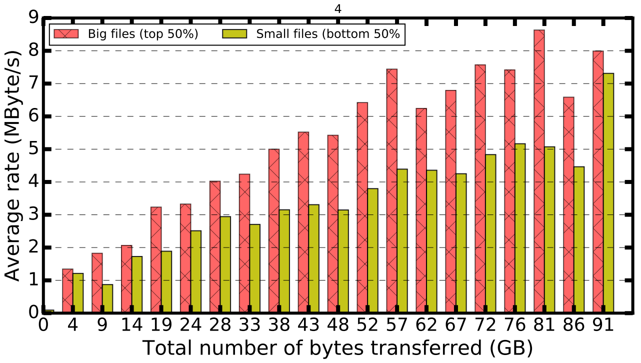


Figure 5: File characteristics versus transfer performance.

comprising transfers with average file size below and above the median in each group, respectively.

Figure 5 shows our results. We observe that transfers with smaller total size achieve a lower rate than do transfers with larger total size. Within each total size bucket, transfers with higher average file size achieve a higher rate than do those with lower average file size. Note that the average rates for "small files" and "big files" transfers are not always directly comparable across different total size buckets, because a larger total size does not necessarily mean a larger average file size. For example, the average file size of "big files" transfers in the "86 GB total bytes" bucket is less than for the "big files" transfers in the "72 GB total bytes" bucket. Similarly, the reason for the small difference between the average rates for "big files" and "small files" in the "91 GB total bytes" bucket is that the average file sizes in those two groups are similar.

Figure 6 presents a view of overall transfer characteristics across all edges. Each transfer is plotted according to its transfer size and estimated transfer distance (great circle distance between source and destination, a lower bound), with color denoting the transfer rate. We see again evidence of tremendous variety in transfer characteristics, with transfer sizes ranging from 1 byte to close to a petabyte and transfer rates from 0.1 bytes/second to a gigabyte/second. Transfer rate clearly correlates somewhat with transfer size and distance, as we would expect. Note the clear distinction between intracontinental and intercontinental transfers.

4.3 Load measurements

We saw in Figure 3 how transfer rate varies with what we defined in §3.2 as *relative external load.* This dependence reflects the reality that Globus data transfers occur in a shared resource environment. Each transfer may contend with both other Globus transfers and other non-Globus tasks that engage the same source and/or destination endpoint. We have information about the competing Globus transfers from Globus logs; here we integrate domain knowledge of the GridFTP protocol and implementation with Globus log data to define features that we expect to influence transfer rate.

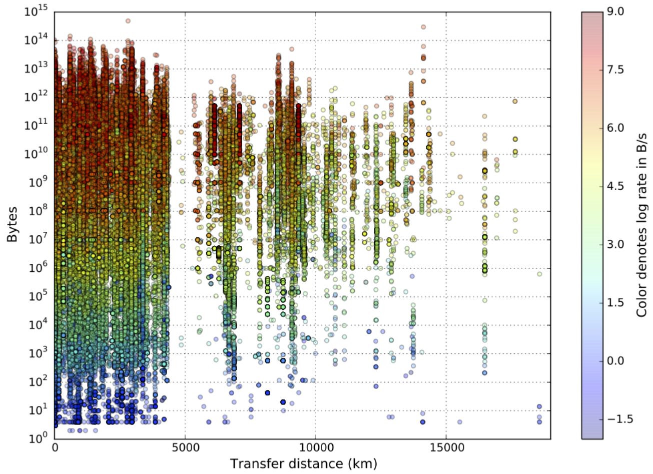


Figure 6: Transfer size vs. estimated transfer distance; color encodes transfer rate.

Table 2: Notation used in this article. We use the lower 15 terms as features in our models.

src_k	Source endpoint of transfer k.
dst_k	Destination endpoint of transfer k.
Ts_k	Start time of transfer k.
Te_k	End time of transfer k.
R_k	Average transfer rate of transfer k.
N_{flt}	Number of faults a transfer experienced.
K^{sin}	Contending incoming transfer rate on src_k.
K^{sout}	Contending outgoing transfer rate on src_k.
K^{din}	Contending incoming transfer rate on dst_k.
K^{dout}	Contending outgoing transfer rate on dst_k.
C	Concurrency: Number of GridFTP processes.
P	Parallelism: Number of TCP channels per process.
S^{sin}	Number of incoming TCP streams on src_k.
S^{sout}	Number of outgoing TCP streams on src_k.
S^{din}	Number of incoming TCP streams on dst_k.
S^{dout}	Number of outgoing TCP streams on dst_k.
G^{src}	GridFTP instance count on src_k.
G^{dst}	GridFTP instance count on dst_k.
Nf	Number of files transferred.
Nd	Number of directories transferred.
Nb	Total number of bytes transferred.

4.3.1 Accounting for competing Globus transfers. The performance of a Globus transfer may be degraded by competing load from other simultaneous Globus transfers that engage the same source and/or destination endpoint. We know a lot about these transfers from Globus logs; the question is how we should translate this information into a small set of features. One obvious feature is the aggregate data transfer rate of the competing transfers. A second feature, given that network performance is often sensitive to interactions among concurrent TCP connections, is the number of TCP connections for the competing transfers. As mentioned

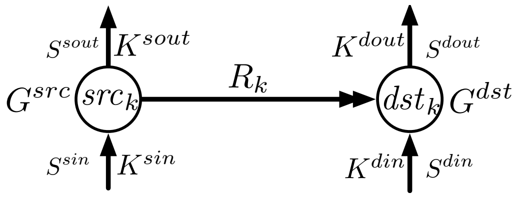


Figure 7: Load experienced by a Globus transfer k from endpoint src_k to endpoint dst_k with rate R_k, from other simultaneous Globus transfers: (a) GridFTP instances at source and destination (G^{src}, G^{dst}); (b) outgoing and incoming TCP streams at source and destination (S^{sout}, S^{sin}, S^{dout}, S^{din}), and (c) contending outgoing and incoming traffic rates at source and destination (K^{sout}, K^{sin}, K^{dout}, K^{din}).

in §4.1, the total number of TCP connections for a transfer is the product of its concurrency (C) and parallelism (P). For example, a GridFTP transfer with C=4, P=4 and a transfer with C=16, P=1 both involve 16 TCP connections. However, these two transfers involve 4 and 16 GridFTP server processes, respectively, and the latter is likely to result in more CPU load than does the former. Thus, we define as a third feature the number of GridFTP server processes associated with competing transfers.

Based on these considerations, we classify the load from such competing Globus transfers in terms of their equivalent contending transfer rate, GridFTP instance count, and parallel TCP streams. Each quantity is an *aggregate*: in each case we sum over all competing transfers. We refer to *equivalent* loads in each case because, as we will see, we scale the load due to a competing transfer by the fraction of the time that it overlaps with the transfer with which it competes. Figure 7 illustrates these different contending features for a transfer k from source src_k to destination dst_k endpoints. As described earlier, we perform a time series analysis to identify the competing Globus transfers.

The Globus **contending transfer rate** for a transfer k at its source (src_k) and destination (dst_k) endpoints (see Figure 7) is

$$K^{x\in\{sout,sin,dout,din\}}(k) = \sum_{i\in A_x} \frac{O(i,k)}{Te_k - Ts_k} R_i, \tag{2}$$

where A_x is the set of transfers (excluding k) with src_k as source when x=*sout*; src_k as destination when x=*sin*; dst_k as source when x=*dout*; and dst_k as destination when x=*din*. $O(i,k)$ is the overlap time for the two transfers:

$$O(i,k) = \max\left(0, \quad \min(Te_i, Te_k) - \max(Ts_i, Ts_k)\right).$$

The **GridFTP instance count** on transfer k's source and destination endpoints (G^{src} and G^{dst}, respectively) due to competing transfers is represented as follows:

$$G^{x\in\{src,dst\}}(k) = \sum_{i\in A_x} \frac{O(i,k)}{Te_k - Ts_k} \min(C_i, F_i),$$

where C_i is the user-specified concurrency and F_i is the number of files transferred in the ith competing transfer, both from the Globus log. The set A_x contains all transfers except k that have src_k as their source or destination. The $\min(C_i, F_i)$ is because a transfer with $F_i < C_i$ can use only F_i GridFTP instances.

The number of simultaneous **parallel TCP streams**, $S(k)$, of the competing transfers in each data flow direction is

$$S^{x\in\{sout,sin,dout,din\}}(k) = \sum_{i\in A_x} \frac{O(i,k)}{Te_k - Ts_k} \min(C_i, F_i) P_i,$$

where P_i is the user-specified parallelism of transfer i. The sets A_x are as in Equation 2.

4.3.2 Accounting for other competing load. Figure 3 illustrates a situation in which transfer rate varies fairly cleanly with external load. We see that the highest transfer rate is always achieved when relative external load (K) is zero, as we expect.

In other settings, things are more complicated. For example, Figure 8 plots transfer rate versus relative external load for each transfer between four edges involving endpoints with high-speed networks and storage systems at the Texas Advanced Computing Center (TACC), Argonne Leadership Computing Facility (ALCF), National Energy Research Scientific Computing Center (NERSC: two different endpoints), San Diego Supercomputer Center (SDSC), and Jefferson Laboratory (JLAB). Here, the relationship between known external load and achieved transfer rate is less clear. In fact, with the exception of the NERSC-DTN to the JLAB edge, the maximum observed transfer rate (marked by a red star) is at a point other than when the load from other Globus transfers is the lowest.

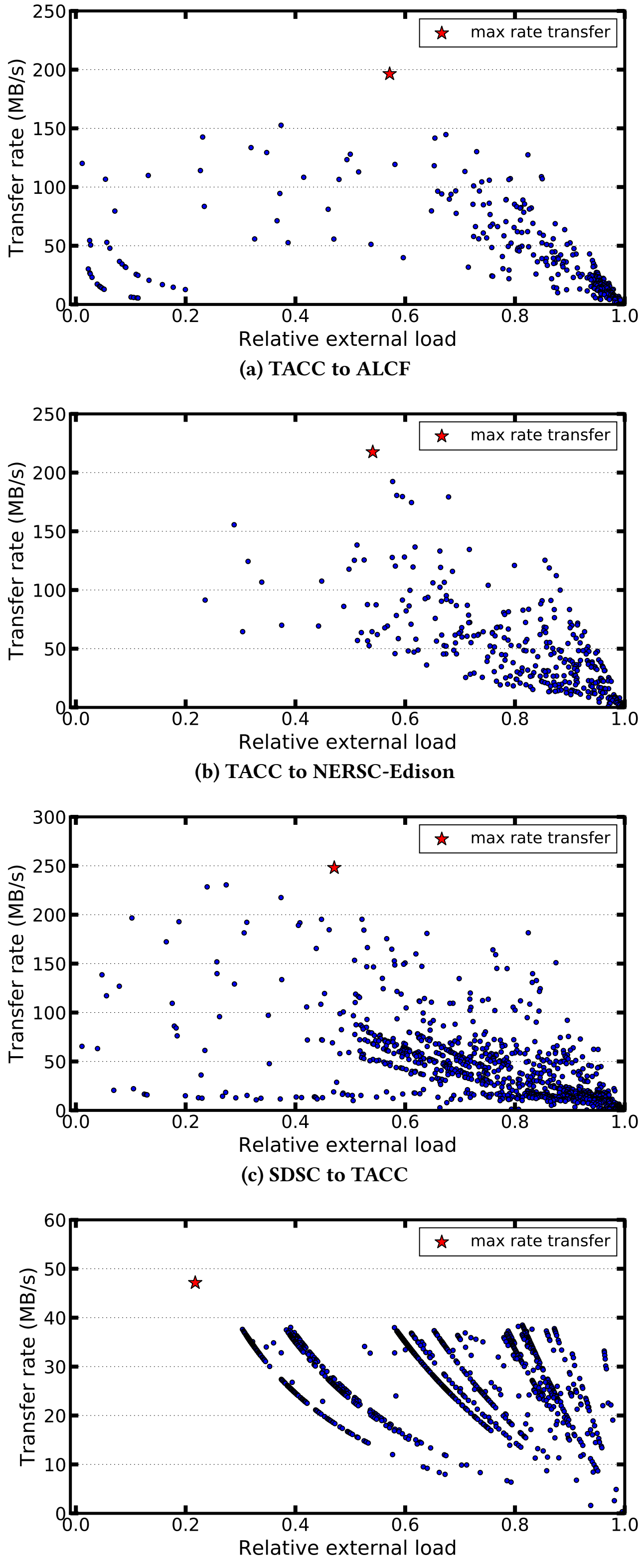
One likely reason for this discrepancy is competition from non-Globus activities, such as file transfers performed with other tools, storage activities performed by other tasks, and other traffic on network link(s) between source and destination. We explore such effects in §5.5.2, but in general we have no information that we can use to quantify this *other competing load.* Thus, we address the limitation of missing information on non-Globus load by considering in our analyses only transfers that achieve a high fraction of peak, under the hypothesis that these transfers are unlikely to have suffered from much other competing load. Specifically, for each edge, E, we first determine the highest transfer rate achieved between the two endpoints, $R^{max}(E)$, and then remove from our dataset transfers that have a rate less than $T.R^{max}(E)$, where T is a load threshold, set to 0.5 except where otherwise specified.

This approach is not ideal. It may also remove transfers that perform badly because of, for example, transfer characteristics (e.g., small files). However, we show in §5.5 that the accuracy of our models improves with load threshold. wide area network conditions.

5 REGRESSION ANALYSIS

We use regression analysis to explain the relationship between the transfer rate and the 15 independent variables in Table 2. In particular, we investigate whether the transfer rate can be modeled as a linear and nonlinear combination of independent variables. To test linear dependence, we use linear regression to fit the data. For the nonlinear testing, there exists a wide range of supervised-machine-learning algorithms of varying complexity. We use gradient boosting [9], a state-of-the-art supervised-machine-learning algorithm that has proven effective on many predictive modeling tasks.

Both methods benefit from preprocessing since the scale of the independent variables is quite different. Therefore, we normalize each input x_i to have zero mean and unit variance, setting $x' =$



(a) TACC to ALCF

(b) TACC to NERSC-Edison

(c) SDSC to TACC

(d) NERSC-DTN to JLAB

Figure 8: Transfer rate vs. relative external load for four edges, each involving heavily used endpoints.

$(x_i - \overline{x_i})/\sigma_i$, where $\overline{x_i}$ and σ_i are the mean and standard deviation of x_i, respectively.

5.1 Linear regression

Linear regression (LR) assumes that the relationship between the rate R_i for each transfer i and the independent variables is linear:

$$R_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im}, \qquad (3)$$

where the x_{ij} are the m features for each of the n transfers i and $\beta_0, \cdots, \beta_m$ are coefficients that are estimated by minimizing the residual sum of squares between the observed transfer rates (R_i) and those predicted ($\hat{R_i}$) by the linear approximation. Mathematically, we solve a problem of the form

$$\min_{\beta_0, \cdots, \beta_m} \sum_{i=1}^{n} \left(R_i - \hat{R_i}\right)^2. \qquad (4)$$

We fit this linear regression model separately on each edge. We use edges that have at least 300 transfers with rate greater than 0.5 R^{max}. For each edge, we use these transfers to train and test the model, since these transfers are less likely to have unknown (non-Globus) competing load (detailed in §4.3.2) and thus are more likely to explain the importance of each feature to transfer performance. These transfers account for 46.5% of the raw data over these 30 heavily used edges. For each edge, we randomly select 70% of the log data to train the model and the other 30% to test the model. Both the training and test data, which include the derived and original transfer features, are available online [27]. (The data have been anonymized to protect the privacy of endpoints and users.)

These 30 edges are representative of all edges in the log in important ways. To demonstrate this, we consider three edge characteristics: *edge length*, which we approximate by determining the great circle distance between source and destination and which serves as a proxy for the round-trip time; maximum observed aggregate transfer *rate*; and *edge type*, which is in turn determined by its source and destination endpoint types. Table 3 compares the distribution of great circle length for all edges versus for the 30 edges. For the maximum observed aggregate rate, the 30 edges range from 6.4 MB/s (5th percentile) to 1.2 GB/s (95th percentile), while all edges range from 2.1 MB/s (5th percentile) to 1.2 GB/s (95th percentile). Table 4 compares the distribution of edge types. [There are two endpoint types—server (Globus Connect Server, or GCS) and personal computer (GCP)—and thus four edge types, of which three are represented in the log. (Globus did not support GCP to GCP transfers before 2016.)]

Table 3: Edge length statistics (km) for three percentiles.

Dataset	25th	50th	90th
All edges	235	1,976	3,062
30 edges	247	1,436	3,947

An advantage of the fitted linear regression model is that it reveals the relationship between a single input variable and the transfer rate when all other input variables in the model are kept constant. In particular, the interpretation of β_i is the expected change in R for a one-unit change in x_i when no other input changes at the

Table 4: Edge type statistics (%).

Dataset	*GCS* ⇒ *GCS*	*GCS* ⇒ *GCP*	*GCS* ⇒ *GCP*
All edges	45	34	20
30 edges	51	30	19

same time. This is called the *unique effect* of x_i on *R*. Figure 9 shows the relative values of the coefficients. (We scaled the coefficients by dividing each coefficient into the maximum value of its edge so that all maximums have the same size). C and P are eliminated for all edges because they do not vary greatly in the log data. Since load on *sout* and *din* represent direct contention, we are not surprised to see that they have considerable influence on transfer rate. Although $S^{\{sin,\ sout,\ din,\ dout\}}$ and $K^{\{sin,\ sout,\ din,\ dout\}}$ all presumably reflect network load, they get different weights in the model. This result tells us that no strong correlation exists between them, which further argues that more TCP streams do not always contribute to higher aggregate transfer rate. G^{src} and G^{dst} are also significant for most edges: more concurrent GridFTP processes mean more contention on CPU, memory, and storage resources.

5.2 Nonlinear model

We conducted an exploratory analysis to check for nonlinear relationships between rate and the independent variables. We computed the Pearson linear correlation and nonlinear maximal information coefficients, as shown in Table 5. Several inputs have a higher nonlinear maximal information coefficient than the Pearson correlation coefficient, indicating nonlinear dependencies between features and rate that cannot be captured by a linear model.

For the nonlinear model, we use a gradient boosting approach, an iterative approach in which at each iteration a new decision tree is added to correct errors made by previous trees. A gradient descent algorithm is used to minimize the error when adding each new tree. Sequentially built trees are combined to produce the final model. An advantage of gradient boosting is that after the trees are constructed, computing the importance scores for each independent variable is straightforward. Intuitively, the more an independent variable is used to make the main splits within the tree, the higher its relative importance. The importance for each independent variable is then averaged across all the decision trees. Note that unlike in the linear model, the importance score does not correspond to the unit increase or decrease in rate.

We use eXtreme Gradient Boosting (XGB) [9], a high-performing gradient boosting implementation that is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges such as Kaggle and KDDCup. The effectiveness of XGB stems from several careful optimizations, both algorithmic (a novel approximate tree learning algorithm and an efficient procedure to handle training point weights in approximate tree learning) and system level (out-of-core computation to process data that is too large to fit in main memory, and cache-aware learning to improve speed and scalability) [9].

For each given edge, we use 70% of the data to train the XGB model and the remaining 30% for testing.

5.3 Prediction results

Now, we compare the LR and XGB prediction errors. We find that nonlinear regression improves over linear regression and that the relationship between input variables and transfer rate is nonlinear. Figures 10 and 11 show the prediction errors for each edge. We see that XGB has lower errors than LR has for most edges, presumably because it captures more information (nonlinear dependencies) about the relationship between features and transfer rate.

Figure 12 shows the importance of features over each edge. Comparing with Figure 9, we see that most features have similar importance across the linear and nonlinear models. Some features (e.g., K^{sout}, S^{sout}, N_b) are important in both. However, the number of faults, N_{flt}, is a far less important feature in the nonlinear case. We know that faults have a significant negative impact on performance, so why are they not important in the nonlinear case? One possible reason is that faults occur when load is high, leading to a correlation between faults and a nonlinear function of load. Thus, the nonlinear model can account for the impact of faults by selecting an appropriate function of load.

5.4 A single model for all edges

The success of our edge-specific regression analyses encourages us to examine the feasibility of capturing endpoint differences in additional features, in order to create a single general model for all edges. Since we lack information about endpoint properties, such as NIC capacity, CPU speed, core count, memory capacity, and storage bandwidth, we use data from Globus logs to construct two new features for each endpoint. Specifically, we define for each endpoint *E* its maximum outgoing rate, $ROmax_E$, as follows:

(1) Let src_E be all transfers with *E* as their source.
(2) For each transfer *x* in src_E, estimate its Globus contending outgoing transfer rate (i.e., from its source endpoint) as $K^{sout}(x)$ from Equation 2.
(3) Determine the maximum outgoing rate for endpoint *E*:

$$ROmax_E = \max_{x \in Src_E} \left(R_x + K^{sout}(x)\right)$$

Similarly, we determine the endpoint's maximum incoming rate, $RImax_E$, also from Equation 2.

$$RImax_E = \max_{x \in Dst_E} \left(R_x + K^{din}(x)\right)$$

We can then extend Equation 3 to obtain the general model.

$$\begin{aligned} R_i = {} & \beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} + \beta_{m+1} ROmax_{s_i} + \\ & \beta_{m+2} RImax_{d_i} \end{aligned} \tag{5}$$

Here x_{ij} are as in Equation 3; s_i and d_i are the source and destination endpoints for the transfer *i*; $RImax_{s_i}$ and $ROmax_{d_i}$ are our two new features; and $\beta_0, \cdots, \beta_{m+2}$ are the coefficients. (Intuitively, $\beta_0, \cdots, \beta_m$ capture the behavior of the Globus service and β_{m+1} and β_{m+2} the capabilities of the source and destination endpoints, respectively.) We then estimate $\beta_0, \cdots, \beta_{m+2}$, as we did $\beta_0, \cdots, \beta_m$ in §5.1 except that we perform the minimization over *N* transfers associated with *all* of our 30 selected edges.

$$\min_{\beta_0, \cdots, \beta_{m+2}} \sum_{i=1}^{N} \left(R_i - \hat{R}_i\right)^2. \tag{6}$$

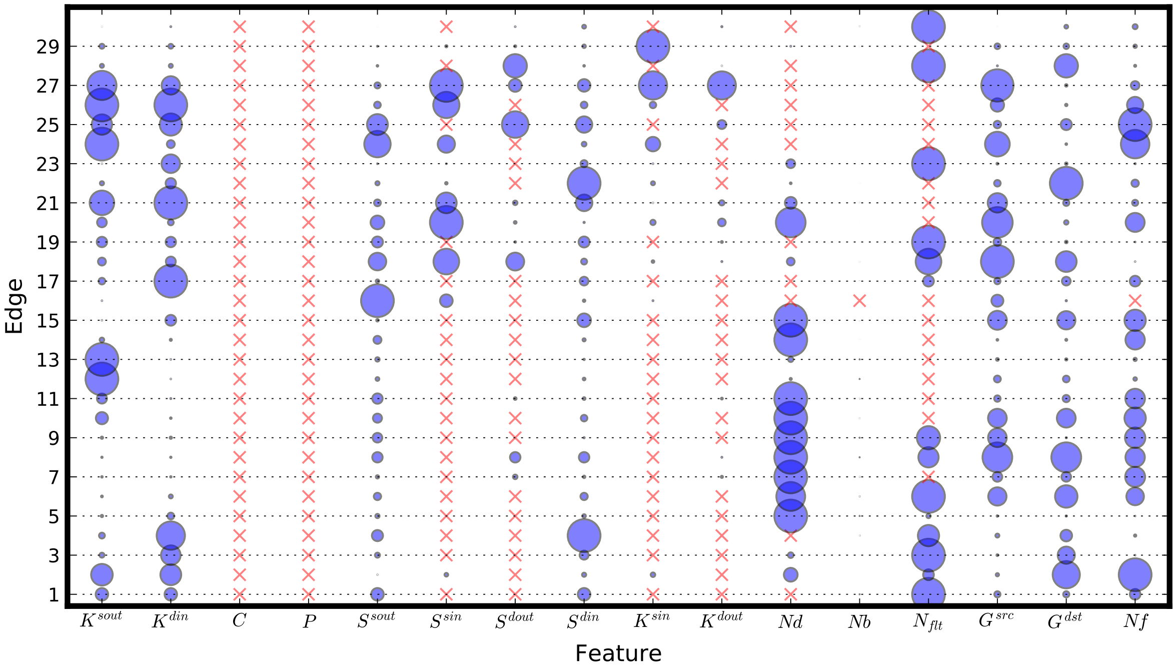


Figure 9: Circle size indicates the relative significance of features in the linear model, for each of 30 edges. A red cross indicates that the corresponding feature is eliminated because of low variance. Features from different samples are not comparable.

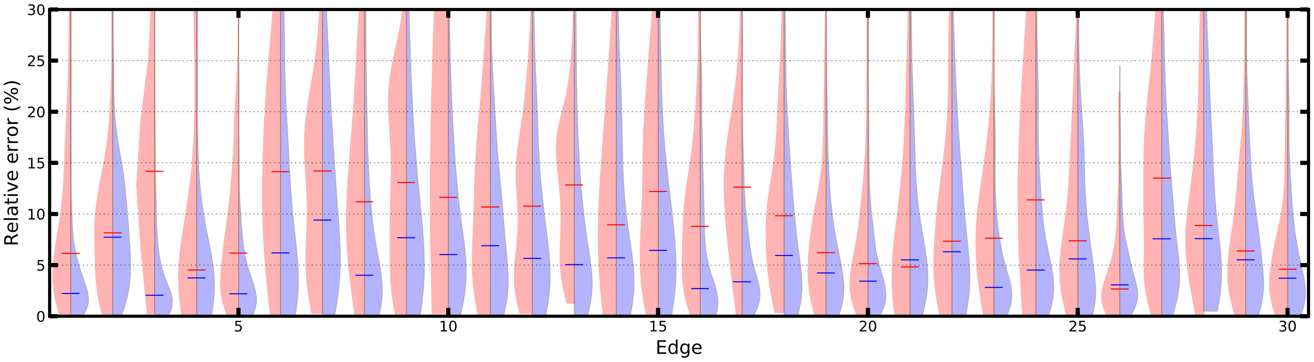


Figure 10: Comparison of linear regression and eXtreme Gradient Boosting models. For each edge, the left Violin plot gives the prediction error of the linear regression model and the right the prediction error of eXtreme Gradient Boosting model.

Specifically, we work with the transfers from the 30 edges with rate greater than 0.5 R^{max}. We use 70% of these 30,653 transfers to train the linear model and the other 30% for testing. We obtain an MdAPE of 19%: higher than when we train individual models for each edge, but still useful for many purposes. For example, this model can be used to predict transfer rates for an edge that has few or no transfers, if that edge's source and destination endpoints have been involved in transfers to other endpoints. We also train the nonlinear model XGB and obtain an MdAPE of 4.9%. In future work, we will incorporate round-trip times for each edge, which we expect to reduce errors further.

5.5 Reducing or eliminating the unknowns

The unknown aspects that impact the transfer rate contribute to the inaccuracies in the models developed so far. Here we try to reduce or eliminate the unknowns and see how doing so can help improve the accuracy of the models.

5.5.1 Reducing the unknowns. As discussed in §4.3.2, the results reported here are for transfers with rate ≥ $0.5R^{max}$, under the hypothesis that such transfers are less likely to suffer from competing load. To explore whether transfers with higher rates are more likely to have less unknown load, we also applied the eXtreme Gradient Boosting method to datasets obtained by setting the threshold as $0.6R^{max}$, $0.7R^{max}$, and $0.8R^{max}$. Figure 13 shows the prediction errors for all four models for the eight edges that have more than

Table 5: Correlation study between the features of Table 2 and transfer rate. CC is the Pearson correlation coefficient and MIC the maximal information coefficient. Missing data in CC rows (–) mean that the corresponding features have uniform value.

ID	K^{sout}	K^{din}	C	P	S^{sout}	S^{sin}	S^{dout}	S^{din}	K^{sin}	K^{dout}	Nd	Nb	G^{src}	G^{dst}	Nf
CC	0.23	0.41	–	–	0.20	0.16	0.51	0.46	0.16	0.40	0.12	0.12	0.11	0.56	0.13
MIC	0.25	0.66	0.00	0.00	0.30	0.06	0.52	0.63	0.06	0.66	0.17	0.39	0.28	0.66	0.19
CC	0.17	0.10	–	–	0.06	0.11	0.21	0.16	0.09	0.21	0.20	0.32	0.19	0.01	0.20
MIC	0.47	0.33	0.00	0.00	0.48	0.23	0.18	0.45	0.23	0.18	0.13	0.49	0.46	0.49	0.13
CC	0.02	0.03	/	/	0.01	/	0.18	0.03	/	0.11	0.11	0.22	0.11	0.06	0.11
MIC	0.16	0.24	0.00	0.00	0.19	0.00	0.18	0.26	0.00	0.18	0.20	0.41	0.20	0.28	0.20
CC	0.03	0.24	/	/	0.01	0.12	/	0.02	0.14	/	0.03	0.43	0.09	0.02	0.04
MIC	0.26	0.17	0.00	0.00	0.24	0.26	0.00	0.29	0.29	0.00	0.06	0.53	0.26	0.18	0.40

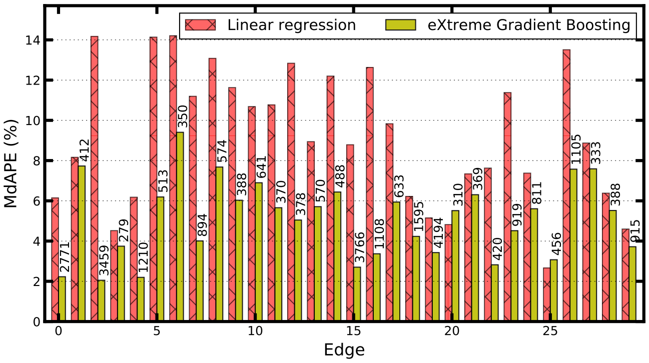


Figure 11: MdAPEs for linear and and eXtreme Gradient Boosting models, and the actual number of samples used.

300 transfers that satisfy the $0.8R^{max}$ threshold. Prediction errors generally decline as the threshold increases, as we expect.

5.5.2 Eliminating the unknowns. An alternative way to reduce the impact of unknown load on model results is to collect more information about the endpoint and possibly the network. To explore the utility of this approach, we performed test transfers over endpoints for which we could monitor all load (including that external to Globus) on storage. We added this new storage load information to the feature set, used the new feature set to train a data-driven model, and evaluated the prediction accuracy of this new model.

We performed these experiments in an environment comprising two Lustre file systems at NERSC: one shared with the Edison supercomputer and one with a DTN. We used Globus to perform a series of test transfers from one Lustre object storage target (OST) to another, keeping 10 additional simultaneous Globus load transfers running at all times in order to mimic a production environment. Throughout the experiments, we used the Lustre Monitoring Tool (LMT) to collect, every five seconds, both disk I/O load for each Lustre OST and CPU load for each Lustre object storage server (OSS). We performed 666 test transfers in total, of which we randomly picked 70% for training and the rest for testing.

To provide a baseline for evaluation of the utility of the LMT-measured data, we first trained the model described in §5.2 with the same features used in earlier sections, namely, the lower 15 terms in Table 2. The 95th percentile error is 9.29%. This error is lower than that seen in §5.1 and §5.2, which we attribute to the fact that the source and destination are at the same site.

We then introduced four new features to represent storage load: CPU load on source OSS, CPU load on destination OSS, disk read on source OST, and disk write on destination OST. With these new features, a nonlinear model of the type described in §5.2 achieved a 95th percentile error of just 1.26%.

These results suggest that if we can characterize all currently unknown loads, we can build an accurate model for transfer rate. We note, however, that the environment in which we performed this study differs from production environments in important respects. In particular, our transfer characteristics were uniform for all transfers (i.e., N_b, N_f, and N_{dir} are the same across all transfers), and since we transferred data only internally at NERSC, we did not have to deal with the challenging issue of network contention.

6 RELATED WORK

Models have been developed for individual components in the end-to-end file transfer, including TCP based on first principles [13, 31, 34], and storage systems [7, 26, 39, 40]. In other work [19], we used models for the individual system components involved in an end-to-end data transfer and optimized the data transfer using the models. But such modeling is challenging because it requires a lot of information for each individual endpoint.

Parallel TCP streams are extensively used in wide-area data transfers to increase the aggregate TCP window size and provide increased resilience to packet losses [15, 29]. Several researchers have modeled the behavior of parallel TCP streams [3, 10, 15, 24, 29], and some studies [18, 30, 41] have focused on such streams specifically in the context of GridFTP. In our work here, we model the end-to-end performance characteristics of file transfers, where the parallel TCP stream is one of many aspects that impact the performance. In addition to parallel streams, we take into account several other features that impact the transfer rate, including dataset characteristics and load on the transfer hosts, storage, and network.

Other prior work has sought to develop end-to-end file transfer models [4, 21, 22, 28, 35]. Vazhkudai and Schopf [35] and Lu et al. [28] propose models that rely on performance data on individual components such as network, disk, and application. Kim et al. [22] and Arslan et al. [4] rely on real-time probing. Although Kettimuthu et al. [21] consider external load in their model, neither

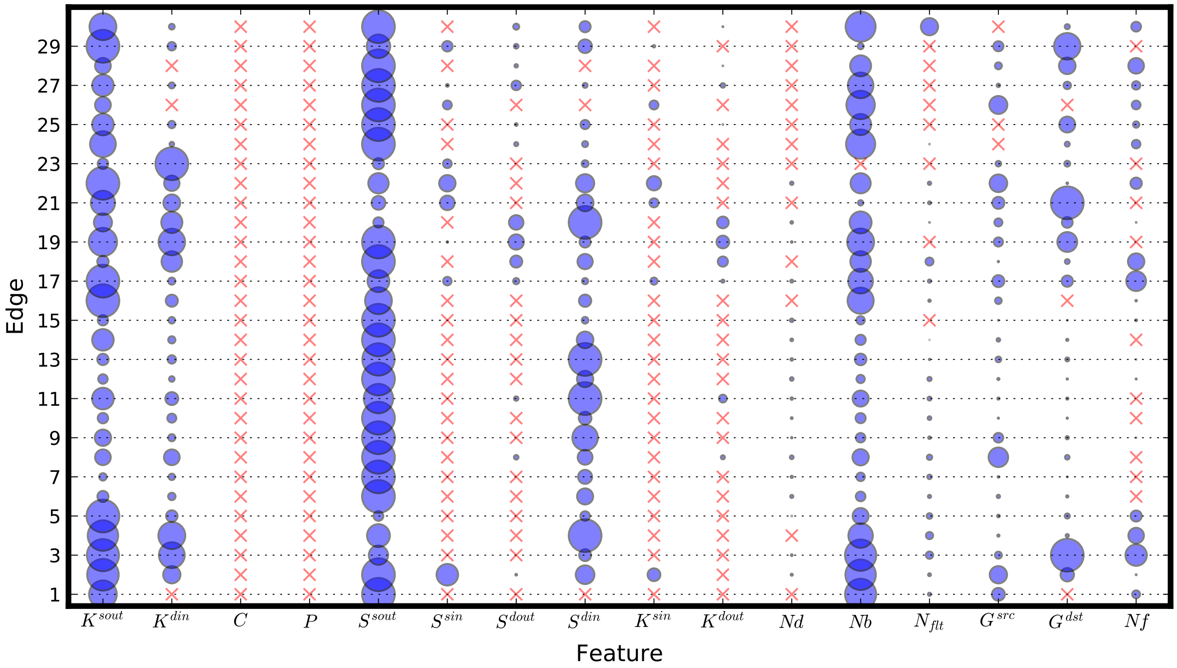


Figure 12: Circle size indicates the relative importance of features in the eXtreme Gradient Boosting model, for each of 30 edges. A red cross indicates a feature that is eliminated due to low variance. Features from different samples are not comparable.

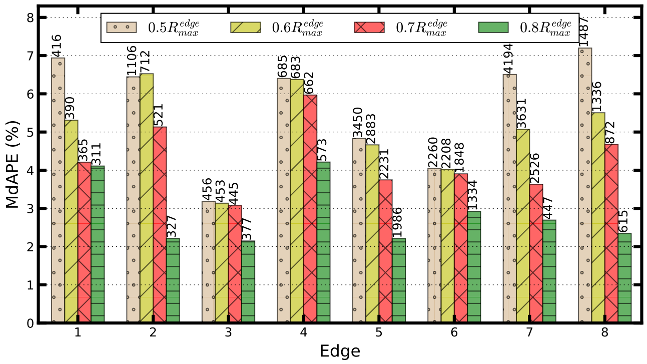


Figure 13: MdAPEs for the linear model on eight edges, when training on sets of transfers with different R^{max} thresholds. We show the number of data points in each case.

they nor other studies examined the impact of the external load as extensively as has been done in this paper.

7 RELEVANCE TO OTHER TOOLS

While we performed this work using Globus data, we believe that our methods and conclusions are applicable to all wide area data transfers. The features that we used (number of TCP connections, number of network and disk I/O threads / processes, size of the data transfer, number of files, competing load) are generic features that impact the performance of any wide area data transfer, irrespective of the tool employed. Features such as number of TCP connections, number of network and disk I/O threads / processes, size of the data transfer, and number of files can be obtained in a straightforward fashion for other data transfer tools such as FTP, rsync, scp, bbcp [6], FDT [12], and XDD [32]. Given these features, our method can be used to compute competing load for an environment in which one of those other tools dominates. In fact, if the logs for all data transfer tools are available, our method can be used to compute the competing load from all tools, which we can expect to uncover more unknowns and thus improve model accuracy.

8 CONCLUSIONS

We have used a large collection of Globus transfer records to gain insight into the behavior of large science data transfers and the factors that affect their behavior. We generate various features from Globus logs and study the importance of these features in models. For 30,653 transfers over 30 heavily used source-destination pairs ("edges"), totaling 2,053 TB in 46.6 million files, we obtained median absolute percentage prediction errors (MdAPE) of 7.0% and 4.6% when using distinct linear and nonlinear models per edge, respectively. When using a single nonlinear model for all 30 edges, we obtain an MdAPE of 7.8%. We are currently applying these models to other Globus transfers.

Although we have focused on Globus transfers, we expect our approach and proposed features to have broad applicability for wide area file transfers that involve parallel TCP streams. In particular, our feature engineering work provides useful hints and insights for data science practitioners in wide area data transfer. We demonstrate, for example, the importance of creating measures of endpoint load to capture the impact of contention for computer, network interface, and storage system resources. One implication is that contention at endpoints can significantly reduce aggregate performance of even overprovisioned networks. This result suggests that aggregate performance can be improved by scheduling transfers and/or reducing concurrency and parallelism.

We have identified several directions for improved transfer service monitoring that we hope can improve our models by improving knowledge of other loads. Globus currently records information only about its transfers: it collects no information about non-Globus load on endpoints or about network load. A new version with the ability to monitor overall endpoint status is under development. Further research is needed to study the influence of network load. To this end, we plan to incorporate SNMP data from routers to characterize network conditions. Another direction for future work is to see whether more advanced machine learning methods, for example multiobjective modeling with machine learning (AutoMOMML) [5], can yield better models.

ACKNOWLEDGMENTS

This material was supported in part by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. We thank Nagi Rao for useful discussions, Brigitte Raumann for help with Globus log analysis, Glenn Lockwood for help with experiments at NERSC described in §5.5, and the Globus team for much good work and advice.

REFERENCES

[1] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster. The Globus striped GridFTP framework and server. In *SC'05*, pages 54–61, 2005.

[2] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, and S. Tuecke. Software as a service for data scientists. *Commun. ACM*, 55(2):81–88, Feb. 2012.

[3] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic. Parallel TCP sockets: Simple model, throughput and validation. In *25th IEEE Intl Conf. on Computer Communications*, pages 1–12, April 2006.

[4] E. Arslan, K. Guner, and T. Kosar. HARP: predictive transfer optimization based on historical analysis and real-time probing. In *SC'16*, pages 25:1–25:12, 2016.

[5] P. Balaprakash, A. Tiwari, S. M. Wild, and P. D. Hovland. AutoMOMML: Automatic Multi-objective Modeling with Machine Learning. In *ISC*, pages 219–239, 2016.

[6] BBCP. http://www.slac.stanford.edu/~abh/bbcp/.

[7] P. H. Carns, B. W. Settlemyer, and W. B. Ligon III. Using server-to-server communication in parallel file systems to simplify consistency and improve performance. In *SC'08*, page 6, 2008.

[8] K. Chard, S. Tuecke, and I. Foster. Globus: Recent enhancements and future plans. In *XSEDE'16*, page 27. ACM, 2016.

[9] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. *arXiv preprint arXiv:1603.02754*, 2016.

[10] J. Crowcroft and P. Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing TCP. *SIGCOMM Comput. Commun. Rev.*, 28(3):53–69, July 1998.

[11] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski. The Science DMZ: A network design pattern for data-intensive science. *Scientific Programming*, 22(2):173–185, 2014.

[12] FDT. *FDT - Fast Data Transfer.* http://monalisa.cern.ch/FDT/.

[13] J. Gao and N. S. V. Rao. TCP AIMD dynamics over Internet connections. *IEEE Communications Letters*, 9:4–6, 2005.

[14] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003.

[15] 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 *16th Intl Parallel and Distributed Processing Symp.*, page 314, 2002.

[16] A. Hanemann, J. W. Boote, E. L. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, D. M. Swany, S. Trocha, and J. Zurawski. PerfSONAR: A service oriented architecture for multi-domain network monitoring. In *3rd Intl Conf. on Service-Oriented Computing*, pages 241–254, Berlin, Heidelberg, 2005. Springer-Verlag.

[17] iperf3. http://software.es.net/iperf/.

[18] T. Ito, H. Ohsaki, and M. Imase. GridFTP-APT: Automatic parallelism tuning mechanism for data transfer protocol GridFTP. In *6th IEEE Intl Symp. on Cluster Computing and the Grid*, pages 454–461, 2006.

[19] E.-S. Jung, R. Kettimuthu, and V. Vishwanath. Toward optimizing disk-to-disk transfer on 100G networks. In *7th IEEE Intl Conf. on Advanced Networks and Telecommunications Systems*, 2013.

[20] T. Kelly. Scalable TCP: Improving performance in highspeed wide area networks. *ACM SIGCOMM Computer Communication Review*, 33(2):83–91, 2003.

[21] R. Kettimuthu, G. Vardoyan, G. Agrawal, and P. Sadayappan. Modeling and optimizing large-scale wide-area data transfers. *14th IEEE/ACM Intl Symp. on Cluster, Cloud and Grid Computing*, 0:196–205, 2014.

[22] J. Kim, E. Yildirim, and T. Kosar. A highly-accurate and low-overhead prediction model for transfer throughput optimization. *Cluster Computing*, 18(1):41–59, 2015.

[23] E. Kissel, M. Swany, B. Tierney, and E. Pouyoul. Efficient wide area data transfer protocols for 100 Gbps networks and beyond. In *3rd Intl Workshop on Network-Aware Data Management*, page 3. ACM, 2013.

[24] G. Kola and M. K. Vernon. Target bandwidth sharing using endhost measures. *Perform. Eval.*, 64(9-12):948–964, Oct. 2007.

[25] T. Kosar, G. Kola, and M. Livny. Data pipelines: Enabling large scale multi-protocol data transfers. In *2nd Workshop on Middleware for Grid Computing*, pages 63–68, 2004.

[26] N. Liu, C. Carothers, J. Cope, P. Carns, R. Ross, A. Crume, and C. Maltzahn. Modeling a leadership-scale storage system. In *Parallel Processing and Applied Mathematics*, pages 10–19. 2012.

[27] Z. Liu, P. Balaprakash, R. Kettimuthu, and I. Foster. Explaining wide area data transfer performance. http://hdl.handle.net/11466/globus_A4N55BB, 2017.

[28] D. Lu, Y. Qiao, P. Dinda, and F. Bustamante. Characterizing and predicting TCP throughput on the wide area network. In *25th IEEE Intl Conf. on Distributed Computing Systems*, pages 414–424, June 2005.

[29] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante. Modeling and taming parallel TCP on the wide area network. In *19th IEEE Intl Parallel and Distributed Processing Symp.*, page 68b, 2005.

[30] H. Ohsaki and M. Imase. On modeling GridFTP using fluid-flow approximation for high speed Grid networking. In *Symp. on Applications and the Internet–Workshops*, pages 638–, 2004.

[31] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Trans. Networking*, 8(2):133–145, 2000.

[32] 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 *27th Symp. on Mass Storage Systems and Technologies*, pages 1–6, May 2011.

[33] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, and D. Gunter. The NetLogger methodology for high performance distributed systems performance analysis. In *7th Intl Symp. on High Performance Distributed Computing*, pages 260–267, 1998.

[34] G. Vardoyan, N. S. V. Rao, and D. Towsley. Models of TCP in high-BDP environments and their experimental validation. In *24th Intl Conf. on Network Protocols*, pages 1–10, 2016.

[35] S. Vazhkudai and J. Schopf. Using regression techniques to predict large data transfers. *Int. J. High Perf. Comp. Appl.*, 2003.

[36] D. X. Wei, C. Jin, S. H. Low, and S. Hegde. FAST TCP: Motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Networking*, 14(6):1246–1259, 2006.

[37] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, pages 293–297, 1951.

[38] R. Wolski. Forecasting network performance to support dynamic scheduling using the Network Weather Service. In *6th IEEE Symp. on High Performance Distributed Computing*, 1997.

[39] J. M. Wozniak, S. W. Son, and R. Ross. Distributed object storage rebuild analysis via simulation with GOBS. In *Intl Conf. on Dependable Systems and Networks Workshops*, pages 23–28, 2010.

[40] Q. M. Wu, K. Xie, M. F. Zhu, L. M. Xiao, and L. Ruan. DMFSsim: A distributed metadata file system simulator. *Applied Mechanics and Materials*, 241:1556–1561, 2013.

[41] E. Yildirim, D. Yin, and T. Kosar. Prediction of optimal parallelism level in wide area data transfers. *IEEE Trans. Parallel Distrib. Syst.*, 22(12):2033–2045, Dec. 2011.