Performance Prediction of Big Data Transfer Through Experimental Analysis and Machine Learning

Daqing Yun*, Wuji Liu†, Chase Q. Wu†, Nageswara S.V. Rao‡, and Rajkumar Kettimuthu§
*Harrisburg University †New Jersey Institute of Technology ‡Oak Ridge National Lab §Argonne National Lab
Email: dyun@harrisburgu.edu, {wl87, chase.wu}@njit.edu, raons@ornl.gov, kettimut@anl.gov

***Abstract*—Big data transfer in next-generation scientific applications is now commonly carried out over connections with guaranteed bandwidth provisioned in High-performance Networks (HPNs) through advance bandwidth reservation. To use HPN resources efficiently, provisioning agents need to carefully schedule data transfer requests and allocate appropriate bandwidths. Such reserved bandwidths, if not fully utilized by the requesting user, could be simply wasted or cause extra overhead and complexity in management due to exclusive access. This calls for the capability of performance prediction to reserve bandwidth resources that match actual needs. Towards this goal, we employ machine learning algorithms to predict big data transfer performance based on extensive performance measurements, which are collected over a span of several years from a large number of data transfer tests using different protocols and toolkits between various end sites on several real-life physical or emulated HPN testbeds. We first identify a comprehensive list of attributes involved in a typical big data transfer process, including end host system configurations, network connection properties, and control parameters of data transfer methods. We then conduct an in-depth exploratory analysis of their impacts on application-level throughput, which provides insights into big data transfer performance and motivates the use of machine learning. We also investigate the applicability of machine learning algorithms and derive their general performance bounds for performance prediction of big data transfer in HPNs. Experimental results show that, with appropriate data preprocessing, the proposed machine learning-based approach achieves 95% or higher prediction accuracy in up to 90% of the cases with very noisy real-life performance measurements.**

***Index Terms*—Performance prediction, experimental analysis, machine learning, big data transfer.**

I. Introduction

Dedicated connections provisioned in High-performance Networks (HPNs) such as ESnet [1] are increasingly used for big data transfer in various domains ranging from extreme-scale scientific research to industrial big data analytics. Provisioning agents in HPNs typically ask end users to request bandwidths as needed and then establish corresponding end-to-end paths with reserved bandwidth.

Predicting end-to-end data transfer performance (mainly throughput) is critical to the utilization of bandwidth resources in HPNs. A dedicated connection, once allocated and granted, is used exclusively by the requesting user during the approved time window. Due to the exclusive access, any granted bandwidth, if not fully utilized, may simply be wasted. Therefore, accurate performance prediction is not only useful for end users to understand and optimize big data transfer performance, e.g., determining what transfer method to use and what control parameter values to set [5], but also important for HPN management to wisely schedule user data transfer requests, e.g., rejecting requests with "over-claimed" bandwidth demands or granting bandwidths in the amount that can be actually utilized.

ISBN 978-3-903176-28-7

However, predicting end-to-end big data transfer performance in HPNs is challenging. Although the exclusive use of dedicated connections minimizes the impact of complex dynamics caused by cross traffic on performance prediction, many other factors involved in a typical big data transfer process may still affect the end-to-end performance to a great extent, including i) end host system configurations, ii) network connection properties, and iii) data transfer methods and their corresponding control parameters. In general, it is very difficult to apply an analytical approach to big data transfer performance prediction in HPNs, due to i) the lack of accurate throughput performance models for high-performance data transfer protocols such as UDT [8], ii) the complex composition of end-to-end dedicated connections, iii) the complexities and varieties of end host configurations; and iv) the time-varying concurrent workloads in end host systems. Consequently, HPN technologies and services have not been fully utilized for big data transfer regardless of the continuous bandwidth upgrades in backbones.

In this paper, we employ a machine learning-based approach to predict big data transfer performance in HPNs based on extensive performance measurements, which are collected and accumulated over a span of several years from a large number of big data transfer tests using different protocols and toolkits between various end sites on several real-life physical or emulated HPN testbeds. Based on such performance measurements, we first identify a comprehensive list of attributes (parameters) involved in a typical big data transfer process, including end host system configurations, network connection properties, and control parameters of data transfer methods. We then conduct an in-depth qualitative exploratory analysis of their impacts on end-to-end performance as observed by end users at the application level. We also investigate the applicability of machine learning in performance prediction in HPNs and conduct experiments for performance evaluation.

We summarize our contributions in this work as follows.

- **Exploratory Analysis**. We conduct an in-depth analysis of a comprehensive list of transport-related attributes to qualitatively explain their impacts on end-to-end big data

transfer performance in HPNs. Such analysis motivates the use of machine learning and further provides insights into feature selection in later learning-based performance prediction.

- **Performance Prediction**. We show that the performance predictor built on the "critical" features selected based on the exploratory analysis is statistically meaningful by deriving a general performance bound incorporating several domain-specific conditions of HPNs, and then apply Support Vector Regression (SVR) to demonstrate the effectiveness of big data transfer performance prediction using machine learning methods.

The rest of the paper is organized as follows. We conduct a brief survey of related work in Sec. II. Sec. III describes the problem of performance prediction of big data transfer in HPNs. We introduce the performance measurement dataset used in this work in Sec. IV. An exploratory analysis of big data transfer performance is conducted in Sec. V. Theoretical performance bounds are derived and practical prediction results using machine learning are presented in Sec. VI. We conclude our work and sketch a research plan in Sec. VII.

II. Related Work

The importance of provisioning dedicated connections to support big data movement over long distances has been well recognized in both science and network communities. We conduct a brief survey of existing work in related areas.

A. Transport Profiling and Optimization

Many profiling-oriented toolkits have been used to conduct data transfer tests to understand and optimize network performance. For example, iperf2 is a handy tool available in most Linux distributions to run TCP tests to estimate available bandwidth. ESnet iperf3 [2], which is a rewrite of iperf2 from scratch, provides a rich set of functions and options for tuning TCP, UDP, and SCTP. Similar to iperf2/3, TPG [4] is a toolkit for conducting UDT [8] data transfer tests. In addition to the common tunable parameters in iperf2/3, TPG enables tuning of UDP socket options for UDT and other UDT-specific parameters, and also supports profiling tests based on multiple physical NIC-to-NIC connections.

Some of these toolkits have been integrated into transport profiling optimization as functional units to carry out data transfer tests guided by established optimization strategies such as stochastic approximation [5]. Performance measurements are collected by measuring throughput in response to various attributes including end host system configurations, network connection properties, and control parameter values, and can be utilized to train performance models for prediction.

B. Throughput Performance Modeling and Prediction

There are numerous efforts devoted to understanding the behaviors of transport protocols and modeling their throughput performance. For example, Padhye *et al*. in [10] develop a throughput model of TCP bulk data transfer on the Internet corresponding to loss rate and connection delay. TCP throughput in response to delay and number of streams is presented in a more recent study [15]. Gu *et al*. present in [20] a throughput model of another representative protocol, UDT [8], and use experimental results over 1 Gbps connections to show that UDT recovers much faster than TCP from a loss event and achieves stable asymptotic performance across different RTTs. Another experimental study [16], however, shows that UDT has certain instability and variety over 10 Gbps connections.

These modeling and experimental studies mainly focus on analytically understanding the behaviors of data transfer protocols in the Internet environments in response to the factors that represent time-varying conditions (e.g., available bandwidth, etc.) without considering the stability of dedicated connections and a comprehensive list of other attributes (e.g., control parameters) that also have non-negligible impacts. Therefore, they have limited capability and accuracy in predicting the performance of big data transfer in HPNs.

Machine learning has great potential for performance prediction of big data transfer in HPNs. For example, Mirza *et al.* in [13] use a support vector regression method to predict TCP file transfer performance in publicly shared Internet environments, using path properties and file transfer size as the features. Liu *et al.* in [21] use regression analysis to explain the observed performance patterns based on the log files of GridFTP-powered disk-to-disk wide-area file transfers and empirically build a wide-area file transfer performance predictor in [22]. Our work differs from the aforementioned efforts in that we identify a comprehensive list of attributes involved in a typical big data transfer process over dedicated connections, including not only the properties and conditions of network connections, but also the configurations of end host systems as well as various control parameters of data transfer methods and their underlying transport protocols. We conduct an exploratory analysis of their impacts on big data transfer performance in HPNs, and then use machine learning algorithms for performance prediction.

III. Problem Statement

End-to-end big data transfer in HPNs is a complex process that involves both network and end-host components, any of which could become the bottleneck and hence limit the performance. Among these components, some can be accessed and controlled by the applications, including packet size, block size, buffer size, and number of parallel streams; while the others are mainly determined by the hardware and system configurations as well as network infrastructures, including CPU frequency, memory size and bandwidth, bus speed, disk I/O speed, path MTU size, RTT, link bandwidth, and loss rate.

Although network connections in HPNs are reserved in advance (typically with high bandwidth) and some of their properties such as RTT and connection loss rate are relatively stable, end-to-end throughput performance is still largely affected and limited by many other issues such as a mismatch of end host capabilities and configurations (e.g., a faster sender host to a slower receiver host), the use of an unsuitable data transfer protocol [5], incorrect settings of CPU affinity [14] and IRQ balance/conflict [12], and suboptimal control parameter values [5]. Due to the quantity and complexity of these factors, it is dauntingly difficult to develop an analytical approach to directly predict transport performance, especially in

the presence of unpredictable factors such as system dynamics and competing workloads on end hosts.

Informally, we attempt to answer the following question. Given two *end hosts* (a sender and a receiver) and a *network connection* between them with *guaranteed bandwidth*, using a specific *data transfer tool* and its underlying *transport protocol* together with their corresponding *control parameter values*, what *end-to-end throughput performance* could be achieved?

The throughput performance y of a data transfer could be expressed as a function of multiple variables including: i) end host system configurations $\mathcal{H}$, ii) network connection properties $\mathcal{P}$, and iii) control parameters $\mathcal{C}$ of the data transfer method and its underlying protocol, which collectively form a feature vector $\mathbf{x} = [\mathcal{H}, \mathcal{P}, \mathcal{C}]$, i.e., $y = y(\mathbf{x})$. The analytical form of function y is essentially unknown (or too complicated to be accurately modeled). Let y_i be the throughput performance in response to $\mathbf{x}_i$ as observed by end users at the application level. We have $y_i = y(\mathbf{x}_i) + \xi_i$, where y_i is the "noise-corrupted" throughput observation, and ξ_i is an i.i.d. random variable that represents the noise including the dynamics in end host systems and the randomness in performance measurements. The expected throughput $\overline{y}_i$ of a big data transfer during time interval $[0, \Delta T]$ is given by $\overline{y}_i = \frac{\int_0^{\Delta T} y(\mathbf{x}_i, t)\ dt}{\Delta T}$, where $y(\mathbf{x}_i, t)$ is the throughput at time point t in response to feature $\mathbf{x}_i$. In practice, we obtain $\overline{y}_i$ by approximating it with its corresponding *observed* goodput $\tilde{y}_i = \frac{F_i(\mathbf{x}_i)}{\Delta T}$, where $F_i(\mathbf{x}_i)$ is the size of transferred user payload and ΔT is the time taken to complete the transfer. We take $\tilde{y}_i$ as the ground truth (i.e., let $y_i = \tilde{y}_i$) and measure y_i in unit of Mbps, unless specifically indicated otherwise.

The variables that may affect end-to-end throughput performance of big data transfer, i.e., $\mathbf{x}$, have a large range of possible values in a multidimensional space, and the performance measurement dataset $\mathcal{T}$ can be viewed as samples in this space. These samples are used to generalize to the whole feature set and performance space for prediction purposes, and we explore machine learning to predict big data transfer performance based on historically collected performance measurements. More specifically, given a training dataset $\mathcal{T}$ of n performance measurements of data transfer tests, i.e., $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ $(i = 1, 2, \ldots, n)$ is an instance of the feature vector that determines the corresponding performance y_i. We propose to employ machine learning methods to estimate y_i based on $\mathcal{T}$ such that the predicted (estimated) value $\hat{y}_i$ is close enough to the true value y_i for all training samples $(\mathbf{x}_i, y_i)$, $i = 1, 2, \ldots, n$ and can be applied to arbitrary future cases with high accuracy.

IV. Throughput Performance Measurements

We briefly describe the acquisition of our performance dataset in this section.

A. Testbeds

Our performance measurements are accumulated and archived in the past several years based on numerous data transfer tests conducted on several HPN testbeds managed by different institutions, as briefly introduced below.

1) Local Testbed at University of Memphis

This testbed, denoted as "UM-Local", is established by back-to-back connecting two workstations (dual-core, 2.9 GB RAM) via a 10 GigE NIC. Both ends (um.dragon and um.rabbit) are installed with FC 17 Linux OS of 3.9.10 kernel.

2) Local Testbed at New Jersey Institute of Technology

This testbed, denoted as "NJIT-Local", is established between two back-to-back connected high-end servers (njit.tiger and njit.rabbit), each with 12 cores, 16 GB RAM, CentOS 3.10 kernel, and a 10 GigE NIC.

3) Physical Testbed between Argonne National Laboratory and University of Chicago

This testbed, denoted as "ANL-UC", consists of several physical network connections of 2 ms, 100 ms, and 380 ms RTTs established between three end hosts at Argonne National Laboratory (ANL), i.e., jlse (64 cores, 64 GB RAM, CentOS 3.10 kernel, and 10 GigE NIC), tubes (16 cores, 48 GB RAM, CentOS 2.6 kernel, and bound 4x10 GigE NIC), and tubes2 (64 cores, 128 GB RAM, CentOS 3.10 kernel, and bound 2x40 GigE NIC), and two identical virtual end hosts at University of Chicago (UC) (g1903 and g1904 with domain name midway) that are dynamically allocated (64 cores, 32 GB RAM, CentOS 2.6 kernel, and 40 GigE NIC). The delay of the 2 ms connection is mainly due to physical distance, while the 100 ms and 380 ms connections are engineered using a layer-2 circuit within ESnet [1] that starts from ANL, extends to the west coast of the US, and then loops back to UC. Note that on this testbed, if the data transfer test is with a single stream, only one of the bound NICs is utilized (thus, the bandwidth is 10 Gbps).

4) Local Testbed at University of Chicago

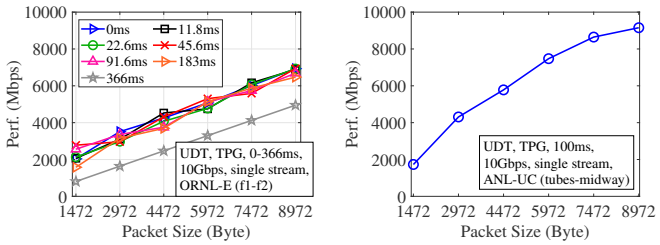
This testbed, denoted as "UC-Local", is a 40 Gbps local connection established between those two dynamically allocated virtual end hosts (g1903 and g1904) located at UC.

5) Emulated Testbed at Oak Ridge National Laboratory

This emulated testbed at Oak Ridge National Laboratory (ORNL), denoted as "ORNL-E", consists of various connections with different emulated RTTs between several pairs of high-end workstations, each pair with identical hardware configurations and 10 GigE NICs. Two 48-core workstations (bohr04 and bohr05, or b4 and b5, with Linux 3.10 kernel) are back-to-back connected with 10 GigE NICs directly connected to two IXIA emulator ports. Another two 32-core Linux workstations (feynman1 and feynman2, or f1 and f2, with Linux 2.6 kernel and CentOS 6.8 release) are connected via a back-to-back fiber connection with two SONET OC192 ANUE emulator ports, where E300 switches are used to convert between 10 GigE LAN-PHY and WAN-PHY frames that are inter-operable with OC192 frames. The peak capacity of this OC192 connection is 9.6 Gbps, less than 10 Gbps of the 10 GigE connection. We use these emulators to collect performance measurements for network connections with various RTTs of $\{0, 11.8, 22.6, 45.6, 91.6, 183, 366\}$ ms. The RTTs in the mid range represent US cross-country connections and higher RTTs represent transcontinental connections.

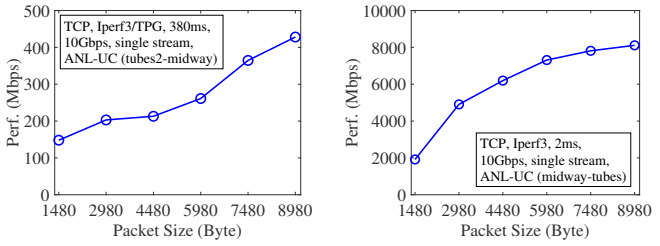
TABLE I
LIST OF ATTRIBUTES IN THE DATA TRANSFER PERFORMANCE DATASET.

Categories	Attributes	Remarks
Identifier	record ID	integer, unique
Testbed	testbed name	string, nominal
End host	CPU frequency	double, hertz
	number of processors	integer
	number of cores per processor	integer
	memory size	integer, MB
	kernel buffer size	integer, byte
Connection	bandwidth	double, Gbps
	RTT	double, ms
	loss rate	double, emulated
Toolkits & Protocols	data transfer protocol	string, nominal
	data transfer toolkit	string, nominal
Control parameters	frame size	integer, byte
	packet size	integer, byte
	payload size	integer, byte
	block size	integer, byte
	TCP send buffer size	double, MB
	TCP recv buffer size	double, MB
	UDP send buffer size	double, MB
	UDP recv buffer size	double, MB
	UDT send buffer size	double, MB
	UDT recv buffer size	double, MB
	number of parallel streams	integer
	data size	double, MB
	time duration	integer, seconds
Performance	throughput / goodput	double, Mbps



(a) ORNL-E, 0–366ms, 10 Gbps (b) ANL-UC, 100ms, 10 Gbps

Fig. 1. **UDT** performance vs. **packet size**. Performance results are collected from different testbeds with single-stream data transfer tests.



(a) ANL-UC, 380ms, 10 Gbps (b) ANL-UC, 2ms, 10 Gbps

Fig. 2. **TCP** performance vs. **packet size**. Performance results are collected from different testbeds with single-stream data transfer tests.

B. Data Transfer Protocols and Toolkits

The data transfer tests are performed using two main data transfer protocols, TCP and UDT. TCP is the *de facto* standard protocol on the Internet and has a number of variants (among which, we mainly use Cubic [9] and STCP [11] in this work); and UDT [8] is a UDP-based data transfer protocol designated for Long Fat Networks (LFNs) and is widely adopted in HPN community. Part of the tests are conducted specifically for data collection purpose and powered by profiling-oriented toolkits including iperf2, ESnet iperf3 [2], and TPG [4]; the rest are mainly the byproducts of the experimental studies for transport profiling optimization (FastProf [5]) and data transfer advising (ProbData [5]). All of these tests enable the tuning of various control parameters of TCP and UDT.

C. Data Acquisition and Introduction

We use the toolkits in Sec. IV-B to run big data transfer tests over dedicated connections on various HPN testbeds. The tests typically take time on the order of minutes to complete, and in each test, a number of attributes and the corresponding throughput performance are measured and recorded. A large number of measurements have been collected and archived in the past several years. The entire dataset consists of total 109,683 tabular data records, 30,433 of which are performance measurements of TCP tests and the rest (79,250 records) are performance measurements of UDT tests. Each data record contains a list of attributes, where the last one is the target attribute (i.e., performance) and the rest are roughly classified into three categories: i) end host configurations; ii) network connection properties; and iii) data transfer protocols/toolkits and their control parameters, as listed in Tab. I.

V. EXPLORATORY ANALYSIS

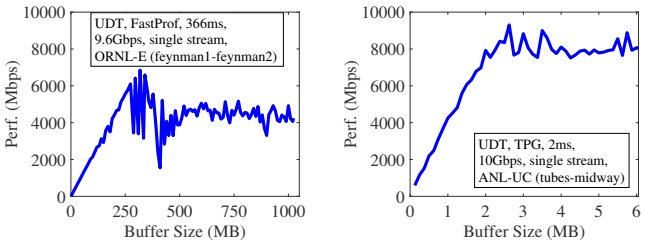
We conduct an exploratory analysis to identify the patterns of big data transfer performance with respect to the attributes as listed in Tab. I.

A. Effects of Application-Accessible Parameters

We focus on the set $\mathcal{C}$ of control parameters that are accessible and tunable at the application level, including packet size, block size, buffer size, and number of parallel streams. We analyze their impacts on throughput performance using both UDT and TCP measurements. To better illustrate the impacts of the parameter being examined, we set other parameters with values that do not cause significant interference.

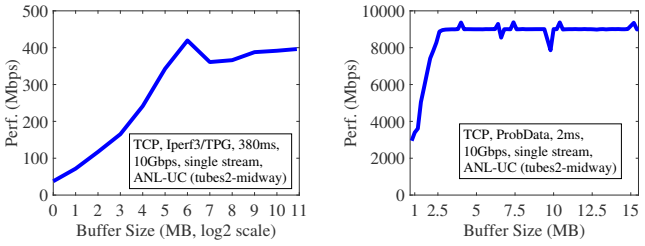
1) Packet Sizes

The throughput performance of big data transfer is affected by the packet size used in the underlying transport protocol. A larger packet size typically increases the performance since it carries more per-packet user payload and reduces per-packet processing overhead. Fig. 1(a) plots the UDT performance corresponds to different packet sizes in network connections with various RTTs and shows that large packet sizes do help improve the UDT performance. When other parameters such as buffer size are fixed, the performance almost linearly increases with packet size (Figs. 1(a) and 1(b)). If the buffer size is fixed at a value that limits performance, e.g., 256 MB, for a 10 Gbps connection of 200+ ms RTT, as illustrated by the 366 ms curve in Fig. 1(a), the performance curve also increases linearly with packet size, but at a slower speed for a given packet size in comparison with other cases where buffer size is sufficiently large. The increasing pattern of performance in response to packet size is observed with different RTTs on different testbeds between different hosts using different



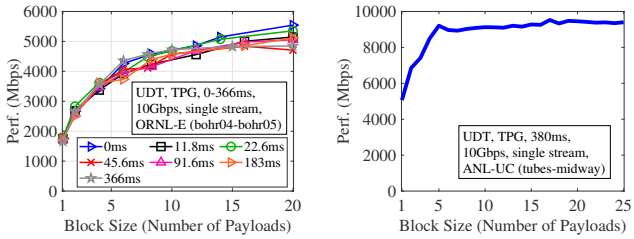
(a) ORNL-E, 366 ms, 9.6 Gbps (b) ANL-UC, 2 ms, 10 Gbps

Fig. 3. **UDT** performance vs. **buffer size**. Performance results are collected from different testbeds with single-stream data transfer tests.

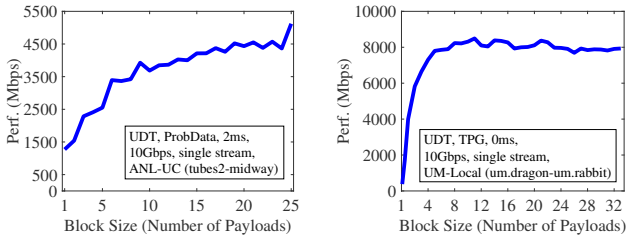


(a) ANL-UC, 380 ms, 10 Gbps (b) ANL-UC, 2 ms, 10 Gbps

Fig. 4. **TCP** performance vs. **buffer size**. Performance results are collected from different testbeds with single-stream data transfer tests.



(a) ORNL-E, 0–366 ms, 10 Gbps (b) ANL-UC, 380 ms, 10 Gbps



(c) ANL-UC, 2 ms, 10 Gbps (d) UM-Local, 0 ms, 10 Gbps

Fig. 5. **UDT** performance vs. **block size**. Performance results are collected from different testbeds with single-stream data transfer tests.

protocols, as shown in Figs. 1(b), 2(a), and 2(b), and is also consistent with the analysis of buffer size in Sec. V-A2.

2) Buffer Sizes

It has been well recognized that a buffer size no less than the Bandwidth-Delay Product (BDP) is required to saturate the connection capacity. For example, ESnet [1] recommends a TCP buffer size of twice of BDP to fill the pipe. While the specific behaviors and peak achievable performances of different protocols with respect to buffer size may be different, the shape of the performance curve is consistent and shows a "piecewise" pattern. Fig. 3 shows UDT performance with respect to buffer size, where the results are collected on different testbeds using different toolkits between different end hosts. Given a dedicated connection, the piecewise pattern is summarized as the following three regions:

In the region where buffer size is relatively small (e.g., less than BDP), TCP and UDT behave similarly and the performance almost linearly increases as buffer size increases. The peak achievable performance is mainly limited by buffer size and is lower than the overall peak achievable performance; the slope of the linear increase depends on end host configurations and network properties, which can be interpreted by comparing Fig. 3 and Fig. 4.

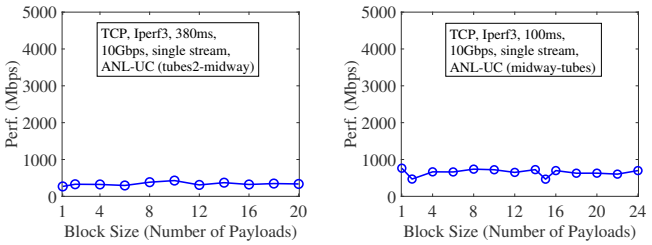
As buffer size increases up to around the BDP, both TCP and UDT reach the overall peak performance, and at this stage, other factors start to impose further limitation on the performance, as shown in Fig. 3. Although the specific buffer size for maximal achievable performance is "agnostic" as other factors such as RTTs play a more important role in such cases, the peak performance is usually achieved around the BDP, as illustrated in Figs. 3 and 4.

In the region where buffer size is larger than the BDP, TCP and UDT significantly diverge and the performance may not stay at the overall peak as buffer size keeps increasing beyond the BDP. In general, single-stream TCP can easily achieve near-capacity performance over a connection with a short delay (e.g., see Fig. 4(b)) and is not significantly affected by buffer size as long as it is not deliberately small. As shown in Fig. 4(a), single-stream TCP appears to be ineffective for long-haul connections, where other parameters such as number of streams and network properties such as RTT play a more dominating role (see Sec. V-A4 and Sec. V-B for more discussions). The case of UDT when buffer size is around or larger than BDP is more complicated as a larger buffer may degrade the achievable performance in certain environments, as shown in Fig. 3(a), which, compared with the case of TCP, is counter-intuitive.

3) Block Sizes

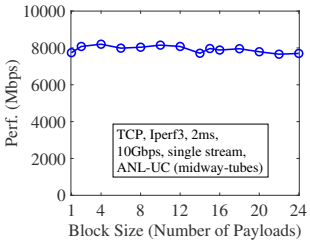
In general, with a sufficiently large buffer, the UDT performance increases as block size increases. The performance curve first shows a certain "concave" shape, indicating that the improvement brought by enlarging data block becomes marginal, and then stabilizes at the peak performance after block size reaches a certain point. The optimal block size is also prone to many other factors, but the performance pattern corresponding to block size appears to be consistent across different testbeds with different end hosts and network properties using different toolkits, as shown in Fig. 5. The performance suffers from a relatively limited buffer due to an overly large block size, e.g., when block size approaches buffer size, which, although not illustrated in Fig. 5, could happen in practice. In such cases, the performance decreases if the block size keeps increasing, as reported in [4].

The performance of TCP is not significantly affected by block size (see Fig. 6) and the stabilized performance is mainly

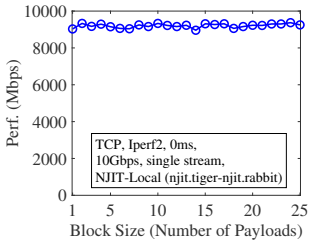


(a) ANL-UC, 380 ms, 10 Gbps

(b) ANL-UC, 100 ms, 10 Gbps

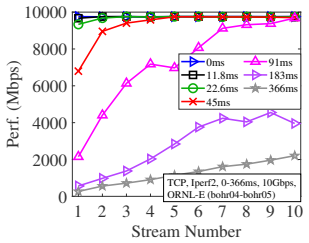


(c) ANL-UC, 2 ms, 10 Gbps

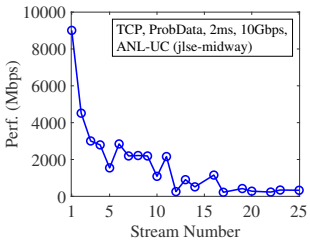


(d) NJIT-Local, 0 ms, 10 Gbps

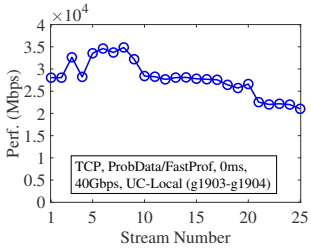
Fig. 6. **TCP** performance vs. **block size**. Performance results are collected from different testbeds with single-stream data transfer tests.



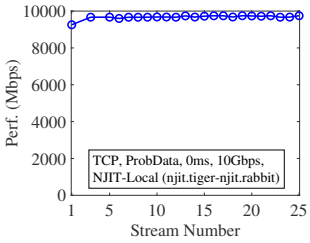
(a) ORNL-E, 0–366 ms, 10 Gbps



(b) ANL-UC, 2 ms, 10 Gbps



(c) UC-Local, 0 ms, 40 Gbps



(d) NJIT-Local, 0 ms, 10 Gbps

Fig. 7. **TCP** performance vs. **number of streams**. Performance results are collected from different testbeds with fixed-buffer data transfer tests.

determined by other factors such as buffer size (Sec. V-A2), RTT (Sec. V-B), and number of streams (Sec. V-A4).

4) Number of Parallel Streams

In our data transfer tests, we observe that UDT performance is not significantly affected by the number of parallel streams used in data transfer applications. This observation matches our expectation because UDT is not designed for environments with high concurrency [8]. This is also verified in some other work, e.g., [20].

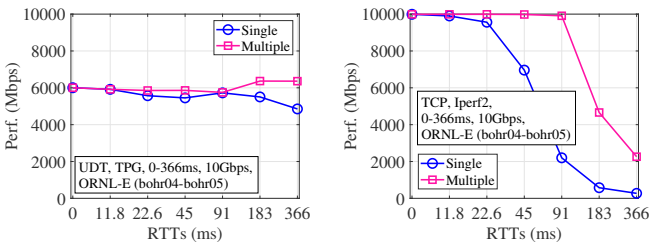
The incapability of standard TCP to achieve a steady-state high throughput over LFN connections is well recognized in networking community [7], and many TCP-based solutions using multiple streams such as GridFTP [17] have been developed and achieved remarkable success in practice. In our experiments, single-stream TCP achieves near-capacity throughput performance over short dedicated connections, as shown in Fig. 7(d), but suffers from long-haul connections, where using multiple streams can help achieve higher performance, as shown in Fig. 7(a).

Despite introducing inter-stream competition and extra processing overhead to the end hosts that may lead to complex transfer dynamics, multi-stream TCP solutions (e.g., GridFTP) are widely used in practice (e.g., for moving big datasets generated by large-scale scientific applications) as they can significantly increase end-to-end throughput performance especially over long-haul connections and between end hosts equipped with multiple bound 10 GigE NICs. As shown in Fig. 7(a), single-stream TCP only utilizes 20% of the bandwidth over longer connections (e.g., RTT $>$100 ms), while multi-stream TCP is able to reach up to 100%. It is difficult to directly derive the optimal number of streams for a data transfer over a given connection in a given network environment. Existing solutions adjust the number of streams based on runtime information and system load approximation [17]. The performance increase pattern as the number of streams increases is conceptually consistent among different pairs of end hosts in different network environments as long as it is not excessively large to overwhelm end hosts. Figs. 7(b) and 7(c) also collectively show that increasing the number of streams may decrease performance (due to extra overhead incurred by multiple streams) after achieving the peak performance.

B. Effects of Network Connection Properties

A dedicated connection has two important properties: i) the reserved bandwidth, which is typically on the order of tens of Gbps currently and Tbps in the forthcoming future; and ii) the connection delay, which is usually measured by the RTT ranging from less than one millisecond for back-to-back connections up to several hundred milliseconds for inter-continental connections. Unlike the control parameters in Sec. V-A, such properties are generally not tunable for a given dedicated connection, and they affect the performance in a different way from the control parameters: the reserved bandwidth provides a theoretical upper bound for the performance of big data transfer conducted over the connection, and the connection delay significantly affects the application-level maximal achievable performance, i.e., the upper bound for performance tuning.

The connection delay has a critical impact on TCP-based data transfer as it not only affects the choice of parameter values (e.g., buffer sizes), but also the maximal achievable performance between the same pair of end hosts. As shown in Figs. 4(a) and 4(b), from tubes2 to midway, even if buffer size is sufficiently larger than BDP, single-stream TCP only obtains a throughput of $<$5% available bandwidth when RTT is 380 ms; if the connection between the same two hosts is of shorter delay, e.g., 2 ms, single-stream TCP achieves a performance of around 90% available bandwidth. Other interesting effects of connection delay can also be derived



(a) UDT (b) TCP

Fig. 8. Maximal achievable performance of **UDT** and **TCP** vs. **RTTs**. Performance results are collected from ORNL-E between hosts bohr04 and bohr05 using (a) TPG UDT tests; and (b) iperf2 TCP tests.

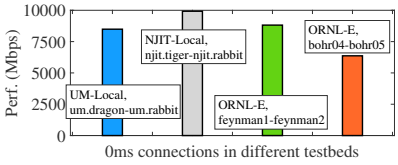


Fig. 9. Maximal achievable performance of **UDT** over four back-to-back connections (RTT $\approx$ 0 ms) between different pairs of end hosts on different testbeds.

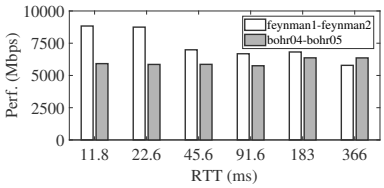


Fig. 10. Maximal achievable performance of **UDT** over emulated connections with various RTTs of $\{11.8, 22.6, 45.6, 91.6, 183, 366\}$ ms between different pairs of end hosts on ORNL-E.

from the performance comparison between the same pair of end hosts on the same testbed with different RTTs, as shown in Figs. 1(a), 5(a), and 7(a).

Fig. 8 plots the maximal achievable performance corresponding to different RTTs based on TPG UDT tests and iperf2 TCP tests over a 10 Gbps dedicated connection established between end hosts bohr04 and bohr05 with various emulated RTTs on ORNL-E. Firstly, the throughput of both UDT and TCP varies and generally decreases (especially for the single stream case) as RTT increases, and both TCP- and UDT-based data transfer requires certain tuning efforts to achieve good performance since the default settings typically do not achieve satisfactory performance especially over connections with long RTTs (e.g., $>$ 90 ms); secondly, single-stream TCP works well with short RTTs (e.g., $<$ 20 ms) and suffers from long RTTs (e.g., $>$ 50 ms), and using multiple TCP streams helps achieve near-connection capacity performance over mid-range connections with RTTs up to 90 ms, as shown in Fig. 8(b); thirdly, UDT is not as sensitive to RTTs as TCP due to its Decreasing Additive Increase and Multiplicative Decrease (DAIMD) rate control algorithm [20], the maximal achievable performance of UDT-based data transfer is more stable than TCP across different RTTs, and using multiple UDT streams only slightly helps for longer RTTs (e.g., $>$ 180 ms), as shown in Fig. 8(a); lastly, comparing Fig. 8(a) and Fig. 8(b), we observe that: i) single-stream TCP outperforms UDT for short RTTs but fails to keep up with UDT for both mid-range and long RTTs; ii) using multiple streams helps TCP outperform UDT for mid-range RTTs; and iii) UDT outperforms TCP in both single- and multi-stream cases for longer RTTs.

C. Effects of End Hosts Configurations

The effects of end host configurations are arguably intuitive. For high-speed data transfer in HPNs, it is important to ensure that both ends are capable of keeping up with the speed of incoming/outgoing traffics. In addition, the systems must be also appropriately configured to operate at 10 Gbps, 40 Gbps, or higher speed, including CPU governing, kernel buffer sizes, core affinities, packet pacing, etc. Furthermore, different environments and different transport methods may require different hardware capacities and software configurations to achieve optimal performance. For example, CPU frequency is critical for a single data flow to reach peak performance; careful packet pacing should be used for data transfer from a faster sender to a slower receiver over a long-haul dedicated connection (e.g., RTT $>$ 50 ms) when using tools augmented by parallel data streams such as GridFTP; TCP buffer size should be configured to be the maximum in Linux to maximize the transport performance over high-speed long-haul connections between end hosts equipped with 10/40/100 Gbps Ethernet NICs.

Such complexities that exist in host hardware/software configurations and time-varying system workloads/dynamics collectively make it non-trivial to tune and predict big data transfer performance in HPNs. They, together with network properties, impose an upper bound on the achievable throughput using different transport methods. Fig. 9 compares the maximal achievable performance of UDT over four different connections of 10 Gbps with a close-to-zero RTT, established via back-to-back connecting four different pairs of hosts on different testbeds (um.dragon to um.rabbit on UM-Local, njit.tiger to njit.rabbit on NJIT-Local, and feynman1 to feynman2 and bohr04 to bohr05 on ORNL-E). Fig. 10 shows the performance difference over connections with different RTTs emulated between the same two pairs of end hosts (feynman1 to feynman2 and bohr04 to bohr05) on ORNL-E. They both show that similar or identical connections between different end hosts may result in very different maximal performance achievable by "near-exhaustive" performance tuning.

VI. Performance Prediction Using Machine Learning

We first show that $\hat{y}(\mathbf{x})$ is indeed a good estimate of big data transfer performance over dedicated connections in a statistical sense with a high probability, and then present prediction results of Support Vector Regression (SVR), a popular machine learning algorithm for multivariate regression.

A. Confidence Estimates

The throughput performance $y(\mathbf{x})$ is a response variable with a complex distribution $\mathbf{P}_{y(\mathbf{x})}$ as it depends on many fac-

tors including: i) end host system configurations and dynamics, ii) network connection properties and randomness, and iii) data transfer applications and their underlying protocols (control parameter values, congestion control mechanisms, etc.). We define the performance regression as the following expectation

$$\overline{y}(\mathbf{x}) = E\left[y(\mathbf{x})\right] = \int y(\mathbf{x})\, d\,\mathbf{P}_{y(\mathbf{x})},$$

which can be estimated based on experimental performance measurements $y(\mathbf{x}_k, t_i^k)$ at $\mathbf{x}_k$ $(k = 1, 2, \ldots, n)$ and time t_i^k $(i = 1, 2, \ldots, n_k)$. We have $0 \leq y(\mathbf{x}_k, t_i^k) \leq B$ due to the reserved bandwidth B of a dedicated connection. The performance estimate $\hat{y}(\mathbf{x}_k)$, given by its *empirical mean*, is computed using measurements as

$$\hat{y}(\mathbf{x}_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} y(\mathbf{x}_k, t_i^k),$$

at $\mathbf{x}_k$'s in the space of attribute vector $\mathbf{x} = [\mathcal{H}, \mathcal{P}, \mathcal{C}]$. Note that $\hat{y}(\mathbf{x}_k)$ is computed completely based on performance measurements, and is indicative of the actual performance at $\mathbf{x}_k$, whose *unknown* expected value is $\overline{y}(\mathbf{x}_k)$ and is to be estimated. We show that $\hat{y}(\mathbf{x}_k)$ is indeed a good estimate of $\overline{y}(\mathbf{x}_k)$, in terms of the estimation of expected error, and furthermore, the prediction accuracy improves with more performance measurements, regardless of the underlying distribution $\mathbf{P}_{y(\mathbf{x})}$.

Consider an estimate $g(\cdot)$ of $\overline{y}(\cdot)$ based on performance measurements from a class $\mathcal{F}$ of unimodal functions bounded in $[0, B]$, i.e., $0 \leq g \leq B, g \in \mathcal{F}$. The *expected quadratic loss* $I(g)$ of the estimator g is

$$I(g) = \int \Big[g(\mathbf{x}) - y(\mathbf{x}, t)\Big]^2 \, d\,\mathbf{P}_{y(\mathbf{x},t)},$$

and the *best estimator* g^* is given by $I(g^*) = \min_{g \in \mathcal{F}} I(g)$. The *empirical error* of g based on performance measurements is given by

$$\hat{I}(g) = \frac{1}{n} \sum_{k=1}^{n} \left\{ \frac{1}{n_k} \sum_{i=1}^{n_k} \Big[g(\mathbf{x}_k) - y(\mathbf{x}_k, t_i^k)\Big]^2 \right\},$$

and the *best empirical estimator* $\hat{g^*} \in \mathcal{F}$ minimizes the empirical error, i.e.,

$$\hat{I}(\hat{g^*}) = \min_{g \in \mathcal{F}} \hat{I}(g).$$

Since $\hat{y}(\mathbf{x}_k)$ is the response mean at each attribute vector $\mathbf{x}_k$, it achieves the minimal empirical error.

Since both $y(\cdot)$ and $g(\cdot)$ are bounded in $[0, B]$, $I(g)$ is also bounded in $[0, K]$ with some $K > 0$. Let $\mathcal{I} = \{I(g) \,|\, g \in \mathcal{F}\}$ be the set of loss functions subject to $\mathcal{F}$. Based on the uniform convergence results of Vapnik-Chervonenkis theory [19] and its generalization [3] and applications (e.g., [18]), we know that, for some $\epsilon > 0$,

$$\begin{aligned} &P\left\{I(\hat{y}) - I(g^*) > 2\epsilon\right\} \\ &\quad \leq P\left\{\sup_{h \in \mathcal{F}} \Big|I(h) - \hat{I}(h)\Big| > \epsilon\right\}, \end{aligned}$$

and furthermore, according to [19], we have

$$\begin{aligned} &P\left\{\sup_{h \in \mathcal{F}} \Big|I(h) - \hat{I}(h)\Big| > \epsilon\right\} \\ &\quad \leq 18 \mathcal{N}_1(\frac{\epsilon}{K}, \mathcal{I}, n) \cdot n \cdot \exp\left(-\frac{n\epsilon^2}{4K^2}\right), \end{aligned}$$

where $\mathcal{N}_1(\frac{\epsilon}{K}, \mathcal{I}, n)$ is the ϵ-cover of $\mathcal{I}$ under d_1 norm.

Since $\mathcal{I}$ satisfies Lipschitz condition, suppose that its Lipschitz constant is $L > 0$, we then have

$$\mathcal{N}_1(\frac{\epsilon}{K}, \mathcal{I}, n) \leq \mathcal{N}_1(\frac{\epsilon}{KL}, \mathcal{F}, n).$$

Also, for any class $\mathcal{M}$ of real-valued functions, any $\delta > 0$ and any $j \in \mathbb{N}$, we have $\mathcal{N}_1(\delta, \mathcal{M}, j) \leq \mathcal{N}_\infty(\delta, \mathcal{M}, j)$ [3]. It follows that

$$\begin{aligned} &P\left\{I(\hat{y}) - I(g^*) > 2\epsilon\right\} \\ &\quad \leq 18 \mathcal{N}_\infty(\frac{\epsilon}{KL}, \mathcal{F}, n) \cdot n \cdot \exp\left(-\frac{n\epsilon^2}{4K^2}\right), \end{aligned}$$

where $\mathcal{N}_\infty(\epsilon, \mathcal{F})$ is the ϵ-cover of $\mathcal{F}$ under d_∞ norm. Due to the unimodality of functions in $\mathcal{F}$, their *total variation* is upper-bounded by $2B$, which provides us the following upper bound [3],

$$\mathcal{N}_\infty(\frac{\epsilon}{KL}, \mathcal{F}, n) < 2\left(\frac{4K^2L^2n}{\epsilon^2}\right)^{\left(1+\frac{4BKL}{\epsilon}\right)\log_2\left(\frac{en}{B}\right)}.$$

By using this bound, we obtain

$$\begin{aligned} &P\left\{I(\hat{y}) - I(g^*) > \epsilon\right\} \\ &\leq 36\left(\frac{16K^2L^2n}{\epsilon^2}\right)^{\left(1+\frac{8BKL}{\epsilon}\right)\log_2\left(\frac{en}{B}\right)} \cdot n \cdot \exp\left(-\frac{n\epsilon^2}{16K^2}\right). \end{aligned}$$

The exponential term on the right-hand side decays faster in n than other terms, and hence for sufficiently large n, it would be smaller than a given probability. In sum, the expected error $I(\hat{y})$ of the response mean is within ϵ of the optimal error $I(g^*)$ with a probability that increases with the number of performance measurements. This performance guarantee is independent of the complexity of $\mathbf{P}_{y(\mathbf{x})}$. Thus, $\hat{y}(\mathbf{x})$ is a good estimate of the actual throughput performance achievable at feature $\mathbf{x}$ independent of the underlying distribution.

B. Performance Prediction using SVR

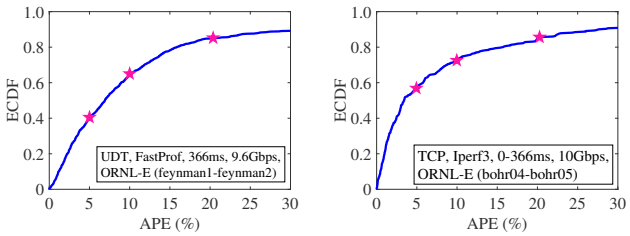
1) Experimental Settings and Results

We implement an SVR-based predictor based on the Scikit-learn library [6], where the radial basis function (RBF) kernel and an error-tolerated tube with a 0.005 radius are used during training. Grid search (with 10-fold cross validation) is performed to find the best hyperparameters with kernel coefficient set $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.5, 0.6, 0.9\}$ and regularization parameter set $\{1, 10, 100, 1000, 10000\}$. We measure the performance of the predictor by calculating its prediction accuracy in terms of Absolute Percentage Error (APE) defined as $\frac{|\hat{y}_i - y_i|}{y_i} \times 100\%$, where y_i is the actual performance, i.e., true value, $\hat{y}_i$ is the predicted value.

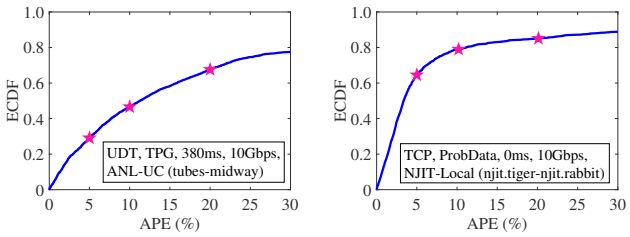
The Empirical Cumulative Distribution Function (ECDF) of performance prediction accuracy corresponding to both UDT and TCP on different HPN testbeds are plotted in Fig. 11. Figs 11(a), 11(b), and 11(d) show that the SVR-based predictor achieves 10% APE roughly among 70% to 80% of all test cases for both TCP and UDT on ORNL-E and NJIT-Local testbeds. These results are obtained based on the "raw" datasets without any noise reduction or other data preprocessing, they confirm that the selected features based on our exploratory analysis in Sec. V are indeed of high predictive power. The worst results are with the UDT measurements over the 380 ms connection on ANL-UC (Fig. 11(c)), where a 10% APE is achieved around 50% of the time among all test cases.

2) Eliminating Negative Effects of Unknowns

While the end host configurations do have certain impact on performance, other unpredictable factors such as system dy-



(a) ORNL-E, UDT (b) ORNL-E, TCP



(c) ANL-UC, UDT (d) NJIT-Local, TCP

Fig. 11. Performance prediction results of SVR without data preprocessing.

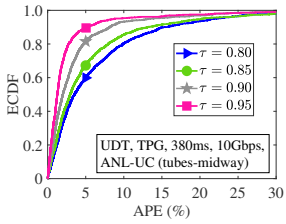


Fig. 12. Performance prediction results of SVR with data preprocessing using the same dataset as in Fig. 11(c).

namics and competing workloads may also affect performance. These factors, however, are not conveniently measurable at the application level during experiments. In other words, there exist latent attributes that are independent of $\mathcal{H}$, $\mathcal{P}$, and $\mathcal{C}$, but are not easily observable. Such factors may cause large variations among multiple performance observations for a given $\mathbf{x}_i$ and eventually lead to inaccurate performance prediction as in Fig. 11(c), where the end hosts (tubes and midway) are simultaneously used by other scientists running their scientific computing jobs during our data transfer experiments.

Our ultimate goal is to avoid any excessive bandwidth reservation beyond actual needs. To meet such a bandwidth requirement for a data transfer request, we need to ensure the reserved (i.e., predicted) bandwidth is around the maximal achievable performance for a given $\mathbf{x}_i$. We employ a simple threshold-based method to eliminate the negative effects of latent attributes on performance prediction by excluding the "abnormal" data points whose performances are below a threshold τ of the corresponding achievable maximum. In particular, if there are multiple measurements for a given $\mathbf{x}_i$, those with a performance y_i below $\tau \cdot \max_i \{y_i\}$ $(0 < \tau < 1)$ are discarded. Incorporating this into data preprocessing, we conduct performance prediction using the same dataset as in Fig. 11(c) and present results in Fig. 12, which shows that the prediction accuracy is significantly improved across different values of $\tau \in \{0.80, 0.85, 0.90, 0.95\}$, e.g., the 5% error percentile is increased from 30% to 90% with $\tau = 0.95$.

VII. Conclusion and Future Work

We conducted an in-depth exploratory analysis of the impacts of a comprehensive set of factors on the end-to-end performance of big data transfer based on extensive performance measurements collected on several real-life physical or emulated HPN testbeds. Based on such analysis, we selected features and built a performance predictor using machine learning. We verified the feasibility and effectiveness of the learning-based performance predictor through theoretical performance bound analysis. The experimental results show that, with appropriate data preprocessing, the predictor is able to achieve satisfactory accuracy based on very noisy datasets.

We plan to use "advanced" bagging- and boosting-based machine learning algorithms to perform such prediction and compare their performance. It is also of our interest to study and derive tighter performance bounds on the estimation loss and sample size by incorporating other HPN domain insights.

Acknowledgment

This research is sponsored by Harrisburg University and the U.S. National Science Foundation under Grant No. CNS-1828123 with New Jersey Institute of Technology.

References

[1] ESnet. http://www.es.net.
[2] Iperf3. https://bit.ly/2rpe6SW.
[3] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, 2009.
[4] D. Yun *et al.* Profiling transport performance for big data transfer over dedicated channels. In *Proc. of ICNC*, pages 858–862, 2015.
[5] D. Yun *et al.* Advising big data transfer over dedicated connections based on profiling optimization. *IEEE/ACM Trans. Netw.*, 27(6):2280–2293, 2019.
[6] F. Pedregosa *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
[7] S. Floyd. Highspeed TCP for large congestion windows. RFC 3649.
[8] Y. Gu and R. Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Comput. Netw.*, 51(7):1777–1799, 2007.
[9] S. Ha, I. Rhee, and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operat. Syst. Rev.*, 42(5):64–74, 2008.
[10] J. Padhye *et al.* Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145, 2000.
[11] T. Kelly. Scalable TCP: Improving performance in highspeed wide area networks. *ACM SIGCOMM Comput. Commun. Rev.*, 33(2):83–91, 2003.
[12] B. Leitao. Tuning 10Gb network cards on Linux. In *Proc. of Linux Symp.*, pages 169–184, 2009.
[13] M. Mirza *et al.* A machine learning approach to TCP throughput prediction. *IEEE/ACM Trans. Netw.*, 18(4):1026–1039, 2010.
[14] N. Hanford *et al.* Analysis of the effect of core affinity on high-throughput flows. In *Proc. of NDM*, pages 9–15, 2014.
[15] N. Rao *et al.* TCP throughput profiles using measurements over dedicated connections. In *Proc. of HPDC*, pages 193–204, 2017.
[16] Q. Liu *et al.* Measurement-based performance profiles and dynamics of UDT over dedicated connections. In *Proc. of ICNP*, 2016.
[17] R. Kettimuthu *et al.* An elegant sufficiency: Load-aware differentiated scheduling of data transfers. In *Proc. of SC*, Article 46, 2005.
[18] N. Rao. Simple sample bound for feedforward sigmoid networks with bounded weights. *Neurocomputing*, 29(1):115–122, 1999.
[19] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.
[20] Y. Gu *et al.* An analysis of AIMD algorithm with decreasing increases. In *Proc. of the 1st Int'l Workshop on Netw. for Grid Appl.*, 2004.
[21] Z. Liu *et al.* Explaining wide area data transfer performance. In *Proc. of HPDC*, pages 167–178, 2017.
[22] Z. Liu *et al.* Building a wide-area data transfer performance predictor: An empirical study. In *Proc. of the 1st Int'l Conf. on Machine Learning for Netw.*, 2018.