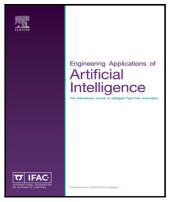
Engineering Applications of Artificial Intelligence 102 (2021) 104285

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Exploratory analysis and performance prediction of big data transfer in High-performance Networks☆

Daqing Yun [a], Wuji Liu [b], Chase Q. Wu [b,*], Nageswara S.V. Rao [c], Rajkumar Kettimuthu [d]

[a] Computer and Information Sciences Program, Harrisburg University, Harrisburg, PA 17101, USA
[b] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
[c] Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[d] Data Science and Learning Division, Argonne National Laboratory, Lemont, IL 60439, USA

ARTICLE INFO

Keywords:
Performance prediction
Latent effect
Machine learning
Big data transfer

ABSTRACT

Big data transfer in large-scale scientific and business applications is increasingly carried out over connections with guaranteed bandwidth provisioned in High-performance Networks (HPNs) via advance bandwidth reservation. Provisioning agents need to carefully schedule data transfer requests, compute network paths, and allocate appropriate bandwidths. Such reserved bandwidths, if not fully utilized, could be simply wasted due to the exclusive access during the approved time window, and cause extra overhead and complexity for resource management. This calls for accurate performance prediction to reserve bandwidths that match actual needs and avoid over-provisioning. We employ machine learning algorithms to predict big data transfer performance based on extensive performance measurements collected in the past several years from data transfer tests using different protocols and toolkits between various end sites on several real-life physical or emulated testbeds. We first analyze the performance patterns in response to a comprehensive list of parameters in end-host systems, network connections, and data transfer applications, which motivate the use of machine learning and also help us identify the effects of latent factors. We then propose threshold- and clustering-based methods to eliminate negative effects of latent factors in data preprocessing and build a robust performance predictor based on customized domain-oriented loss functions. The performance of the proposed methods is verified by extensive experiments using SVR and RFR as well as theoretical analysis of the general performance bound.

1. Introduction

High-speed network connections with guaranteed bandwidth provisioned in High-performance Networks (HPNs) such as ESnet (ESnet, 2021), Internet2 (Internet2, 2021), XSEDE (XSEDE, 2021), and Google's SDN (Jain et al., 2013) are increasingly used for big data transfer in support of applications in various domains ranging from extreme-scale scientific research to industrial big data analytics. Provisioning agents (e.g., OSCARS Guok et al., 2006; OSCARS, 2021) typically ask users to request bandwidth as needed in advance and then establish end-to-end network paths with reserved bandwidths. Such network paths are comprised of a sequence of end-host systems, edge switches/routers, core switches/routers, and physical circuits or lightpaths, which are typically time-shared among geographically distributed users, hence resulting a high level of topological and temporal complexity in resource sharing. Therefore, efficient resource scheduling is needed to cope with such complexity for bandwidth reservation to improve HPN resource utilization and user satisfaction.

The predictability of end-to-end big data transfer performance (mainly throughput) is critical to the scheduling and planning of HPN resources. An on-demand "bandwidth-guaranteed" connection, once allocated and granted, is used exclusively by the requesting user during the approved time window. Due to the nature of exclusive access, the reserved bandwidth, if not fully utilized, could be wasted during the approved time window, and cause extra overhead for HPN resource management. Therefore, accurate performance prediction is not only useful for end users to design and optimize their strategies for satisfactory data transfer performance, e.g., determining what transport methods to use and what parameter values to set (Yun et al., 2016), but also important for HPN management to wisely schedule data transfer requests for better resource utilization, e.g., rejecting requests with "over-claimed" bandwidth demands or granting an appropriate amount of bandwidths that could be actually utilized.

However, predicting the performance of big data transfer in HPNs is challenging. Although the exclusive use of HPN connections minimizes

☆ Some preliminary results in this manuscript were presented at IFIP Networking 2020 (Yun et al., 2020) and IEEE ICC 2020 (Liu et al., 2020).

* Corresponding author.
E-mail addresses: dyun@harrisburgu.edu (D. Yun), wl87@njit.edu (W. Liu), chase.wu@njit.edu (C.Q. Wu), raons@ornl.gov (N.S.V. Rao), kettimut@anl.gov (R. Kettimuthu).

https://doi.org/10.1016/j.engappai.2021.104285
Received 23 November 2020; Received in revised form 19 March 2021; Accepted 26 April 2021
Available online 4 May 2021

the impact of complex dynamics caused by some factors such as cross traffic, many other elements involved in a typical big data transfer process still affect the performance to a great extent, including (i) configurations of end host systems, (ii) properties of network connections, and (iii) control parameters of data transfer methods and their underlying transport protocols. It is generally very difficult to apply an analytical approach to big data transfer performance prediction, due to (i) the lack of accurate throughput performance models for high-performance transport protocols such as UDT (Gu and Grossman, 2007), (ii) the complex composition of end-to-end HPN connections, (iii) the complexities of end host configurations; (iv) the time-varying workloads in end host systems; and (v) other latent variables that may not even be accessible or measurable. Consequently, HPN technologies and services have not been fully utilized for big data transfer regardless of the continuous bandwidth upgrades in backbones.

Informally, we attempt to answer the following question: given a sender host, a receiver host, and a dedicated connection between them with (high) bandwidth reserved in advance, for a data transfer application with an underlying transport protocol and its control parameter values, what end-to-end application-level throughput performance could be achieved? In turn, the answer to the above question would help us determine how much bandwidth should be reserved and allocated to the corresponding data transfer request such that we could meet its bandwidth requirement without resource waste.

In this work, we develop an important capability of performance prediction of memory-to-memory big data transfer for HPN management to facilitate effective resource scheduling and planning. We focus on memory-to-memory big data transfer because it is critical to a wide range of scientific applications for various purposes such as collaborative computational steering among geographically distributed users as well as on-line analysis and visualization of scientific data generated on remote computing facilities. Furthermore, in most of the practical scenarios, disk-to-disk data transfer is bottlenecked by disk I/O speed, which is typically much slower than the speed of HPN connections. With memory-to-memory transfer, we are able to sustain high throughput over HPN connections and examine the behaviors of transport methods deployed on end hosts with heavy incoming/outgoing traffic.

We employ machine learning methods in both data preprocessing and model training based on comprehensive performance measurements that have been collected and accumulated in the past several years. These measurements are recorded from a large number of big data transfer tests that are conducted between various end sites on several real-life physical or emulated HPN testbeds, using different data transfer protocols and toolkits. These datasets carry very important information about the patterns and behaviors of existing transport methods in different HPN environments, and can be used to train machine learning models to shed light on performance optimization and prediction of big data transfer. Based on these performance measurements, we first identify a comprehensive list of attributes involved in a typical big data transfer process, including end host system configurations, network connection properties, control parameters of data transfer methods, and other unobservable latent factors. We then conduct qualitative and comparative exploratory analysis of the impacts of these attributes on end-to-end transport performance observed by end users at the application level. We propose latent effect elimination methods and incorporate them into data preprocessing, and further build a performance predictor using machine learning algorithms based on customized domain-oriented loss functions. We conduct experiments to illustrate the quality of our predictor and perform theoretical analysis to understand the applicability of machine learning methods to data transfer performance prediction in HPN environments.

We summarize our contributions in this work as follows.

- **Exploratory Analysis**. We conduct in-depth analysis of a comprehensive list of transport-related attributes to qualitatively explain their impacts on 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.
- **Latent Effect Elimination**. We show the (negative) effects of latent factors on performance prediction based on comparative experimental studies. Such latent factors are difficult to observe, predict, or estimate, and therefore may severely impair the accuracy of performance prediction models. We propose threshold- and clustering-based methods to eliminate such negative effects in data preprocessing and show that such elimination significantly improves the efficiency of model training and the accuracy of performance prediction.
- **Loss Function Customization**. Inspired by the domain knowledge of HPN resource management and the requirements of big data transfer requests from end users, we design a "one-side" ϵ-insensitive loss function specifically for the performance prediction of big data transfer to facilitate better bandwidth resource utilization in HPNs.

We evaluate the effectiveness of the proposed latent effect elimination and domain-oriented loss customization methods by incorporating them into the model training of a support vector regression (SVR)-based performance predictor and conducting experiments for performance evaluation in comparison with the default SVR method. We further compare the SVR-based predictor with another representative method, random forest regression (RFR). The experimental results show that the proposed methods not only achieve more effective performance predictions but also reduce resource waste. We also show that the performance predictor built on the "critical" features selected based on the exploratory analysis is statistically meaningful by deriving a performance bound with several domain-specific conditions incorporated.

The rest of the paper is organized as follows. In Section 2, we conduct a brief survey of existing work in related fields. Section 3 describes the problem of big data transfer performance prediction in HPNs. We introduce the performance measurement dataset used in this work in Section 4. An exploratory analysis of big data transfer performance is conducted in Section 5. In Section 6, we detail the proposed methods for threshold- and clustering-based latent effect elimination as well as loss function customization. In Section 7, we evaluate the performance of the proposed methods and provide a confidence analysis of our machine learning-based performance prediction by deriving theoretical performance bounds. We conclude our work and sketch a research plan in Section 8.

2. Related work

The importance of provisioning bandwidth over HPN connections to support big data movement over long distances has been well recognized in both science and networking communities. Many efforts have been devoted to achieving predictable transport performance and efficient resource utilization. We conduct a survey of such existing work.

2.1. Bandwidth scheduling

To support big data transfer, bandwidths need to be scheduled and allocated over network connections in HPNs with time-varying exclusive access. Bandwidth scheduling is usually formulated as optimization problems typically of NP-completeness, and many heuristics have been designed to optimize various objectives such as earliest available resource (Rao et al., 2006), minimal data transfer time (Lin and Wu, 2013), minimum end-to-end delay (Grimmell and Rao, 2003), minimal number of path switches (Gangulay et al., 2004), maximal resource utilization (Zuo et al., 2018), energy efficiency (Shu et al., 2013), etc. Due to the heuristic property, the solutions based on such service models have inherited limitations that can lead to inefficient resource utilization and unsatisfactory transport performance, hence making their applications and adoptions very limited in practice. Furthermore, these methods always make an unrealistic assumption that the reserved

bandwidth be fully utilized, which may cause severe resource waste due to bandwidth over-provisioning. This calls for an effective solution to accurately predict the actual *achievable* performance of data transfer over HPN connections with guaranteed bandwidth to overcome the inherent limitations of advance bandwidth reservation.

2.2. Transport profiling and optimization

Many profiling-oriented toolkits have been developed for conducting data transfer tests to help understand and optimize transport performance. For example, iperf2 (Iperf2, 2021) is a handy tool available in most Linux distributions to run TCP tests for bandwidth estimation. ESnet iperf3 (Iperf3, 2021) is a toolkit for actively measuring the maximum achievable performance along an end-to-end network path via continuous data transfer tests. It is a rewrite of iperf2 and provides a rich set of functions and options for tuning various parameters of TCP, UDP, and SCTP, including packet size, block size, buffer size, and number of data streams. Similar to iperf2/3, TPG (Yun et al., 2015) is a toolkit for conducting data transfer tests using UDT (UDT, 2021a), another widely-used protocol (UDT, 2021b). In addition to the common tunable parameters in iperf2/3, TPG enables tuning of UDP socket options for UDT and other UDT-specific parameters (UDT, 2021c) (e.g., `UDT_MAXBW`, as detailed in UDT, 2021c), and also supports profiling tests over multiple NIC-to-NIC connections as a complement.

Some of these toolkits have been integrated into transport profiling optimization (e.g., FastProf Yun et al., 2016) and data transfer advising (e.g., ProbData Yun et al., 2019) as functional units to carry out data transfer tests guided by various optimization methods such as stochastic approximation (Spall, 2003) and time series analysis (Sapkota et al., 2019). Performance measurements are collected using these toolkits by measuring throughput in response to various attributes including end host system configurations, network connection properties, control parameter values, and other unobservable latent factors, and can be utilized to train performance models of data transfer for prediction.

2.3. Performance modeling and characterization

There are numerous efforts devoted to understanding the behaviors of transport protocols and modeling their throughput performance. For example, Padhye et al. in Padhye et al. (2000) developed a throughput model of TCP bulk data transfer corresponding to loss rate and Round-Trip Time (RTT). A TCP throughput profile in response to RTT and number of streams is presented in a more recent study (Rao et al., 2017). Gu et al. proposed a throughput model of UDT protocol in Gu et al. (2004a) and used experimental results over 1 Gbps connections to show that UDT recovers much faster than TCP from a loss event to reach 90% capacity of Long-Fat Network (LFN) connections. The model in Gu et al. (2004a) also shows that UDT achieves stable asymptotic performance across different RTTs due to its fixed control interval. Another experimental study of UDT (Liu et al., 2016), however, shows that UDT appears to have certain instability and variation over 10 Gbps connections with different RTTs and between end hosts with different configurations. These modeling and experimental studies focus on performing rigorous analyses of the behaviors of transport methods in response to the factors that represent real-time network conditions (e.g., loss rate, available bandwidth, etc.) without considering 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.

Instead of analytically modeling the behaviors of transport methods, machine learning learns the performance patterns with respect to the involved attributes based on historical data transfers and has great potential for performance prediction of big data transfers in HPNs. For example, He et al. in He et al. (2007) showed that a simple predictor using moving average based on a few historical samples provides accurate prediction under certain conditions. In Mirza et al. (2010), Mirza et al. used a support vector regression method to predict TCP file transfer performance in publicly shared Internet environments, using path properties and file transfer sizes as the features. Liu et al. in Liu et al. (2017) used 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 built a performance predictor in Liu et al. (2018). Our work differs from the aforementioned efforts in that (i) we focus on memory-to-memory data transfer over network connections with guaranteed bandwidth up to several tens of Gbps, and (ii) we identify a comprehensive list of attributes involved in a typical data transfer process, including not only connection properties and end host configurations, but also control parameters of data transfer applications and their underlying protocols as well as unobservable latent factors. We conduct an exploratory analysis of their impacts on the end-to-end application-level throughput performance of big data transfer and investigate the applicability of machine learning to performance prediction in HPNs.

3. Problem statement

End-to-end data transfer is a complex process that involves various components in both network segments and end hosts that may affect application-level throughput. Among these components, some can be accessed and controlled by user applications, including packet size, block size, buffer size, and the number of data streams, while others are mainly determined by the hardware and system (kernel) configurations as well as network infrastructures, including CPU frequency, memory size and bandwidth, bus speed, disk I/O speed, path MTU, RTT, link bandwidth, and loss rate.

Although network connections reserved in advance with guaranteed bandwidth have some relatively stable properties such as RTT and connection loss rate, 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 Tierney, 2016), the use of an unsuitable data transfer protocol (Yu et al., 2015), incorrect settings of CPU affinity (Hanford et al., 2016) and IRQ balance/conflict (Leitao, 2009), and suboptimal control parameter values (Yun et al., 2016). Due to the quantity and complexity of these factors, it is dauntingly difficult to develop an analytical approach to predict transport performance, especially in the presence of latent factors such as system dynamics and competing workloads on end hosts.

Given two end hosts (a sender and a receiver) and a network connection with guaranteed bandwidth between them, our goal is to predict the maximal achievable end-to-end throughput performance of a given data transfer, using a specific data transfer tool and its underlying transport protocol together with their corresponding control parameter values. Such prediction can not only help end users understand and optimize transport performance, but also facilitate effective resource scheduling and planning for HPN management.

The end-to-end throughput performance y of a data transfer could be expressed as a function f of a set of variables including: (i) end host system configurations $\mathcal{H}$, (ii) network connection properties $\mathcal{P}$, and (iii) control parameters $\mathcal{C}$ of data transfer methods and their underlying transport protocols, which collectively form a feature vector $\mathbf{x} = [\mathcal{H}, \mathcal{P}, \mathcal{C}]$, i.e., $y = f(\mathbf{x})$. The analytical form of $f(\mathbf{x})$ 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 $y_i = f(\mathbf{x}_i) + \zeta_i + \xi(\mathbf{u}_i)$, where y_i is actually the "noise-corrupted" throughput observation, ζ_i is an i.i.d. random variable reflecting the noise such as inherent system dynamics and observation randomness, $\mathbf{u}_i$ represents the unpredictable and unobservable latent variables such as competing workloads on end hosts and other unknowns, and $\xi(\mathbf{u}_i) < 0$ represents the collective (negative) effects caused by $\mathbf{u}_i$.

The expected throughput performance $\overline{y}_i$ of a big data transfer during time interval $t \in [0, \Delta T]$ is given by

$$\overline{y}_i = \frac{\int_0^{\Delta T} y_i\left(\mathbf{x}_i, \mathbf{u}_i(t), t\right) dt}{\Delta T},$$

where $y_i(\mathbf{x}_i, \mathbf{u}_i(t), t)$ is the throughput at time point t in response to a specific feature vector $\mathbf{x}_i$ and latent variable $\mathbf{u}_i(t)$. Note that $\mathbf{u}_i(t)$ is time-dependent and is often unmeasurable in practice. For a given data transfer, we obtain $\overline{y}_i$ by approximating it with its corresponding ***observed*** throughput performance, i.e.,

$$\overline{y}_i \approx \tilde{y}_i = \frac{F_i(\mathbf{x}_i, \mathbf{u}_i)}{\Delta T},$$

where $F_i(\mathbf{x}_i, \mathbf{u}_i)$ is the total size of the user payload delivered by a data transfer with feature $\mathbf{x}_i$ under latent effects of $\mathbf{u}_i$ during the time window $[0, \Delta T]$, and ΔT is the time period used for completing the transfer. In this work, 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 attributes of the feature vector $\mathbf{x}$ have a large range of possible values in a multidimensional parameter 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 the performance space for prediction purposes, and we use machine learning to train such prediction models based on historical data transfers. More specifically, we collect a training dataset $\mathcal{T}$ that consists of n performance measurements of big 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 $\mathbf{x}$ that collectively determines the corresponding throughput performance y_i. We propose to employ machine learning-based 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. For convenience, we tabulate the notations used in this work in Table 1, some of which are also used in the loss function customization and general performance bound analysis.

Table 1
List of notations used in the paper.

Notations	Definitions
$\mathcal{H}$	Properties of end host
$\mathcal{P}$	Properties of network connection
$\mathcal{C}$	Control parameters
B	Connection bandwidth (reserved in HPN)
$\mathbf{x}$	Feature vector, $\mathbf{x} = [\mathcal{H}, \mathcal{P}, \mathcal{C}]$
$\mathbf{x}_i$	The ith feature vector
y	Data transfer performance
$f(\mathbf{x})$	y as a function of $\mathbf{x}$
y_i	Data transfer performance w.r.t. $\mathbf{x}_i$
$\overline{y}_i$	The expected data transfer performance w.r.t. $\mathbf{x}_i$
$\tilde{y}_i$	The observed data transfer performance w.r.t. $\mathbf{x}_i$
ζ_i	i.i.d. random noise in measurements
$\mathbf{u}$	Latent variable vector
$\mathbf{u}_i$	The ith latent variable vector
$\xi(\mathbf{u}_i)$	The collective (negative) effects caused by $\mathbf{u}_i$
$\mathcal{T}$	Performance measurement dataset
$\{y'\}$	Performance measurements without latent effects
$\{y''\}$	Performance measurements with latent effects
τ	Latent effect elimination threshold
θ	Parameter set of prediction model
ϵ	Error tolerance in the customized loss function
$\mathcal{L}(\theta, \epsilon)$	The customized loss function
$\hat{y}_i^{\theta}$	The predicted performance w.r.t. $\mathbf{x}_i$ and θ
λ, L, K	All are positive constants

4. Performance measurements

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

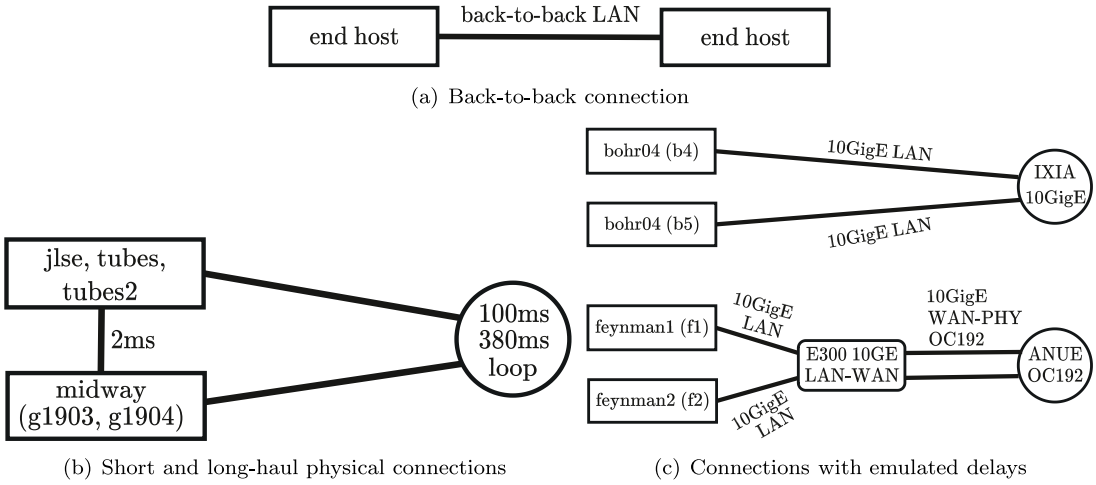
4.1. Testbeds

Our throughput performance measurements are accumulated and archived in the past several years based on numerous data transfer tests conducted on several HPN testbeds, as depicted in Fig. 1 and briefly described below.

- **UM-Local**: This testbed is established by back-to-back (Fig. 1(a)) connecting two regular workstations (dual-core, 2.9 GB RAM) via 10 GigE NICs. Both ends (um.dragon and um.rabbit) are installed with FC 17 Linux OS of 3.9.10 kernel.
- **NJIT-Local**: This testbed is established between two back-to-back connected (Fig. 1(a)) high-end servers (njit.tiger and njit.rabbit), each with 12 cores, 16 GB RAM, CentOS 3.10 kernel, and a 10 GigE NIC.
- **ANL-UC**: This testbed 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) including 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 the physical distance, while the 100 ms and 380 ms connections are engineered using a layer-2 circuit within ESnet that starts from ANL, extends to the west coast of the US, and loops back to UC (Fig. 1(b)).
- **UC-Local**: This testbed is a 40 Gbps local connection established between those two dynamically allocated virtual end hosts (g1903 and g1904) located at UC.
- **ORNL-E**: This emulated testbed at Oak Ridge National Laboratory (ORNL) as depicted in Fig. 1(c) 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, such as those provisioned via OSCARS (OSCARS, 2021) between Department of Energy sites, and the higher RTTs represent transcontinental connections.

4.2. 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 with a number of variants; and UDT (Gu and Grossman, 2007)



(a) Back-to-back connection

(b) Short and long-haul physical connections

(c) Connections with emulated delays

Fig. 1. Network structures of the testbeds used for data collection.

is a high-performance UDP-based data transfer protocol designated for LFNs and is widely adopted in HPN community (UDT, 2021b). Part of the tests are conducted specifically for data collection purposes and powered by a number of profiling-oriented toolkits including iperf2 (Iperf2, 2021), ESnet iperf3 (Iperf3, 2021), and TPG (Yun et al., 2015), while the rest are mainly the byproducts of the experimental studies for transport profiling optimization (Yun et al., 2016, 2015) and data transfer advising (Yun et al., 2019). All of these tests enable the tuning of various control parameters of TCP and UDT.

4.3. Data acquisition and introduction

We use the toolkits in Section 4.2 to run big data transfer tests on the testbeds in Section 4.1. 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 in three categories: (i) end host configurations; (ii) network connection properties; and (iii) data transfer protocols/toolkits and their control parameters, as listed in Table 2. Note that the latent attributes such as competing workloads on end hosts are omitted from Table 2 because they are typically unobservable with the limited system access granted to data transfer applications. We identify the effects of such latent attributes based on comparative experimental studies in Section 6.1.

5. Exploratory analysis

We perform exploratory analysis to investigate the impact of a comprehensive list of attributes on throughput performance of both TCP- and UDT-based big data transfers over connections with guaranteed bandwidth. Such analysis inspires feature selection in our machine learning-based performance prediction, helps identify the negative effects of latent variables, and further motivates the proposed threshold- and clustering-based latent effect eliminations for higher prediction accuracy. In this section, we present some representative results to illustrate the performance patterns in response to different attributes (more details can be found in Yun et al., 2020; Liu et al., 2020).

Table 2
List of attributes in the data transfer performance dataset.

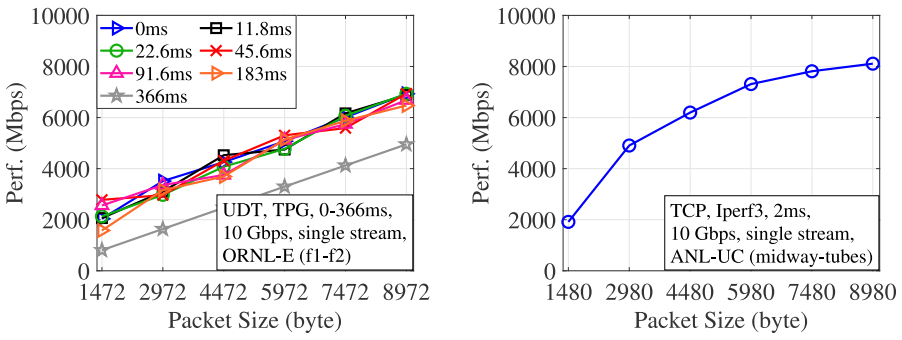
Categories	Attributes	Remarks
Identifier	Record ID	Integer, unique
Testbed	Testbed	String, nominal
End host	CPU frequency	Double, hertz
	# of processors	Integer
	# of cores per processor	Integer
	Memory size	Integer, MB
	Kernel buffer size	Integer, byte
Connection	Bandwidth	Double, Gbps
	RTT	Double, millisecond
	Loss rate	Double, emulated
Toolkits and 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 receive buffer size	Double, MB
	UDP send buffer size	Double, MB
	UDP receive buffer size	Double, MB
	UDT send buffer size	Double, MB
	UDT receive buffer size	Double, MB
	Number of streams	Integer
	Data size	Double, MB
	Time duration	Integer, second
Performance	Throughput	Double, Mbps

5.1. Effects of application-accessible parameters

We focus on a set C of parameters that are accessible and tunable in data transfer applications (e.g., iperf3 Iperf3, 2021 and TPG Yun et al., 2015) running in the user space, including packet size, block size, buffer size, and number of streams. To illustrate the independent impact of a parameter being examined, we empirically set other parameters in C with values that do not cause significant interferences.

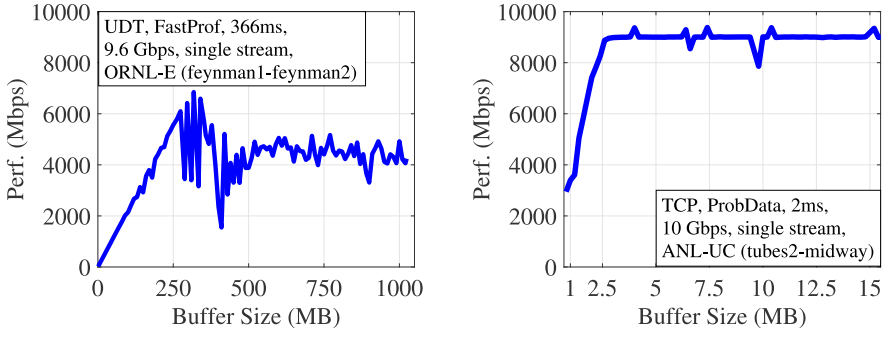
5.1.1. Packet size

Fig. 2 shows that a larger packet size typically increases the performance since it carries more per-packet user payload and reduces per-packet processing overhead (Chase et al., 2001). The increase pattern is almost linear when other parameters such as buffer size are fixed, which is consistent over connections with different RTTs on different testbeds between different hosts using both UDT (Fig. 2(a)) and TCP (Fig. 2(b)) protocols. If the buffer size is fixed at a value that



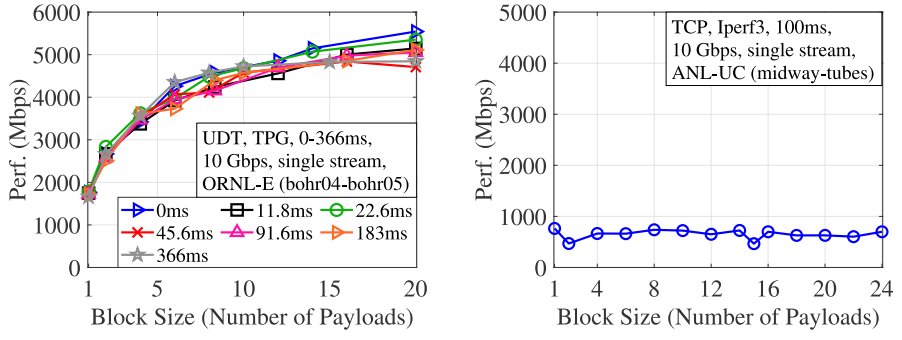
(a) ORNL-E, 0–366 ms, 10 Gbps, UDT (b) ANL-UC, 2 ms, 10 Gbps, TCP

Fig. 2. Performance vs. packet size. Results are collected on different testbeds with single-stream data transfer tests.



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

Fig. 3. Performance vs. buffer size. Results are collected on different testbeds with single-stream data transfer tests.



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

Fig. 4. Performance vs. block size. Results are collected on different testbeds with single-stream data transfer tests.

limits the performance, e.g., 256 MB, for a 10 Gbps connection with a 200+ ms RTT, the performance still linearly increases with packet size, but at a slower speed for a given packet size in comparison with other cases where the buffer size is sufficiently large, as represented by the 366 ms curve in Fig. 2(a).

5.1.2. Buffer size

The UDT and TCP performance in response to buffer size in Fig. 3 shows a "piecewise" pattern.

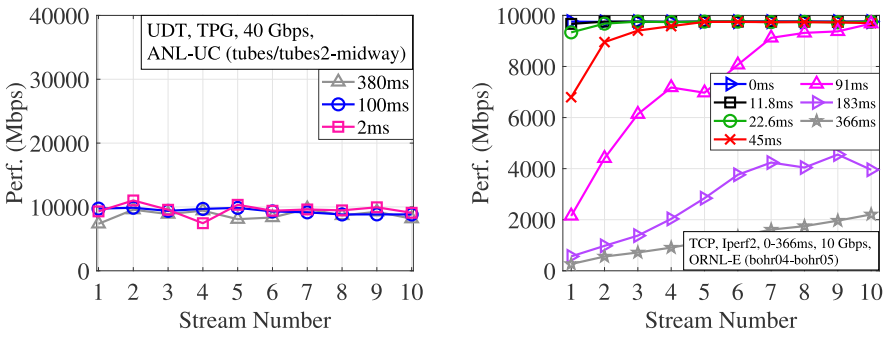
In the region where buffer size is insufficiently small, e.g., less than the bandwidth-delay product (BDP), TCP and UDT behave similarly and the performance linearly increases with buffer size. The slope of such increase varies across different hosts and connections, which can be interpreted by comparing Figs. 3(a) and 3(b). The maximal achievable performance in this region is mainly limited by buffer size and thus is lower than the overall peak.

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

In the region where buffer size is larger than the BDP, TCP and UDT significantly diverge and the performance may not stay at the peak after buffer size increases beyond the BDP. Fig. 3(b) shows that TCP performance stabilizes when buffer size exceeds the BDP. UDT performance when buffer size is around or larger than BDP is more complicated as an overly large buffer may hurt the performance, as shown in Fig. 3(a), which, compared with the TCP case, is counter-intuitive.

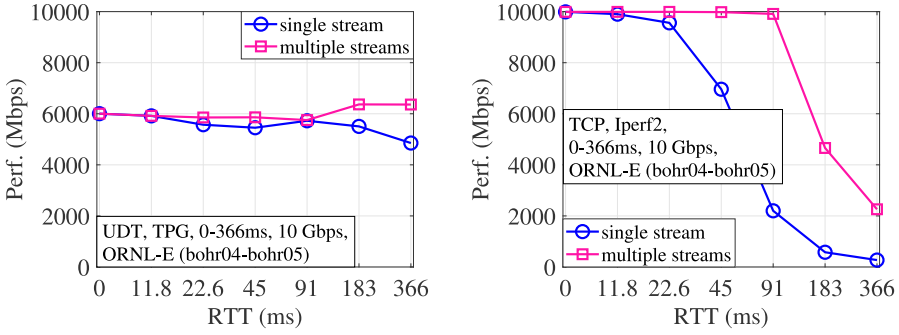
5.1.3. Block size

Fig. 4(a) shows the UDT performance increases with block size given a sufficiently large buffer. The performance curves first show a certain "concave" shape, indicating that the improvement brought



(a) ANL-UC, 2–380 ms, 40 Gbps, UDT (b) ORNL-E, 0–366 ms, 10 Gbps, TCP

Fig. 5. Performance vs. number of streams. Results are collected on different testbeds with fixed-buffer data transfer tests.



(a) UDT (b) TCP

Fig. 6. Maximal achievable performance of UDT and TCP vs. RTT. Results are collected from ORNL-E between hosts bohr04 and bohr05 using: (a) TPG UDT tests; and (b) iperf2 TCP tests.

by enlarging data block becomes marginal, and then stabilize at the peak after block size reaches a certain point. The optimal block size is also prone to other factors, but the performance pattern with respect to block size appears to be consistent across network connections with different properties, e.g., RTT. Fig. 4(b) shows the TCP performance is not significantly affected by block size, and the stabilized performance is mainly determined by other factors such as buffer size (Section 5.1.2), RTT (Section 5.2), and number of streams (Section 5.1.4).

5.1.4. Number of parallel streams

The UDT performance is expected to be insensitive to the number of streams used in data transfer applications since it is not designed for environments with high concurrency (Gu et al., 2004b). Fig. 5(a) shows that when the end hosts are able to keep up with extra overhead incurred by multi-stream UDT due to user-space buffer copying and context switching, the change to the number of streams does not affect the throughput significantly.

The effectiveness of using multi-stream TCP to achieve high throughput over LFN connections is well recognized in the network research community (Allcock et al., 2005). As shown in Fig. 5(b), single-stream TCP achieves near-capacity throughput over connections of short RTTs, but suffers over long-haul connections, where using multiple streams helps achieve higher performance. The performance increase pattern as the stream number increases is consistent over connections with various RTTs given that the number of streams is not excessively large to overwhelm the end hosts.

5.2. Effects of network connection properties

There are two important properties for a HPN connection: (i) the reserved bandwidth, which sets a theoretical upper bound for the achievable performance; and (ii) the connection delay, which affects the performance to a large extent but in a different way from shared

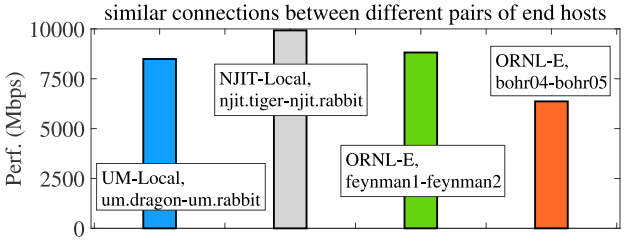


Fig. 7. Maximal achievable performance of UDT over four connections (RTT ≈0 ms) between different pairs of end hosts.

Internet connections. All else in ORNL-E being equal, Fig. 6 plots the maximal achievable performance of UDT and TCP in response to various emulated RTTs, and shows that the performance varies and generally decreases as RTT increases. Comparing Figs. 6(a) and 6(b), we observe that: (i) UDT is more stable than TCP across different RTTs; (ii) TCP outperforms UDT for short RTTs but fails to keep up with UDT for mid-range and long RTTs; (iii) using multiple streams helps TCP outperform UDT for mid-range RTTs; and (iv) UDT outperforms TCP in both single- and multi-stream cases for longer RTTs.

5.3. Effects of end hosts configurations

The complexities in host hardware/software configurations and time-varying system loads and dynamics make it non-trivial to predict big data transfer performance. They, together with connection properties, impose an upper bound on the achievable performance using different transport methods. Fig. 7 compares the maximal achievable performance of UDT over similar 10 Gbps connections established between four different pairs of hosts, respectively. Fig. 8 shows the

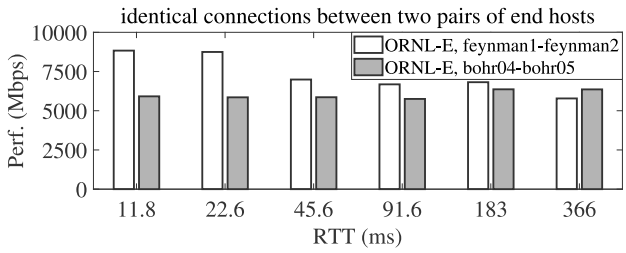


Fig. 8. Maximal achievable performance of UDT over connections with different RTTs emulated between the same two pairs of end hosts.

performance difference over identical 10 Gbps connections with different delays emulated between the same two pairs of end hosts. They both show that similar or identical connections between different hosts may result in very different maximal performance achievable by "near-exhaustive" tuning.

5.4. Remarks

The results presented in Sections 5.1, 5.2, and 5.3 suggest the use of machine learning for performance prediction of big data transfer in HPN environments. There exist clear patterns between throughput performance y and the feature vector $\mathbf{x}$, and such patterns are qualitatively consistent and stable across different connections established between different end hosts on different testbeds, e.g., the performance increases as the buffer size and the number of streams increase; the maximum achievable performance decreases as the connection delay increases. However, the hyper-dimensional parameter space makes it very difficult, if not impossible, to analytically model these patterns for performance prediction. For example, the slope of performance increase with respect to buffer size increase may vary across different connections; the optimal number of data streams may depend on not only the connection properties but also the end system configurations.

6. Performance prediction

We first study the effects of latent factors on big data transfer performance (Section 6.1), and then propose latent effect elimination methods and conduct experiments to illustrate their effectiveness (Section 6.2). We also develop a domain-oriented loss function for performance prediction based on practical requirements of HPN management (Section 6.3). We develop a performance predictor that incorporates latent effect elimination and customized loss function.

6.1. Effects of latent factors

There exist certain latent factors $\mathbf{u}$ that also have effects on transport performance. Such latent effects, if not eliminated, may cause dissatisfactory prediction results. To illustrate such latent effects, we compare the TCP measurements of the ***same*** set of data transfer tests conducted on two testbeds: (i) a production HPN (ANL-UC) where the hosts are simultaneously shared by many users and latent effects are significant; (ii) a local testbed (NJIT-Local) where the hosts are strictly controlled and latent effects are mild. Fig. 9(b) shows that the performance pattern is obvious under mild latent effects. In Fig. 9(a), under significant latent effects, although the maximal achievable performance also follows an obvious pattern, there are also a non-negligible number of data points below the maximal ones for the same (or similar) buffer sizes. Similar phenomenons are also observed in UDT tests shown in Fig. 10.

6.2. Elimination of latent effects

6.2.1. Rationale

As mentioned in Section 3, the throughput y of a data transfer is determined by accessible variables in feature vector $\mathbf{x}$, non-accessible variables in latent factors $\mathbf{u}$, and noise ζ, in some form of function $y = f(\mathbf{x})+\zeta+\xi(\mathbf{u})$. The "normal" data points $\{y'\}$ are the measurements from environments with $\xi(\mathbf{u}) \approx 0$ and the performance is mainly determined by the feature vector $\mathbf{x}$, i.e., $y' = f(\mathbf{x}) + \zeta$. The "corrupted" data points $\{y''\}$ are the measurements from environments with $\xi(\mathbf{u}) < 0$ and the performance is determined by both $\mathbf{x}$ and latent factors in $\mathbf{u}$, i.e., $y'' = f(\mathbf{x}) + \zeta + \xi(\mathbf{u})$. In the presence of significant latent effects, e.g., $\xi(\mathbf{u}) \ll 0$, the data points in the training set $\mathcal{T}$ are essentially sampled from a combined set of both $\{y'\}$ and $\{y''\}$ that are governed by different functions, as collectively shown in Figs. 9 and 10.

For bandwidth scheduling, we wish to predict the *maximum achievable* performance of a data transfer, and thus reserve a suitable amount of bandwidth to meet the actual need while minimizing the resource waste caused by over-provisioning. Ideally, the prediction model should be trained completely with $\{y'\}$, since $\{y''\}$ under significant $\xi(\mathbf{u})$ may result in high noise and large variances in $\mathcal{T}$, and thus impair prediction quality. However, it is difficult to build an accurate performance predictor without over-fitting if $\{y'\}$ and $\{y''\}$ coexist in $\mathcal{T}$. In addition, due to the access limit in the user space, most latent factors in $\mathbf{u}$ are not directly observable. Due to the unpredictability and randomness of $\mathbf{u}$, it is practically infeasible to model or estimate $\xi(\mathbf{u})$. Therefore, we propose to identify and eliminate $\{y''\}$ from $\mathcal{T}$ and then train our prediction model based on $\mathcal{T}$.

6.2.2. Threshold-based latent effect elimination

When $\xi(\mathbf{u}) \approx 0$, the performance measurements in response to a given $\mathbf{x}_i$ are data points sampled from a certain Gaussian distribution with mean $\overline{f}(\mathbf{x}_i)$ and variance $\sigma(\mathbf{x}_i)$, and $\sigma(\mathbf{x}_i)$ is bounded by some scale determined by ζ_i. In other words, significant performance differences

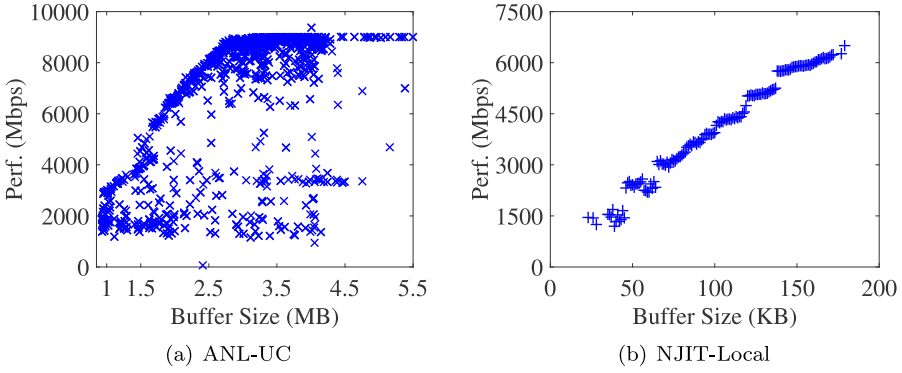
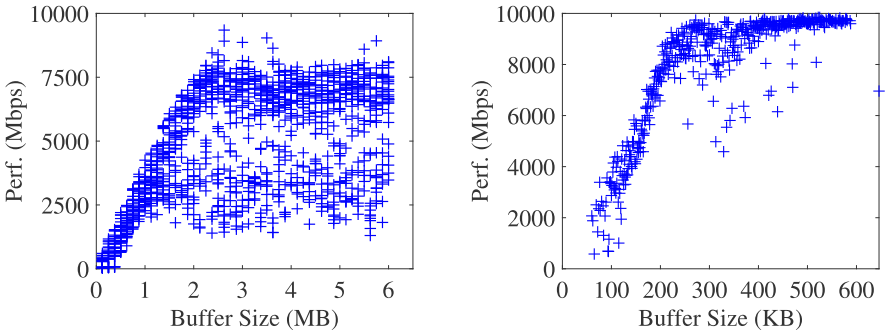


Fig. 9. TCP performance vs. buffer size: (a) with latent effects; (b) without latent effects.



(a) ANL-UC (b) NJIT-Local

Fig. 10. UDT performance vs. buffer size: (a) with latent effects; (b) without latent effects.

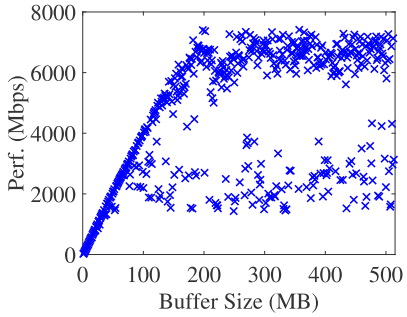


Fig. 11. UDT performance in response to buffer size diverges.

observed in ***repeated*** measurements with the ***same*** set of values of $\mathbf{x}_i$ indicate the existence of $\{y''\}$ that may diverge the observed performance pattern. Fig. 11 presents such a case where the UDT performance in response to buffer size diverges into two different patterns. Since our goal is to avoid any excessive bandwidth reservation beyond actual needs and meanwhile ensure that the reserved (i.e., predicted) bandwidth is around the maximal achievable for a given $\mathbf{x}_i$, we propose a simple threshold-based method to eliminate the latent effects in performance prediction by excluding the "undermined" data points whose performances are below a threshold τ of the corresponding achievable maximum. In particular, if there are multiple measurements with the ***same*** set of values for a given $\mathbf{x}_i$, those with an observed performance y_i below $\tau \cdot \max_i \{y_i\}$ $(0 < \tau < 1)$ are discarded in data preprocessing.

6.2.3. Clustering-based latent effect elimination

The function $f(\mathbf{x})$ is bounded by the connection capacity and satisfies certain smoothness conditions. For example, we have $|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L \cdot |\mathbf{x}_1 - \mathbf{x}_2|$ for some constant $L > 0$ and arbitrary but feasible $\mathbf{x}_1$ and $\mathbf{x}_2$. When $\xi(\mathbf{u}) \approx 0$, with a small variation to $\mathbf{x}$, the corresponding change in the observed performance should be bounded as well. In other words, large differences observed in ***repeated*** measurements with only ***small*** changes to $\mathbf{x}$ also indicate the existence of $\{y''\}$ caused by $\xi(\mathbf{u})$. In such a case, if a sufficient number of data points for different values of $\mathbf{x}$ are measured, performance divergence may occur as exemplified in Fig. 11.

If there are no repeated measurements with the same values of $\mathbf{x}_i$, the proposed threshold-based method is not applicable. In this case, we propose to use a clustering-based method to divide $\{y'\}$ and $\{y''\}$ in $\mathcal{T}$ into different groups such that the measurements in the same group are more likely to be observed under similar conditions with similar $\xi(\mathbf{u})$, and then eliminate those that are mainly manifested by $\{y''\}$.

Particularly, we use the DBSCAN clustering algorithm (Ester et al., 1996) to differentiate $\{y'\}$ and $\{y''\}$ in $\mathcal{T}$. DBSCAN splits the data points in $\mathcal{T}$ into different clusters based on their densities. Tightly-packed points are grouped together and those in low-density regions are classified as performance measurements with severe latent effects. Fig. 12 shows the effectiveness of DBSCAN in identifying $\{y''\}$ from $\mathcal{T}$. Note that the results in Fig. 12(c) are based on the same dataset as in Fig. 11. In addition, with the same dataset as in Fig. 12(c), we test three other commonly-used clustering algorithms and present their results in Fig. 13. We observe that K-means, Gaussian Mixture Model (GMM), and Spectral Clustering are incapable of identifying $\{y''\}$ from $\mathcal{T}$ as they simply divide the data points in $\mathcal{T}$ into two groups with a roughly equal radius measured in certain distance metrics such as Euclidean distance. Similar results are also observed when such clustering is performed in the kernel feature space, e.g., K-means in the sigmoid kernel space as shown in Fig. 13(d).

6.3. Customized loss function

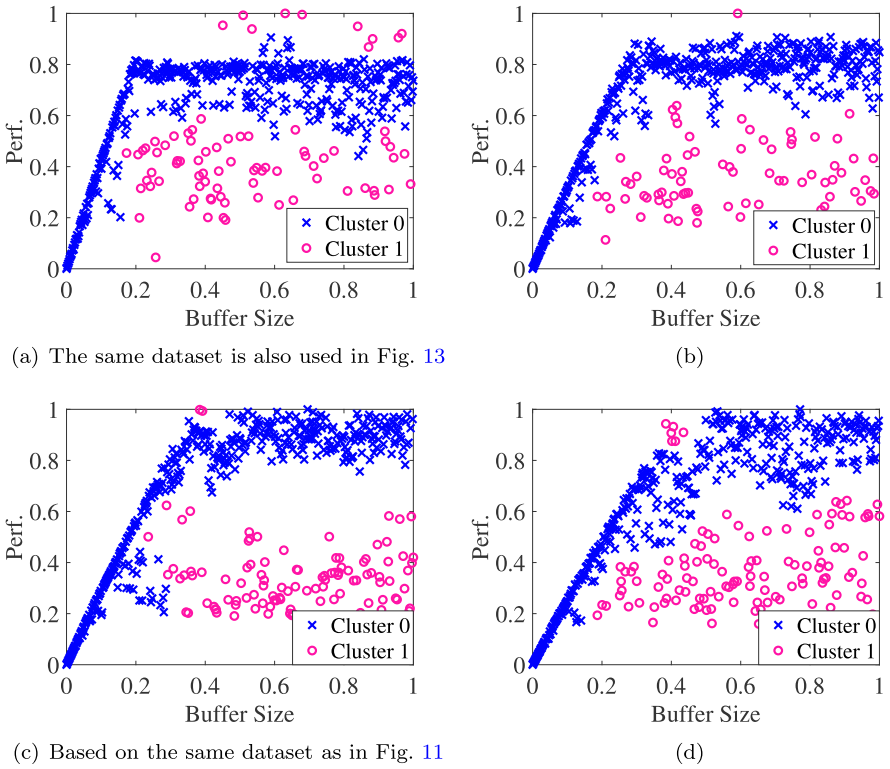
Different from traditional supervised learning methods (Mirza et al., 2010; Liu et al., 2018) that seek an optimal label for a given feature vector $\mathbf{x}_i$, for bandwidth scheduling, the goal is to build a model that provides a loosened prediction. We customize the loss function of a model based on practical requirements of bandwidth reservation in HPNs, where the reserved bandwidth must match the actual demand of a data transfer request with minimal over-provisioning. Therefore, the optimal predicted performance $\hat{y}_i$ for a given $\mathbf{x}_i$ should lie within the range of $[y_i, y_i+\epsilon]$, where $\epsilon \geq 0$ is a small tunable parameter and y_i is the ground truth of the achievable performance with respect to $\mathbf{x}_i$. In other words, the reserved (predicted) bandwidth $\hat{y}_i$ should be slightly higher than what a data transfer can utilize to satisfy the user request and meanwhile minimize resource waste. Inspired by the ϵ-insensitive loss used by SVR, we customize the ϵ-insensitive loss function (Fig. 14(a)) by restricting the tolerable errors to be only positive. As shown in Fig. 14(b), the customized loss function $\mathcal{L}(\theta, \epsilon)$ is parameterized by an error tolerance ϵ as,

$$\mathcal{L}(\theta,\epsilon) = \begin{cases} -(\hat{y}_i^\theta - y_i), & \text{if } \hat{y}_i^\theta - y_i < 0 \\ 0, & \text{if } 0 \leq \hat{y}_i^\theta - y_i \leq \epsilon \\ \hat{y}_i^\theta - y_i, & \text{if } \hat{y}_i^\theta - y_i > \epsilon \end{cases}, \tag{1}$$

where θ is the parameter set of the prediction model. If the predicted performance $\hat{y}_i^\theta$ is larger than the true value y_i but within the tolerable range bounded by ϵ, $\mathcal{L}(\theta, \epsilon) = 0$; otherwise, $\mathcal{L}(\theta, \epsilon)$ is the distance between $\hat{y}_i^\theta$ and the tolerable ϵ loss range. Our objective is to minimize $\mathcal{L}(\theta, \epsilon)$.

7. Performance evaluation

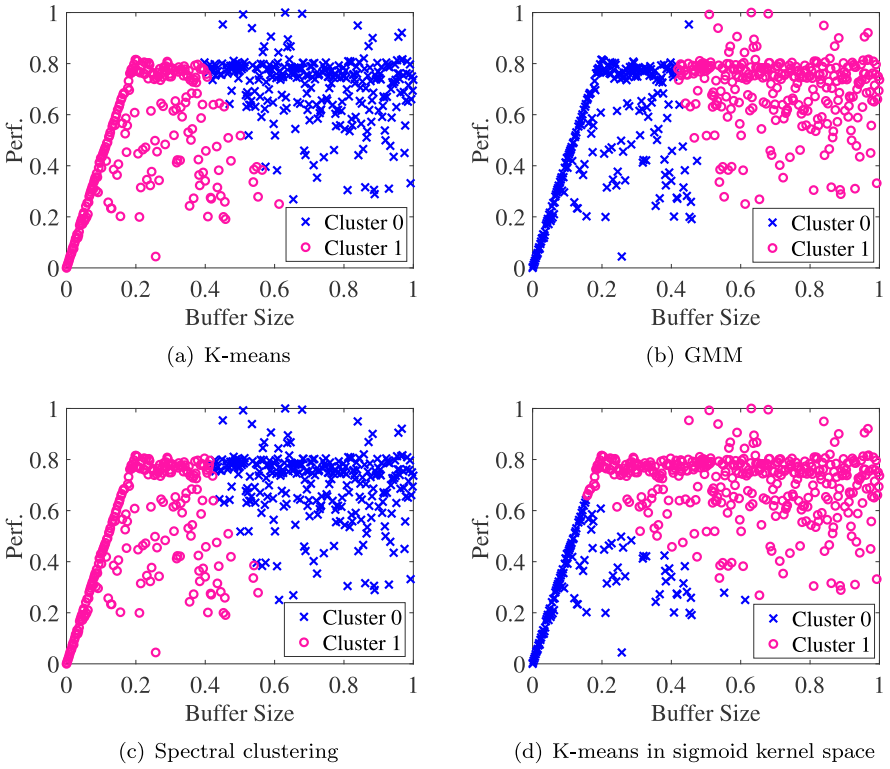
In this section, we evaluate the performance of the proposed methods. We first study the efficacy of the latent elimination methods (Section 7.1), and then compare the prediction results of Support Vector Regression (SVR) with and without using the domain-oriented



(a) The same dataset is also used in Fig. 13 (b)

(c) Based on the same dataset as in Fig. 11 (d)

Fig. 12. Clustering results of DBSCAN (values are normalized). Cluster 0: "normal" data points; Cluster 1: data points with latent effects.

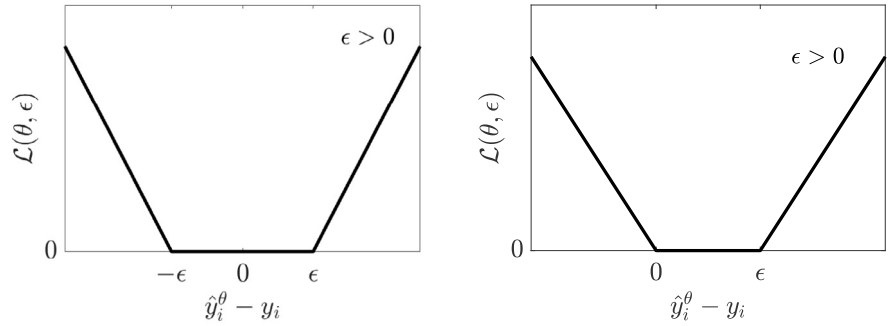


(a) K-means (b) GMM

(c) Spectral clustering (d) K-means in sigmoid kernel space

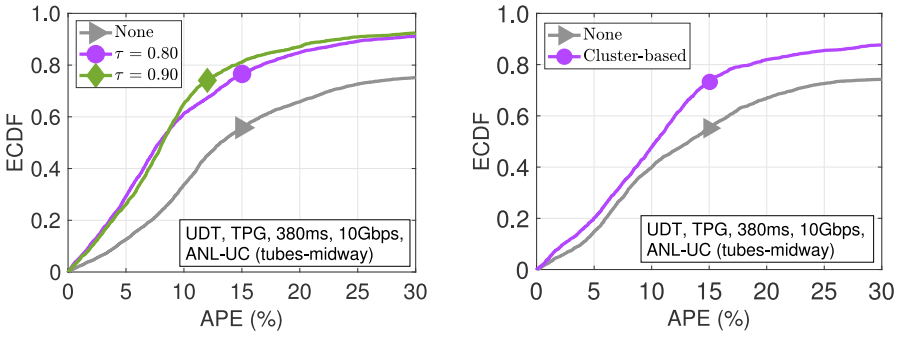
Fig. 13. Clustering results of different algorithms (values are normalized). Cluster 0: "normal" data points; Cluster 1: data points with latent effects.

customized the loss function in training (Section 7.2). We build a performance predictor by integrating latent elimination into data pre-processing and incorporating the customized loss function into model training of SVR, and compare its prediction performance with another representative method, Random Forest Regression (RFR) (Section 7.3). We provide theoretical analysis of the confidence of prediction by deriving general performance bounds (Section 7.4), which show that $y = f(\mathbf{x})$ is indeed a good estimate in a statistical sense with a high probability. Note that all datasets used for both training and testing are normalized.



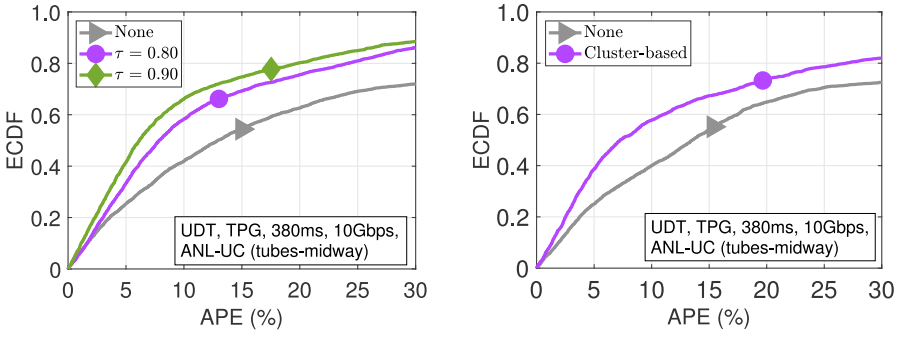
(a) ϵ-insensitive loss (b) one-side ϵ-insensitive loss

Fig. 14. Loss functions.



(a) Threshold-based (b) Clustering-based

Fig. 15. Performance prediction results using **default** SVR with and without latent effect elimination in data preprocessing.



(a) Threshold-based (b) Clustering-based

Fig. 16. Performance prediction results using **customized** SVR with and without latent effect elimination in data preprocessing.

7.1. Effectiveness of latent effect elimination

To illustrate the effectiveness of the proposed latent elimination methods, we implement a predictor using SVR algorithm based on the `scikit-learn` library (Pedregosa et al., 2011). We train this predictor using performance measurements with and without eliminating latent effects from the dataset using the methods proposed in Section 6.2. In particular, this SVR-based predictor uses the Radial Basis Function (RBF) kernel and the default and customized loss functions, as presented in Section 6.3 with an error tolerance $\epsilon = 0.05$. It performs grid search (with 5-fold cross validation) with a kernel coefficient set $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.5, 0.6, 0.9, 10\}$ and a regularization parameter set $\{0.01, 0.1, 1.0, 10\}$. In each case, the dataset is randomly split for training (80%) and testing (20%), respectively. The prediction accuracy of a test case is measured by the Absolute Percentage Error (APE) defined as $\frac{|\hat{y}_i - y_i|}{y_i} \times 100\%$, where y_i is the true value and $\hat{y}_i$ is the predicted value. The performance measurements used here are collected from the UDT data transfer tests performed over the 380 ms connection on ANL-UC testbed under non-negligible latent effects that are mainly caused by competing workloads on the end hosts (tubes and midway) of the connection. Note that these hosts are simultaneously used by other scientists for running their scientific computing jobs during our data transfer experiments.

Figs. 15 and 16 plot the empirical cumulative distribution function (ECDF) that are measured in terms of APE over all test cases with and without data preprocessing using the proposed latent elimination methods based on the SVR algorithm with (Fig. 15) and without (Fig. 16) loss function customization. The plots marked with "None" are obtained using the "raw" datasets without latent effect elimination in data preprocessing.

In Fig. 15(a), we conduct performance prediction using the ANL-UC 380 ms UDT dataset combined with necessary synthetic repeated measurements. Without data preprocessing, a 15% APE is achieved only around 55% of the time among all test cases. Incorporating the

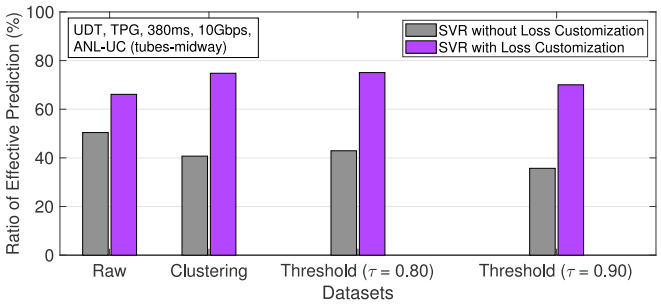


Fig. 17. Comparison of the ratio of effective prediction using SVR with and without loss function customization.

threshold-based method for latent effect elimination into data preprocessing, when $\tau = 0.80$, 26.7% of the test data points under latent effects are removed and the prediction accuracy is significantly improved and the 15% percentile of APE is increased to 77% (a 20%+ improvement); when $\tau = 0.90$, 37.6% of the test data points under latent effects are removed and the 15% percentile of APE is further increased to 81%. In Fig. 15(b), we conduct performance prediction using the "raw" ANL-UC 380 ms UDT dataset, which shows that the prediction performance is also significantly improved by incorporating the proposed clustering-based latent effect elimination into data preprocessing. For example, a 15% percentile of APE is increased from 55% to 73%, and in this case, 34.9% of the test data points under latent effects are removed.

In Fig. 16, we conduct similar experiments using the same datasets as in Fig. 15 but using the SVR algorithm with the customized loss function stated in Eq. (1). Similar improvements on prediction performance are observed as well. For example, as shown in Fig. 16(a), a 10% percentile of APE is increased from 41% to 59% and 67% with $\tau = 0.80$ and $\tau = 0.90$, respectively.

7.2. Effectiveness of customized loss function

To illustrate the effectiveness of the proposed customized loss function in Eq. (1), we test SVR with and without customizing its loss function based on the ANL-UC 380 ms UDT measurements as described in Section 7.1. We count the number of effective predictions (i.e., $\hat{y}_i \geq y_i$) and calculate its corresponding ratio among all test cases. As expected, Fig. 17 shows that, using the "raw" dataset without preprocessing, SVR with loss function customization produces 20% more effective performance predictions than without loss function customization. With data preprocessing using the proposed clustering- and threshold-based (with $\tau \in \{0.80, 0.90\}$) latent elimination methods, such improvement in terms of effective prediction ratio is up to 40%. The results in Fig. 17 show that using SVR with customized loss function has a much higher chance to produce a performance prediction that guarantees sufficient resource reservation to obtain the maximal achievable performance of a data transfer request.

Although small over-provisioning is inevitable and necessary to obtain the maximum achievable performance, significant resource waste could be caused by predictions when $\hat{y}_i \gg y_i$ and should be avoided in practice. Therefore, we further measure the resource waste that might be potentially caused by over-provisioning when $\hat{y}_i > y_i$ in terms of absolute percentage error and present the ECDF plots in Fig. 18. The results in Fig. 18 show that whether using the raw dataset (Fig. 18(a)) or applying latent effect elimination using the proposed methods (Figs. 18(b), 18(c), and 18(d)), SVR with customized loss always performs better than without customized loss.

Note that the results in Figs. 17 and 18 are obtained with the same sets of tuning parameter values for SVR as in Section 7.1.

7.3. Comparison between SVR and RFR

We compare the prediction performance of SVR with Random Forest Regression (RFR). The RFR-based predictor is also implemented based on the `scikit-learn` library (Pedregosa et al., 2011).

7.3.1. Settings

For the ANL-UC 380 ms UDT dataset, we apply the proposed clustering-based latent effect elimination in data preprocessing and train the SVR model with its default loss function replaced by the proposed "one-side" ϵ-insensitive loss function (Eq. (1)). Other parameter settings for this SVR training remain the same as in Section 7.1.

In comparison, we train two RFR models using the same dataset with and without applying the clustering-based latent elimination in data processing. We perform grid search (with 5-fold cross validation) to find the best hyperparameters for the RFR model with the set of numbers of trees $\{100, 200, 300, 400, 500\}$, the set of maximum depths of trees $\{5, 10, 15, 20, 25, 30\}$, the set of minimum numbers of data samples to split internal nodes of trees $\{2, 5, 10, 15, 100\}$, and the set of minimum numbers of samples in leaf nodes of trees $\{1, 2, 5, 10\}$.

7.3.2. Results

As shown in Fig. 19(a), using the raw dataset without latent effect elimination, RFR performs poorly and the 10% percentile of APE is achieved only around 25% of the time among all test cases, as denoted by "RFR+None". Using the raw dataset with the proposed clustering-based latent elimination in data preprocessing, SVR and RFR perform roughly equally well and the 10% percentile of APE is achieved 70% of the time among all test cases, as denoted by "C-SVR+Clustering" and "RFR+Clustering", respectively. Taking a deeper look, as shown in Fig. 19(b), SVR outperforms RFR in terms of effective prediction ratio. It indicates that using SVR with customized loss is more likely to make an effective prediction that meets the requirement of a data transfer request, while incurring a comparable level of resource waste to other methods such as RFR.

7.4. Theoretical confidence analysis

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

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

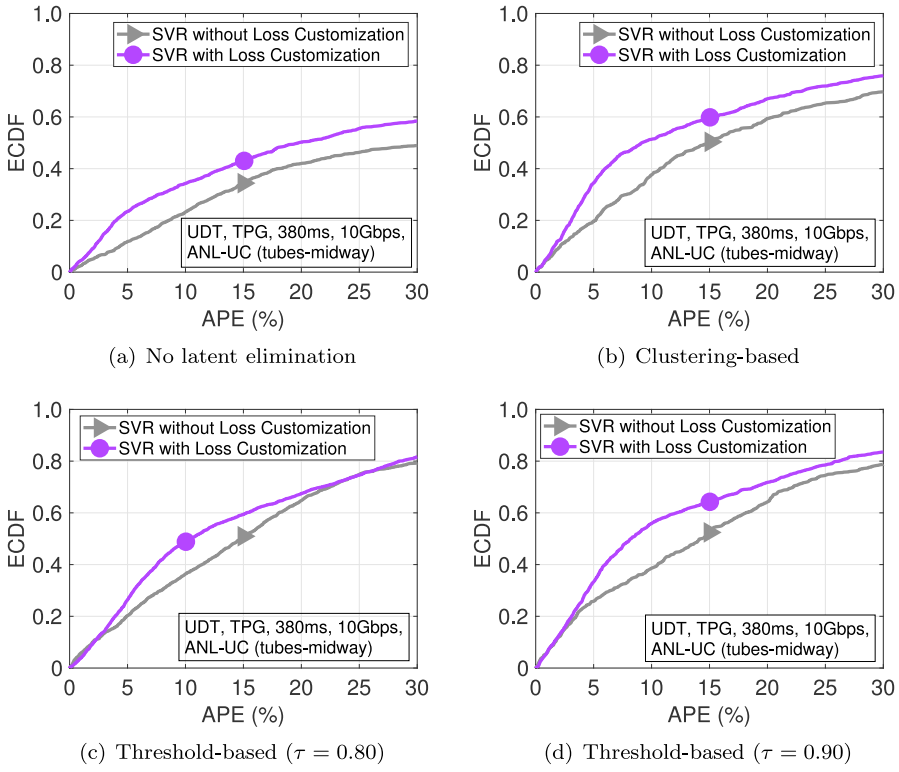
which can be estimated based on experimental performance measurements $y(\mathbf{x}_k, t_i^k)$ at $\mathbf{x}_k$ $(k = 1, 2, \ldots, n)$ and time t_i^k $(i = 1, 2, \ldots, n_k)$. We have $0 \leq y(\mathbf{x}_k, t_i^k) \leq B$ due to the reserved bandwidth B of an HPN connection. The performance estimate $\hat{y}(\mathbf{x}_k)$, 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 $\mathbf{x} = [\mathcal{H}, \mathcal{P}, C]$. 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 is improved 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

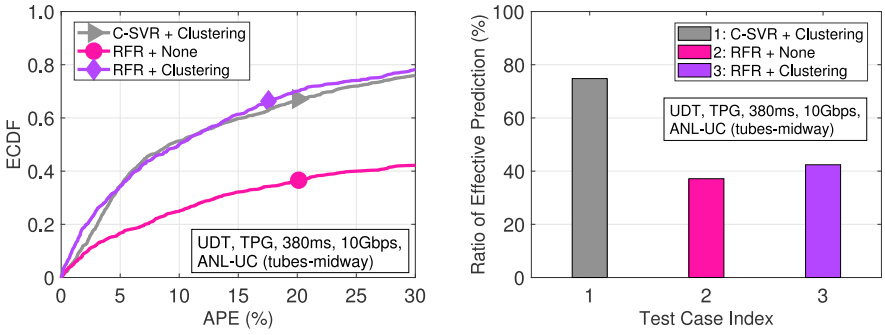
$$I(g) = \int \left[g(\mathbf{x}) - y(\mathbf{x}, t)\right]^2 d\, \mathbf{P}_{y(\mathbf{x},t)},$$



(a) No latent elimination (b) Clustering-based

(c) Threshold-based ($\tau = 0.80$) (d) Threshold-based ($\tau = 0.90$)

Fig. 18. Effective performance prediction results using SVR with and without loss function customization.



(a) ECDF (b) Effective prediction ratio

Fig. 19. Comparison of the prediction results using SVR and RFR.

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

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

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 $\mathcal{I} = \{I(g) \mid g \in \mathcal{F}\}$ be the set of loss functions subject to $\mathcal{F}$. Based on the uniform convergence results of Vapnik–Chervonenkis theory (Vapnik, 1995) and its generalization (Anthony and Bartlett, 2009) and applications (e.g., Rao, 1999), we know that, for some $\lambda > 0$,

$$P\{I(\hat{y}) - I(g^*) > 2\lambda\} \le P\left\{\sup_{h\in\mathcal{F}}\left|I(h) - \hat{I}(h)\right| > \lambda\right\},$$

and furthermore, according to Vapnik (1982), we have

$$P\left\{\sup_{h\in\mathcal{F}}\left|I(h) - \hat{I}(h)\right| > \lambda\right\} \le 18\mathcal{N}_1\left(\frac{\lambda}{K}, \mathcal{I}, n\right)\cdot n\cdot\exp\left(-\frac{n\lambda^2}{4K^2}\right),$$

where $\mathcal{N}_1(\frac{\lambda}{K}, \mathcal{I}, n)$ is the λ-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

$$\mathcal{N}_1\left(\frac{\lambda}{K}, \mathcal{I}, n\right) \le \mathcal{N}_1\left(\frac{\lambda}{KL}, \mathcal{F}, n\right).$$

Also, for any class $\mathcal{M}$ of real-valued functions, any $\delta > 0$ and any $j \in \mathbb{N}$, we have $\mathcal{N}_1(\delta, \mathcal{M}, j) \le \mathcal{N}_\infty(\delta, \mathcal{M}, j)$ (Anthony and Bartlett, 2009). It follows that

$$P\{I(\hat{y}) - I(g^*) > 2\lambda\} \le 18\mathcal{N}_\infty\left(\frac{\lambda}{KL}, \mathcal{F}, n\right)\cdot n\cdot\exp\left(-\frac{n\lambda^2}{4K^2}\right),$$

where $\mathcal{N}_\infty(\lambda, \mathcal{F})$ is the λ-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 (Anthony and Bartlett, 2009),

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

By using this bound, we obtain

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

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 λ 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, which is a complex composition of the impacts of end host configurations and dynamics, network properties and randomness, and data transport methods as well as control parameters.

Note that in the derivation, since we only consider the fact that the loss function $I(g) \in \mathcal{I}$ varies slowly as the function value $g(\cdot)$ varies, the specific loss function does not affect the bound analysis but only its constant coefficients. For example, if considering quadratic loss functions, then the Lipschitz constant is $L = 2B$ and $K = B^2$. Also note that the actual coefficients in the derived bound depend on detailed algebra and are not critical in terms of the uniform convergence. The same is also true for the specific bounds of $g(\cdot)$ and $y(\cdot)$ since one can easily derive covering number bounds for different classes of functions that map to any bounded intervals with appropriate scaling and shifting (Anthony and Bartlett, 2009).

8. Conclusion and future work

We conducted exploratory analysis of the impacts of a comprehensive set of factors on the application-level performance of big data transfer in HPNs based on extensive performance measurements collected on real-life physical or emulated HPN testbeds. Based on such analysis, we further identified latent factors and analyzed their negative impacts on performance prediction through comparative experimental studies. We proposed novel methods to eliminate the negative impacts of latent factors, and incorporated them into data preprocessing to improve training efficiency and prediction accuracy. We then selected features and built a performance predictor using machine learning methods with customized domain-oriented loss functions. The experimental results show that, based on very noisy datasets, the proposed latent effect elimination methods and the customized loss function help achieve significantly better prediction performance in comparison with other methods. We also investigated the feasibility and effectiveness of learning-based performance prediction through theoretical performance bound analysis.

We plan to synthetically study the effects of latent variables on the performance of big data transfer and further improve prediction accuracy. We will also explore the feasibility and efficacy of other techniques such as deep learning and data fusion for performance prediction in HPNs. It is also of our interest to derive tighter performance bounds on the estimated loss and the sample size by incorporating other HPN domain insights.

CRediT authorship contribution statement

Daqing Yun: Conceptualization, Methodology, Writing - original draft, Formal analysis, Investigation. **Wuji Liu:** Methodology, Writing - original draft, Visualization, Validation, Investigation, Data curation. **Chase Q. Wu:** Conceptualization, Methodology, Funding acquisition, Supervision, Project administration, Writing - review & editing. **Nageswara S.V. Rao:** Resources, Data curation, Writing - review & editing. **Rajkumar Kettimuthu:** Resources, Data curation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is sponsored by Harrisburg University, USA under Grant No. PRG-2020-15 and by the U.S. National Science Foundation under Grant No. CNS-1828123 with New Jersey Institute of Technology.

References

Allcock, W., et al., 2005. The Globus striped GridFTP framework and server. In: Proc. ACM/IEEE Conf. Supercomput. pp. 54–65.
Anthony, M., Bartlett, P., 2009. Neural Network Learning: Theoretical Foundations. Cambridge University Press, New York, NY.
Chase, J., et al., 2001. End system optimizations for high-speed TCP. IEEE Commun. Mag. 39 (4), 68–74.
ESnet, 2021. http://www.es.net.
Ester, M., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. Int. Conf. Knowl. Discovery Data Mine. pp. 226–231.
Gangulay, S., et al., 2004. Optimal routing for fast transfer of bulk data files in time-varying networks. In: Proc. IEEE Int. Conf. Commun., Vol. 2. pp. 1182–1186.
Grimmell, W., Rao, N., 2003. On source-based route computation for quickest paths under dynamic bandwidth constraints. Int. J. Found. Comput. Sci. 14 (3), 503–523.
Gu, Y., Grossman, R., 2007. UDT: UDP-based data transfer for high-speed wide area networks. Comput. Netw. 51 (7), 1777–1799.
Gu, Y., et al., 2004. An analysis of AIMD algorithm with decreasing increases. In: Proc. Int. Workshop Netw. Grid Appl.
Gu, Y., et al., 2004. Experiences in design and implementation of a high performance transport protocol. In: Proc. ACM/IEEE Conf. Supercomput. pp. 22–35.
Guok, C., et al., 2006. Intra and interdomain circuit provisioning using the OSCARS reservation system. In: Proc. 3rd Int. Conf. on Broadband Commun., Netw. Syst.
Hanford, N., et al., 2016. Improving network performance on multicore systems: Impact of core affinities on high throughput flows. Future Gener. Comput. Syst. 56, 277–283.
He, Q., et al., 2007. On the predictability of large transfer TCP throughput. Comput. Netw. 51 (14), 3959–3977.
Internet2, 2021. http://www.internet2.edu.
Iperf2, 2021. https://bit.ly/2WmMPhN.
Iperf3, 2021. https://github.com/esnet/iperf.
Jain, S., et al., 2013. B4: Experience with a globally-deployed software defined WAN. SIGCOMM Comput. Commun. Rev. 43 (4), 3–14.
Leitao, B., 2009. Tuning 10 Gb network cards on Linux. In: Proc. Linux Symp. pp. 169–184.
Lin, Y., Wu, Q., 2013. Complexity analysis and algorithm design for advance bandwidth scheduling in dedicated networks. IEEE Trans. Netw. 21 (1), 14–27.
Liu, Q., et al., 2016. Measurement-based performance profiles and dynamics of UDT over dedicated connections. In: Proc. Int. Conf. Netw. Protocols.
Liu, Z., et al., 2017. Explaining wide area data transfer performance. In: Proc. Int. Symp. High-Perform. Parallel Distrib. Comput. pp. 167–178.
Liu, Z., et al., 2018. Building a wide-area data transfer performance predictor: An empirical study. In: Proc. Int. Conf. Machine Learn. for Netw.
Liu, W., et al., 2020. On performance prediction of big data transfer in high-performance networks. In: Proc. IEEE Int. Conf. Commun.
Mirza, M., et al., 2010. A machine learning approach to TCP throughput prediction. IEEE Trans. Netw. 18 (4), 1026–1039.
OSCARS, 2021. https://bit.ly/2Ou9qVe.
Padhye, J., et al., 2000. Modeling TCP reno performance: A simple model and its empirical validation. IEEE Trans. Netw. 8 (2), 133–145.
Pedregosa, F., et al., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Rao, N., 1999. Simple sample bound for feedforward sigmoid networks with bounded weights. Neurocomputing 29 (1), 115–122.
Rao, N., et al., 2006. Control plane for advance bandwidth scheduling in ultra high-speed networks. In: Proc. 25th Int. Conf. Comput. Commun.
Rao, N., et al., 2017. TCP throughput profiles using measurements over dedicated connections. In: Proc. Int. Symp. High-Perform. Parallel Distrib. Comput. pp. 193–204.

Sapkota, H., et al., 2019. Time series analysis for efficient sample transfers. In: Proc. Workshop Syst. Netw. Telemetry Analytics. pp. 11–18.

Shu, T., et al., 2013. Advance bandwidth reservation for energy efficiency in high-performance networks. In: Proc. 38th IEEE Conf. Local Comput. Netw. pp. 541–548.

Spall, J., 2003. Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. John Wiley & Sons, Inc., Hoboken, NJ.

Tierney, B., 2016. Advantages of TCP pacing using FQ. https://bit.ly/30g4QO0.

UDT, 2021a. UDT: UDP-based data transfer. http://udt.sourceforge.net/.

UDT, 2021b. UDT-powered projects. https://bit.ly/2JZtA7n.

UDT, 2021c. UDT socket options. https://bit.ly/2VOoBsJ.

Vapnik, V., 1982. Estimation of Dependences Based on Empirical Data. Springer-Verlag, New York, NY.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag, Berlin, Heidelberg.

XSEDE, 2021. https://www.xsede.org/.

Yu, S., et al., 2015. Comparative analysis of big data transfer protocols in an international high-speed network. In: Proc. 34th IEEE Int. Perf. Comput. Commun. Conf.

Yun, D., et al., 2015. Profiling transport performance for big data transfer over dedicated channels. In: Proc. Int. Conf. Comput., Netw. Commun. pp. 858–862.

Yun, D., et al., 2016. Profiling optimization for big data transfer over dedicated channels. In: Proc. 25th Int. Conf. Comput. Commun. Netw.

Yun, D., et al., 2019. Advising big data transfer over dedicated connections based on profiling optimization. IEEE Trans. Netw. 27 (6), 2280–2293.

Yun, D., et al., 2020. Performance prediction of big data transfer through experimental analysis and machine learning. In: Proc. IFIP Networking Conf. pp. 181–189.

Zuo, L., et al., 2018. Bandwidth reservation strategies for scheduling maximization in dedicated networks. IEEE Trans. Netw. Serv. Manag. 15 (2), 544–554.