Data Transfer Advisor with Transport Profiling Optimization

Daqing Yun†, Chase Q. Wu‡, Nageswara S.V. Rao§, Qiang Liu§, Rajkumar Kettimuthu¶, and Eun-Sung Jung¶
†Harrisburg University ‡New Jersey Institute of Technology §Oak Ridge National Laboratory ¶Argonne National Laboratory
Email: dyun@harrisburgu.edu, chase.wu@njit.edu, raons@ornl.gov, liuq1@ornl.gov, kettimut@anl.gov, esjung@anl.gov

***Abstract*—The network infrastructures have been rapidly upgraded in many high-performance networks (HPNs). However, such infrastructure investment has not led to corresponding performance improvement in big data transfer, especially at the application layer, largely due to the complexity of optimizing transport control on end hosts. We design and implement ProbData, a PRofiling Optimization Based DAta Transfer Advisor, to help users determine the most effective data transfer method with the most appropriate control parameter values to achieve the best data transfer performance. ProbData employs a profiling optimization-based approach to exploit the optimal operational zone of various data transfer methods in support of big data transfer in extreme-scale scientific applications. We present a theoretical framework of the optimized profiling approach employed in ProbData as well as its detailed design and implementation. The advising procedure and performance benefits of ProbData are illustrated and evaluated by proof-of-concept experiments in real-life networks.**

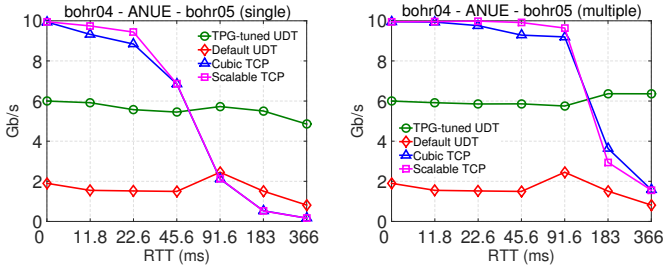
***Index Terms*—Big data transfer, profiling optimization, data transfer advising, high-performance networks**

I. Introduction

High-performance networking (HPN) technologies and services featuring advance bandwidth reservation such as OSCARS [3] in ESnet [1] are being rapidly developed and deployed across the nation and around the globe to support big data transfer in extreme-scale scientific applications. To reap the benefits of such HPN technologies and services, a number of high-performance data transfer protocols and methods have emerged, including TCP variants such as Scalable TCP [15] and UDP-based protocols such as UDT [11]. However, end-to-end data transfer is a complex process that involves many components, some of which may require significant system and network knowledge for parameter tuning and configuration. End users are typically domain experts who lack such knowledge and may find it very difficult to determine what data transfer method to use and what control parameter values to set in order to achieve satisfactory data transfer performance over high-speed dedicated connections in high-performance networks.

To illustrate the effects of different transport methods on end-to-end data transfer performance, we compare in Fig. 1 the maximum throughput performance achieved by two TCP variants (i.e., Cubic TCP [13] and Scalable TCP [15]), default UDT [7], [11], and TPG-tuned UDT[1] using both single and multiple data streams over various connections with different Round Trip Time (RTT) delays emulated between host `bohr04` and host `bohr05` at Oak Ridge National Laboratory (ORNL). We observe that TCP outperforms UDT with default settings for short RTTs; UDT is not as sensitive to RTT as TCP; and

[1]The control parameter values of UDT are determined based on transport profiling conducted by the TPG toolkit [21].



(a) (b)

Fig. 1. Comparison of maximum throughput between Cubic TCP, Scalable TCP, default UDT, and TPG-tuned UDT over 10 Gbps emulated connections with different RTTs at ORNL: (a) single stream, (b) multiple streams.

TPG-tuned UDT outperforms TCP beyond a certain RTT. These measurements indicate that UDT is generally more suitable than TCP for big data transfer (although requires tuning) over long-haul high-speed dedicated connections.

Moreover, control parameter values may also play a significant role in determining end-to-end transfer performance. As shown in Fig. 1(a), with the default settings, UDT achieves a slightly higher performance (around 1 Gbps) than TCP over connections of long RTTs using a single stream, which is far below the connection bandwidth of 10 Gbps. Using multiple streams, TCP outperforms default UDT for all RTTs, as shown in Fig. 1(b). However, if we carefully tune the control parameter values of UDT, e.g., using TPG [21], over long-haul (e.g., longer than 90 ms) connections, UDT is able to outperform TCP in both single- and multi-stream cases.

To further illustrate and investigate the effects of control parameter values on data transfer performance, we plot in Fig. 2 the performance comparison of UDT in response to different buffer sizes over connections of different RTTs emulated between two other hosts `feynman1` and `feynman2` at ORNL. We observe that the buffer space needed by UDT [7] to achieve the peak performance increases as the RTT increases. This behavior is different from traditional transport protocols such as TCP, where the increase of buffer space does not significantly affect the transport performance after reaching a certain point such as the bandwidth-delay product (BDP) over a given connection.

Figs. 1 and 2 show that the performance of both TCP and UDT over high-speed dedicated connections is sensitive to network environments (e.g., connection delays) and significantly affected by control parameters. Therefore, it is critical to identify a suitable data transfer method together with a set of appropriate control parameter values to achieve satisfactory transport performance (mainly throughput) at the application layer. However, it is not straightforward to determine the

optimal control parameter values, e.g., the buffer size for UDT. As shown in Fig. 2, an "over-sized" buffer might slow down the data transfer speed observed by the end user. In addition, due to the lack of accurate models for high-performance transport protocols such as UDT [7], [11], which is widely adopted in the HPN community [6], and the complex dynamics of network environments, it is generally very difficult to derive the optimal operational zone using an analytical approach.

Transport profiling, which sweeps through the entire space of the control parameter values of a given transport method, has been proved to be useful [21] and may be used for transport selection and parameter setting to support big data transfer in high-performance networks (HPNs) [20]. However, exhaustive profiling is prohibitively time consuming when there exists a large parameter space and is impractical to meet data transfer requirements of scientific applications in network environments that are subject to frequent changes (e.g., configurations of the sender or receiver host, connection delay, connection bandwidth, etc.). Generally, it is not favorable to conduct exhaustive profiling when the overhead of "on-line" profiling is comparable with the data transfer time itself.

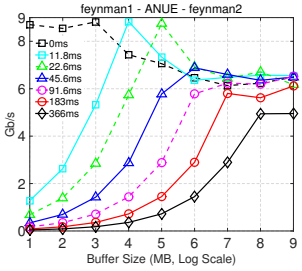


Fig. 2. UDT profiling on buffer size over 10 Gbps emulated connections of different delays at ORNL.

We propose a profiling optimization based data transfer advisor, referred to as ProbData, to identify the most suitable transport method and the most appropriate control parameter values for a given data transfer request. ProbData supports both TCP and UDT protocols and is developed on top of two profiling toolkits, i.e., Transport Profile Generator (TPG) [5], [21] and iperf3 [2]. Specifically, ProbData conducts memory-to-memory data transfer profiling on TCP using iperf3 and on UDT using TPG, respectively. To further improve the efficiency of transport profiling, ProbData employs the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [18] to accelerate the exploration of parameter space. We first discuss a theoretical framework of the optimized profiling approach employed in ProbData, and then present the design and implementation details of ProbData. The advising procedure and performance benefits of ProbData are illustrated by proof-of-concept experiments in real-life networks.

The rest of the paper is organized as follows. Section II introduces transport profiling. Section III describes SPSA-based profiling optimization. Section IV presents the design and implementation of ProbData. Section V illustrates the advising procedure and evaluates the performance benefits of ProbData.

II. Transport Profiling

End-to-end data transfer is a complex process that involves various network segments and end-host components, whose parameter settings have a significant impact on the end-to-end performance observed by end users at the application layer. It is difficult to decide transport selection and parameter setting using an analytical approach due to complex system dynamics and frequent changes in network environments. Parameter tuning may help achieve a better performance, but it typically requires extensive network and system knowledge that many science users lack. Moreover, even if they are able to manually conduct "fine tuning" on some aspects such as core affinities [8], [14] and IRQ balance/conflict [17] at the system level, many application-level control parameters may still affect end-to-end performance to a large degree.

A transport profile $TP_t(\langle h_s, h_r\rangle, e, \theta)$ is a control-response plot illustrating how a set of control parameters θ affect the performance of a given transport protocol t over a network connection or link e between a sender host h_s and a receiver host h_r. Such a profile indicates the quantitative and qualitative behavior of each component involved in the data transfer process and provides an insight into maximizing the overall transport performance, which can be obtained by exhausting the combinations of the parameter values and collecting the corresponding performance measurements. Every data point in the profile is produced by a "one-time profiling" that sends a certain amount of data with a specific combination of parameters θ during time interval $[0, \Delta T]$ and measures the average throughput performance as

$$G(\theta) = \frac{\int_0^{\Delta T} s(x,\theta)\ dx}{\Delta T},$$

where $s(x, \theta)$ is the sending rate with respect to parameter θ at time point x.

The goal of transport profiling is to find the parameter values θ^*, at which the throughput $G(\theta^*)$ reaches its global maximum. Exhaustive transport profiling is able to find the optima, but is too time consuming for practical use.

As a numerical example, the UDT [11] protocol includes a few commonly accessible parameters including packet size ($m \in \{m_1, m_2, \cdots, m_{N_m}\}$), block size ($l \in \{l_1, l_2, \cdots, l_{N_l}\}$), buffer size ($f \in \{f_1, f_2, \cdots, f_{N_f}\}$), and number of parallel data streams ($p \in \{p_1, p_2, \cdots, p_{N_p}\}$). If a one-time profiling takes Δt (typically on the order of several minutes) to finish, it takes a total of $\Delta t \cdot N_m \cdot N_l \cdot N_f \cdot N_p$ to generate a complete profile prior to the actual data transfer. In the emulations conducted in [20], we fix the packet size m (i.e., $N_m = 1$) and the number of parallel data streams p (i.e., $N_p = 1$), and only vary the block size from 1 to 25 times of the payload size (i.e., $N_l = 25$) and the buffer size from 1.0 MB to 1.0 GB with a 2.0 MB step (i.e., $N_f = 513$). If a one-time profiling takes $\Delta t = 2$ minutes, the exhaustive search would take 25,650 minutes (around 18 days!). As both the number of control parameters and the profiling resolution increase, the time to produce a complete profile rapidly increases, making the exhaustive search-based approach practically infeasible. Therefore, we focus on the design of transport profiling with minimized profiling time to achieve satisfactory data transfer performance.

III. Profiling Optimization Based On Stochastic Approximation

We conduct transport profiling at the application layer rather than system tuning at lower layers. The entire data transfer process could be treated as a "black box" system, where the input is the set of control parameters θ and the output is the corresponding throughput measurement $G(\theta)$. Based on this model, it is appropriate to use the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [18] to quickly determine the optimal parameter values because: i) it does not require an explicit formula of $G(\theta)$, which is unavailable in practice, but only "noise-corrupted" measurements $y = G(\theta) + \xi$, which can be obtained by running a "one-time profiling" using existing tools such as iperf3 and TPG with a set of specified parameter values; ii) it does not require any additional information about system dynamics or input distribution. These are highly desirable features as they account for the dynamics in the data transfer process and the randomness in various network environments and performance measurements.

A. Stochastic Approximation (SA) Methods

Suppose that the average throughput performance G is a function of control parameter set θ, i.e., $G = G(\theta)$. The goal is to find the control parameter values θ^* that maximize $G(\theta)$ within the feasible space Θ, i.e., $\max_{\theta \in \Theta} G(\theta)$.

Based on the standard Kiefer-Wolfowitz Stochastic Algorithm (KWSA) [16], we have the following multi-variable recursive optimization procedure

$$\hat{\theta}_{k+1} = \hat{\theta}_k + a_k \cdot \hat{g}_k(\hat{\theta}_k), \tag{1}$$

where $a_k > 0$ is a scalar gain coefficient, $g(\theta) \equiv \frac{\partial G(\theta)}{\partial \theta}$ is the gradient of G, $\hat{\theta}_k$ is the set of control parameter values in the k-th iteration, and $\hat{g}(\hat{\theta}_k)$ is an approximation of $g(\theta_k)$.

The "noise-corrupted" performance observation, denoted by $y(\theta)$, is available at any value of $\theta \in \Theta$ and given by

$$y(\theta) = G(\theta) + \xi,$$

where ξ is the noise incurred by the randomness in networks and the dynamics in end-host systems. In fact, $y(\theta)$ is the observed average throughput performance of a one-time data transfer profiling with a specific θ during a specific duration.

The gradient $g(\theta)$ of $G(\theta)$ is approximated by an appropriate finite difference given by

$$\hat{g}_k(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k) - y(\hat{\theta}_k - c_k)}{2c_k},$$

where c_k is a small positive number.

The coefficients a_k and c_k in the above equations should satisfy the following conditions to guarantee the convergence,

$$\lim_{k\to\infty} a_k = 0, \lim_{k\to\infty} c_k = 0, \sum_{k=1}^{\infty} a_k = \infty, \sum_{k=1}^{\infty} (\frac{a_k}{c_k})^2 < \infty. \tag{2}$$

Based on the above stochastic approximation method, ProbData utilizes SPSA [18], [19] to further reduce profiling overhead: instead of collecting observations along all dimensions of the gradient, it randomly perturbs the control parameter set in two opposite directions and collects corresponding performance measurements to approximate the gradient, i.e.

$$\hat{g}_k(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k - c_k\Delta_k)}{2c_k} \begin{bmatrix} \Delta_{1,k}^{-1} \\ \Delta_{2,k}^{-1} \\ \vdots \\ \Delta_{d,k}^{-1} \end{bmatrix},$$

where the coefficient sequence $\{\Delta_{i,k}\}$ ($i = 1, \cdots, d$ for d dimensional control parameters) are independent and symmetrically distributed around 0 with finite inverse $E|\Delta_{i,k}^{-1}|$ over all parameter components i and time steps (i.e., iterations) k. ProbData decides each component of $\{\Delta_{i,k}\}$ based on the symmetric Bernoulli ± 1 distribution with a probability of 0.5 for each outcome of either +1 or –1 [19], which has been proven to be simple and effective [20].

For a given data transfer protocol t and its corresponding control parameter set $\theta_t = [\theta_{t,1}, \theta_{t,2}, \cdots, \theta_{t,d}]^\top$, we define the following multi-variable recursive optimization procedure for SA-based profiling with iteration index k:

$$\hat{\theta}_{t,i,k+1} = \hat{\theta}_{t,i,k} + a_k\, \hat{g}_{t,i,k}(\hat{\theta}_{t,i,k}),\ i = 1, 2, \cdots, d,$$

and the corresponding gradient approximation of $G(\theta)$:

$$\hat{g}_{\theta_{t,i,k}}\left(\hat{\theta}_{t,i,k}\right) =$$

$$\frac{y\left(\begin{bmatrix} \hat{\theta}_{t,1,k} \\ \hat{\theta}_{t,2,k} \\ \vdots \\ \hat{\theta}_{t,d,k} \end{bmatrix} + c_k \begin{bmatrix} \Delta_{1,k} \\ \Delta_{2,k} \\ \vdots \\ \Delta_{d,k} \end{bmatrix}\right) - y\left(\begin{bmatrix} \hat{\theta}_{t,1,k} \\ \hat{\theta}_{t,2,k} \\ \vdots \\ \hat{\theta}_{t,d,k} \end{bmatrix} - c_k \begin{bmatrix} \Delta_{1,k} \\ \Delta_{2,k} \\ \vdots \\ \Delta_{d,k} \end{bmatrix}\right)}{2c_k\Delta_{i,k}},$$

where $i = 1, \cdots, d$ for d dimensions of control parameter θ.

B. UDT and TCP Profiling based on SPSA

The profiling for two major data transfer protocols, TCP and UDT, are supported in the current version of ProbData. TCP is the *de facto* standard transport protocol on the Internet and UDT [7] is a high-performance UDP-based data transfer protocol widely adopted in HPN community [6]. ProbData recommends the better protocol choice based on: i) historical profiling data, ii) online profiling results, and iii) several well-known (user-specified optional) rules.

Based on existing profiling results of UDT on various control parameters including packet size (m), block size (l), buffer size (f), and number of parallel data streams (p) over various network environments, one may decide some parameter values without profiling. In particular, if a jumbo frame is supported along the path for a given data transfer, it is desirable to enable it to minimize per packet overhead [9]. Hence, the packet size m can be decided by exploring the Path MTU (PMTU), i.e.,

$$\begin{cases} m_{TCP} = MTU - 40; \\ m_{UDT} = MTU - 44. \end{cases} \tag{3}$$

Since UDT is not best suited for environments with a high level of concurrency [12], we focus on UDT profiling in a single-stream case (i.e., $p = 1$). Therefore, the control parameter set for UDT includes block size (l) and buffer size (f), i.e.,

$\theta_{UDT} = [l, f]^\top$, and we have the following 2-variable recursive optimization procedure for SPSA-based UDT profiling:

$$\begin{bmatrix} \hat{l}_{k+1} \\ \hat{f}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{l}_k \\ \hat{f}_k \end{bmatrix} + a_k \begin{bmatrix} \hat{g}_{l_k}\left(\begin{bmatrix} \hat{l}_k \\ \hat{f}_k \end{bmatrix}\right) \\ \hat{g}_{f_k}\left(\begin{bmatrix} \hat{l}_k \\ \hat{f}_k \end{bmatrix}\right) \end{bmatrix}, \tag{4}$$

and the corresponding gradient approximation $\hat{g}(\hat{\theta})$:

$$\hat{g}_k(\hat{\theta}_{UDT,k}) = \begin{bmatrix} \hat{g}_{l_k}\left(\begin{bmatrix} \hat{l}_k \\ \hat{f}_k \end{bmatrix}\right) \\ \\ \hat{g}_{f_k}\left(\begin{bmatrix} \hat{l}_k \\ \hat{f}_k \end{bmatrix}\right) \end{bmatrix} = \begin{bmatrix} \dfrac{y_k^+ - y_k^-}{2c_k\Delta_{l,k}} \\ \\ \dfrac{y_k^+ - y_k^-}{2c_k\Delta_{f,k}} \end{bmatrix}, \tag{5}$$

where y^+ and y^- are calculated as

$$\begin{cases} y_k^+ = y(\hat{\theta} + c_k\Delta_k) = y\left(\begin{bmatrix} \hat{l}_k + c_k\Delta_{l,k} \\ \hat{f}_k + c_k\Delta_{f,k} \end{bmatrix}\right) \\ \\ y_k^- = y(\hat{\theta} - c_k\Delta_k) = y\left(\begin{bmatrix} \hat{l}_k - c_k\Delta_{l,k} \\ \hat{f}_k - c_k\Delta_{f,k} \end{bmatrix}\right) \end{cases}. \tag{6}$$

Note that ProbData conducts such SPSA-based UDT profiling by leveraging the capabilities of TPG.

Similarly, based on our previous profiling results, the effect of packet size (m) and block size (l) for TCP is not as critical as that for UDT, and the socket buffer size (w) and number of parallel data streams (p) play a critical role on the end-to-end throughput performance. ProbData performs SPSA-based TCP profiling on these two control parameter, i.e., $\theta_{TCP} = [w, p]^\top$. The corresponding 2-variable recursive optimization procedure and the gradient approximation are defined as follows:

$$\begin{bmatrix} \hat{w}_{k+1} \\ \hat{p}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{w}_k \\ \hat{p}_k \end{bmatrix} + a_k \begin{bmatrix} \hat{g}_{w_k}\left(\begin{bmatrix} \hat{w}_k \\ \hat{p}_k \end{bmatrix}\right) \\ \hat{g}_{p_k}\left(\begin{bmatrix} \hat{w}_k \\ \hat{p}_k \end{bmatrix}\right) \end{bmatrix}, \tag{7}$$

$$\begin{cases} y_k^+ = y(\hat{\theta} + c_k\Delta_k) = y\left(\begin{bmatrix} \hat{w}_k + c_k\Delta_{w,k} \\ \hat{p}_k + c_k\Delta_{p,k} \end{bmatrix}\right) \\ \\ y_k^- = y(\hat{\theta} - c_k\Delta_k) = y\left(\begin{bmatrix} \hat{w}_k - c_k\Delta_{w,k} \\ \hat{p}_k - c_k\Delta_{p,k} \end{bmatrix}\right) \\ \\ \hat{g}_k(\hat{\theta}_{TCP,k}) = \begin{bmatrix} \hat{g}_{w_k}\left(\begin{bmatrix} \hat{w}_k \\ \hat{p}_k \end{bmatrix}\right) \\ \\ \hat{g}_{p_k}\left(\begin{bmatrix} \hat{w}_k \\ \hat{p}_k \end{bmatrix}\right) \end{bmatrix} = \begin{bmatrix} \dfrac{y_k^+ - y_k^-}{2c_k\Delta_{w,k}} \\ \\ \dfrac{y_k^+ - y_k^-}{2c_k\Delta_{p,k}} \end{bmatrix} \end{cases}. \tag{8}$$

Note that ProbData conducts SPSA-based TCP profiling based on iperf3, which has a similar profiling process as TPG. An outline of the SPSA-based profiling algorithm is provided in Alg. 1 with more details in Sec. IV.

C. Termination Conditions

ProbData employs the following three simple and practical termination conditions to guarantee the performance and the termination of the SPSA-based UDT/TCP profiling.

1) Performance Gain Ratio (PGR): the performance gain ratio $C(0 < C < 1)$ is defined as $C = \dfrac{y}{y^*}$, where y is the observed throughput of a one-time profiling, and y^* is the best throughput performance of a given data transfer method over a given network connection, which actually is unknown until a complete transport profile is obtained. In ProbData, we set y^* to be the connection bandwidth[2]. When it reaches an operational zone that results in a throughput y with a PGR no less than a certain user-specified value, SPSA-based profiling stops. Note that this condition may or may not be satisfied in a certain profiling.
2) Impeded progress: ProbData terminates TCP/UDT profiling when the number of consecutive iterations that do not produce any performance improvement over the best one observed so far exceeds an upper bound L.
3) Upper bound: ProbData terminates TCP/UDT profiling when the total number of profiling iterations exceeds a threshold N.

Algorithm 1 SPSA-based TCP/UDT profiling

1: $k \leftarrow 0$, $\hat{\theta}_0 \in \Theta$; // randomly pick a starting point [20]
2: **while** *termination conditions* are not met **do**
3: $a_k \leftarrow \dfrac{a}{(A+k+1)^\alpha}$; $c_k \leftarrow \dfrac{c}{(k+1)^\gamma}$;
4: Generate a pair of perturbations $\Delta_k \in \{+1, -1\}$ following the symmetric Bernoulli ± 1 distribution with a probability of 0.5 for each outcome;
5: Run one-time profiling twice to collect two observations y_k^+ and y_k^-;
6: Generate simultaneous perturbations to approximate the unknown gradient $g(\hat{\theta}_k)$ using Eq. 5 / Eq. 8;
7: Apply the standard stochastic approximation form in Eq. 1 to update $\hat{\theta}_k$ to a new value $\hat{\theta}_{k+1}$;
8: $k \leftarrow k + 1$;

D. Preventing Local Optima

In addition to the aforementioned three termination conditions, we take a simple but "scalable" approach to move the profiling process out of a local optima region under the constraints of L and N. Specifically, if no performance improvement is observed after a certain number of consecutive iterations that have reached a certain fraction of L, we enlarge the profiling step sizes by certain factors $\tau_a > 1$ and $\tau_c > 1$, i.e., $a = a \cdot \tau_a$ and $c = c \cdot \tau_c$. This operation repeats until ProbData yields a better performance or one of the conditions in Sec. III-C is met.

IV. Implementation and Operating Procedure

ProbData is implemented with 18,000+ lines of C/C++ code in Linux and is available for download at [4].

A. Overview

As shown in Figure 3, ProbData integrates several existing profiling toolkits such as TPG [5], [21] and iperf3 [2]. To obviate the need of conducting exhaustive transfer profiling, ProbData employs SPSA [18], [20] to realize optimized profil-

[2]The bandwidth of a dedicated connection in HPNs is considered as a constant since it is reserved in advance and provisioned in real time.

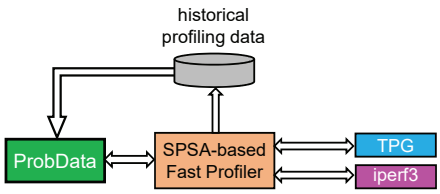


Fig. 3. Design of ProbData.

ing approaches, with the following major components:

- ProbData drives the entire advising process and work with a command-line user interface. It first searches in the historical profiling data for data transfer advising to avoid unnecessary on-line profiling and then perform on-line profiling if the recommended settings are not satisfactory.
- The SPSA-based fast profiler employs Alg. 1 to conduct profiling for TCP and UDT based on TPG and iperf3.
- The historical profiling database stores all profiling results as the advising progresses.
- The UDT profiler employs TPG.
- The TCP profiler employs iperf3.

B. High Level Control Logic

Upon the arrival of a user request, if there exist some historical profiling data that match the request, ProbData searches the historical profiling data for the best data transfer method and corresponding control parameter values, and presents the advising results to the user. It conducts on-line data transfer profiling only if the advising results are not satisfactory to the user. If no historical data exists, ProbData employs Alg. 1 to carry out on-line data transfer profiling for the request.

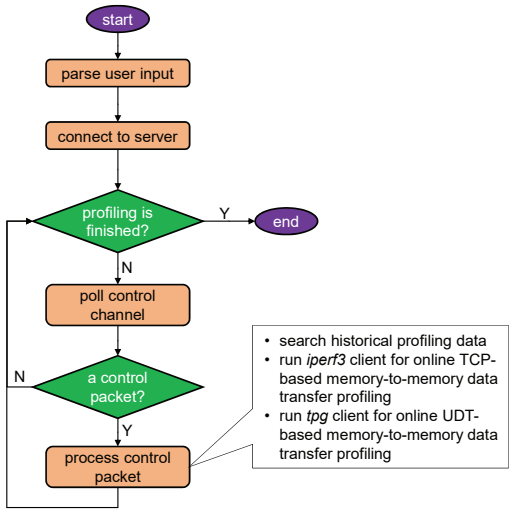
ProbData consists of a pair of sender and receiver, which communicate with each other to exchange control parameters and move the profiling process forward through a TCP-based control channel. The client and server are also responsible for running the corresponding client and server of TPG and iperf3 to conduct memory-to-memory data transfer profiling. The main steps and control flow charts of the client and server of ProbData are shown in Figs. 4(a) and 4(b), respectively. The entire profiling process is mainly driven by the ProbData client, in which each step is acknowledged by the ProbData server prior to the actual execution.

C. Functional Components

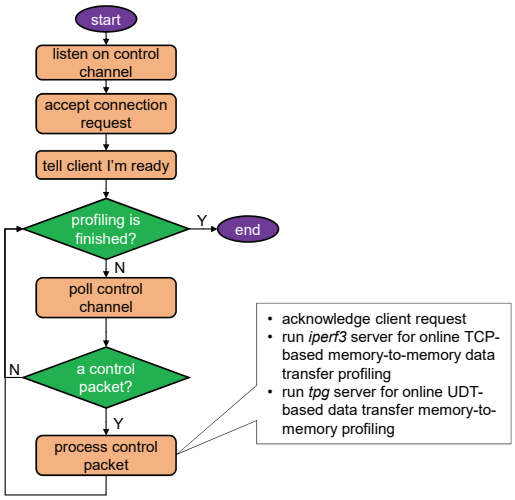
ProbData supports memory-to-memory data transfer profiling for TCP and UDT in the current version. As shown in Fig. 3, ProbData uses iperf3 [2] to perform profiling for TCP and uses TPG [21] to perform profiling for UDT [7].

1) Iperf3

Iperf3, developed at ESnet [1], is a toolkit for actively measuring the maximum achievable bandwidth over a network connection using TCP, UDP, or SCTP. It is a rewrite from scratch of the original well-known iperf, but with a relatively smaller and simpler code base and a library that can be incorporated in other programs. Iperf3 supports various parameters including packet size, block size, buffer size, and number of



(a) client



(b) server

Fig. 4. Control flow charts of ProbData.

parallel data streams. ProbData incorporates iperf3 library in the implementation to repeatedly conduct "one-time" TCP-based profilings to realize SPSA-based profiling. In each iteration, the control parameter values are calculated based on Eqs. 7 and 8, and the resulted achievable bandwidth (i.e., the average throughput performance) is reported and stored.

2) TPG

Transport Profile Generator (TPG) [21], similar to iperf3, is a toolkit for conducting transport profiling using UDT. In addition to those control parameters as in iperf3, TPG supports tuning of UDP socket options and other UDT protocol-specific configurations, and also supports transport profiling based on multiple physical NIC-to-NIC connections. Similarly, ProbData repeatedly calls the TPG APIs to conduct "one-time" UDT-based profiling to realize SPSA-based profiling. In each iteration, the control parameter values are calculated based on Eqs. 4, 5, and 6, and the resulted average throughput performance is reported and stored as well.

D. Parameters

Extensive profiling results (e.g., [20], [21]) have shown that if carefully tuned using TPG [21] or FastProf [20], high-performance protocols such as UDT can improve the transport performance over default parameter setting and may also outperform traditional data transfer protocols such as TCP and its variants. Such performance can be consistently achieved when the control parameter values are within certain ranges. Therefore, ProbData should rule out the unnecessary profiling space based on network domain knowledge, big data transfer properties, and past profiling experiences. In this section, we first present the setting of parameter values for the SPSA-based profiling algorithm as shown in Alg. 1 and Section III, and then describe the approach ProbData takes to adjust the profiling range for the control parameters.

1) Parameter Selection for SPSA-based Method

The coefficients a_k and c_k of the stochastic approximation algorithm need to satisfy the conditions in Eq. 2 to guarantee the convergence. As shown in Alg. 1, we set these coefficients as follows:

$$a_k \leftarrow \frac{a}{(A+k+1)^{\alpha}}, \quad c_k \leftarrow \frac{c}{(k+1)^{\gamma}},$$

where $\alpha = 0.602$ and $\gamma = 0.101$ as suggested by Spall in [19], and the step sizes a and c are empirically determined based on the size of the entire search space. In the current version of ProbData, we set $a = 15.0$ and $c = 5.5$ by default, and they are also tunable through command-line options. We also use a normalization technique to scale the values of a, c, and θ_k in the cases where different dimensions of the control parameters have different magnitudes. The value of A should be much less than the expected/allowed number of iterations in Alg. 1 and we set $A = 0.0$ by default. The iteration index k starts from 0. A more detailed guideline for setting the parameters of the SPSA-based method could be found in [19].

2) Control Parameter Calculation

We set the control parameters in the SPSA-based Alg. 1, denoted by $\theta'_{UDT} = [l',\, f']^{\top}$ and $\theta'_{TCP} = [w',\, p']^{\top}$, to be positive numbers within a reasonably selected range (see Section IV-D3) to ensure a comparable magnitude of each parameter. These "iterative" parameters are scaled and mapped to decide the actual control parameter values in each iteration. We perform a rounding operation in calculating the actual values of the control parameters $\theta_{UDT} = [l,\, f]^{\top}$ and $\theta_{TCP} = [w, p]^{\top}$ in the cases where the intermediate results are fractional.

The profiling unit of block size, denoted by μ_l, is defined as one payload size of the data transfer protocol being profiled, which is given by Eq. 3 for UDT and TCP. The block size $l\,(l \geq 1)$ is defined as an integer multiplicity of the payload size. For a UDP-based protocol such as SABUL [10] or UDT [7], it is recommended to set the block size to be a multiplicity of the protocol's payload size if possible to avoid UDP automatic segmentation and improve the performance. Note that the block size is the number of bytes that are transferred by ProbData by calling the `appsend()/apprecv()` API functions of the lower layer profilers (i.e., TPG and iperf3), in which the `send()/recv()` API functions of the underlying data transfer protocol may be called several times to completely send an entire data block.

Similarly, the profiling unit of buffer size, denoted by μ_f and μ_w for UDT and TCP, respectively, is decided by the user within a feasible profiling range, e.g., 1 Byte, 512 KB, 1.0 MB, 2.0 MB, or others. We set it to be 1 Byte as default.

The profiling unit of parallel data stream number, denoted by μ_p, is simply set to be 1.

Based on the above profiling units, we calculate the actual values of block size (l), buffer size (f, w), and number of streams (p) for performance observations (i.e., the calculations of y^+ and y^- through one-time profilings) as follows:

$$\begin{cases} l = round\,(\lambda_l(l') \cdot \mu_l) \\ f = round\,(\lambda_f(f') \cdot \mu_f) \\ w = round\,(\lambda_w(w') \cdot \mu_w) \\ p = round\,(\lambda_p(p') \cdot \mu_p) \end{cases}, \tag{9}$$

where λ are scaling functions that may take different profiling patterns. For example, with a function $\lambda_f(f') = 2^{f'}$, the buffer size would exponentially increase as f' increases.

In ProbData, we use a linear normalization approach to perform the scaling functions to calculate the actual values of the control parameters. For example, the UDT buffer size (f) in unit of bytes is calculated as

$$f = round\,[\lambda_f(f') \cdot \mu_f] = \\ round\left(\left[(f' - f'_{min}) \cdot \frac{f_{max} - f_{min}}{f'_{max} - f'_{min}} + f_{min}\right] \cdot \mu_f\right), \tag{10}$$

where the iterative value f' is the one used in Alg. 1, and $[f'_{min}, f'_{max}]$ and $[f_{min}, f_{max}]$ specify the profiling ranges for the "iterative" and actual UDT buffer size, respectively.

3) Control Parameter Ranges

In the implementation of ProbData, we set the iterative profiling range to be $[1.0,\, 25.0]$ based on extensive profiling results, i.e., $l'_{min} = f'_{min} = w'_{min} = p'_{min} = 1.0$ and $l'_{max} = f'_{max} = w'_{max} = p'_{max} = 25.0$.

The number of parallel data streams is simply set from 1 to 25 by default. A larger number is optional, depending on the user preference.

If UDT is being profiled, the profiling unit μ_l of block size is 8,956 bytes (see Eq. 3). The profiling range in unit of bytes is from $1 \times 8,956$ bytes to $25 \times 8,956 = 223,900$ bytes, and other parameters in between are calculated similarly as Eq. 10. The range for TCP is from 8,960 bytes to 224,000 bytes.

We set the profiling range of buffer size for both UDT and TCP to be from $f_{min} = w_{min} = 16$ KB to $f_{max} = w_{max} = 2$ GB[3] by default to accommodate various delays of network connections of different bandwidths up to 40 Gbps and higher. In addition, to improve the profiling efficiency, ProbData adaptively makes adjustment to restrict the profiling range of the buffer size based on the current network status. In particular, ProbData estimates the average RTT using ICMP echo requests, based on which, it chooses the profiling range of buffer sizes

[3]This is the maximal value given that the buffer size is represented by a 32-bit integer.

for TCP and UDT covering the BDP as follows,

$$\begin{cases} f \in [\omega_1 \cdot BDP, \omega_2 \cdot BDP] \\ w \in [\omega_1 \cdot BDP, \omega_2 \cdot BDP] \end{cases}, \quad (11)$$

where BDP is the estimated bandwidth-delay product (BDP), and $0 < \omega_1 < 1$ and $\omega_2 > 1$. Note that in HPNs, the reserved bandwidth is specified by the user through the command-line option `-y` of ProbData; otherwise, ProbData rolls back to the default profiling range for buffer size.

The RTT estimation is conducted in each iteration of SPSA-based profiling and updated using moving average to reflect the current network status, i.e.,

$$SRTT_{i+1} = (1 - \beta) \cdot SRTT_i + \beta \cdot RTT_i,$$

where $SRTT_i$ is the estimated/smoothed RTT in iteration i and RTT_i is the newly measured RTT in iteration i. The initial RTT_0 could be either provided by the user (by setting option `-T`) or measured by ProbData at its initialization stage. The weighting factor β ($0 \leq \beta \leq 1$) is chosen to be 0.125 in the current implementation of ProbData.

E. Historical Data Storage

ProbData keeps track of the entire process by saving all intermediate one-time profiling results and the final advising results as historical profiling data in human-readable format (text file) for future advising use. The historical profiling data files are stored in a separate folder `profile` within the ProbData software package. We create a subfolder `mem` to store each record of one-time memory-to-memory data transfer profiling, under which, the profiling results for the same connection (defined by a source-destination IP pair) are stored in a separate text file. Each record of a one-time profiling result takes one line in the file and the most recently achieved records are appended to the end of the file. The profiling results of all one-time profilings over the same connection are stored as time-stamped records in the same profiling data file. The subfolder may contain multiple text files that are labeled by the source-destination IP pair.

V. Performance Evaluations

In this section, we illustrate the advising and recommendation process and evaluate the performance benefits of ProbData.

A. Data Transfer Advising and Recommendation

We present two experimental case studies over a local 10Gbps back-to-back connection and a 10Gbps wide-area connection between Argonne National Laboratory (ANL) and University of Chicago (UChicago) with a RTT of about 2 ms. We use these two case studies to demonstrate the usage, advising steps, and typical advising and recommendation outputs.

In ProbData, end users can specify the value of C, L, and N in the command line and larger values generally lead to longer profiling time and better performances. We plot the data transfer performance achieved in each iteration in Fig. 5. In Fig. 5(a), we set $C = 0.999$ (a sufficiently large value), $L = 20$, and $N = 40$; while in Fig. 5(b), we set $C = 0.85$ (a reasonable user expectation), $L = 20$, and $N = 40$. Note that L and N limit the total number of iterations and there are total three one-time profilings in each iteration for y^+, y^-, and y^*, respectively.

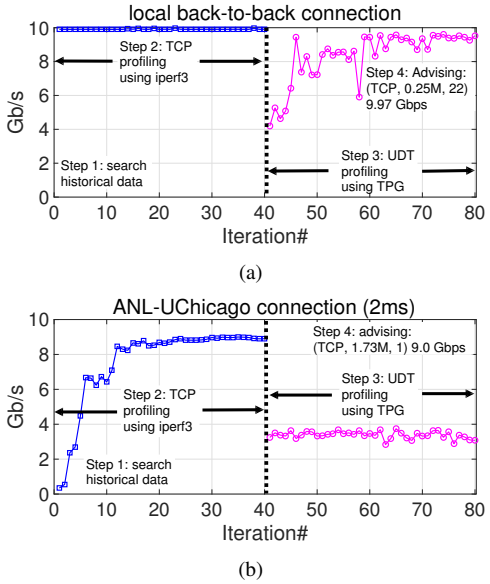


Fig. 5. Profiling and advising process of ProbData.

Given a user request, ProbData conducts data transfer advising and recommendation by taking the following steps:

Step 1 ProbData locates the appropriate historical profiling file and searches for the best option (including data transfer protocol and corresponding control parameter values and performance). If the user is satisfied with the historical results, ProbData stops without performing on-line profiling.

Step 2 If the user is not satisfied or there is no historical profiling data, ProbData lunches SPSA-based TCP profiling using iperf3 following Alg. 1, during which ProbData keeps updating the best option resulted from the entire TCP profiling.

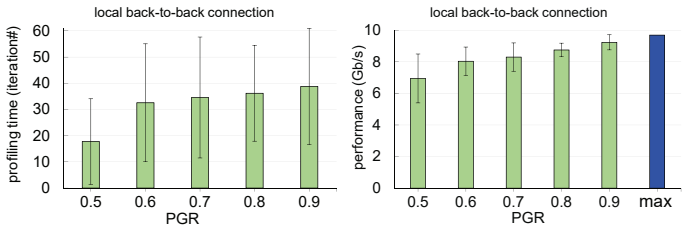
Step 3 ProbData performs SPSA-based UDT profiling using TPG following Alg. 1 as well and keeps updating the best option resulted from the entire UDT profiling.

Step 4 ProbData compares the three best recommendation options resulted from: i) historical profiling data, ii) SPSA-based TCP profiling, and iii) SPSA-based UDT profiling, and then presents the data transfer protocol selection and corresponding control parameter values with the highest throughput.

It is generally difficult to predict the performance of TCP or UDT over a given connection through an analytical approach. The dynamics in different environments necessitate transport profiling to guarantee a satisfactory data transfer performance. The performance and advising results in Fig. 5(a) and Fig. 5(b) show that these connections with short RTTs prefer TCP to UDT. However, as the connection length increases, the protocol choice become more challenging and could also be affected by end hosts and their configurations.

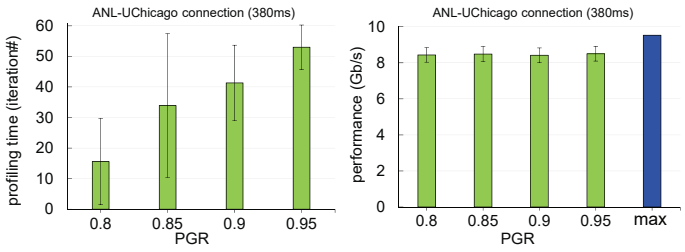
B. Profiling Efficiency and Achieved Performance

As mentioned in Section II, it takes about 18 days to conduct profiling using an exhaustive approach at a coarse-grained



(a) (b)

Fig. 6. Profiling time (number of iterations) and performance of ProbData over a 10Gb/s local back-to-back connection.



(a) (b)

Fig. 7. Profiling time and performance of ProbData over an emulated 10Gb/s connection of 380ms RTT between ANL and UChicago.

profiling resolution. The default profiling process in ProbData does not require any specific profiling resolution as any control parameter value in the feasible space is an option for achieving satisfactory performance. Thus, we set the profiling resolution to be the minimal possible value (i.e., one payload, one byte, and one data stream) in the experiments in this section. We run each test for 10 times, measure the average performance and profiling time (decided by the number of iterations) together with their standard deviations, and plot the results in Figs. 6 and 7. Each comparison includes the average throughput and the best overall throughput, denoted by "max", achieved by ProbData. The results show that ProbData is able to consistently find a set of control parameter values that produce a satisfactory throughput in a short period in various networks.

As shown in Fig. 6, across different values of PGR (C), ProbData is able to discover an appropriate set of control parameter values that result in an average performance between 7+Gbps (for smaller values of C such as 0.5) and 9.0+Gbps (for larger values of C such as 0.9). The average performance achieved by ProbData is comparable with the best overall performance and is quite stable as indicated by the corresponding standard deviations. As shown in Fig. 6(b), larger C values generally lead to better performance, and take longer profiling time (Fig. 6(a)), but compared with the exhaustive profiling approach, the profiling time is significantly reduced from 18 days to 2-3 hours at most.

Similarly, over a long-haul connection of 380ms RTT emulated by looping back between ANL and UChicago, ProbData finds an appropriate set of control parameter values that result in an average performance of 8.0+Gbps, as shown in Fig. 7(b). Here, the PGR values (i.e., C values) are chosen from a set of relatively higher values $\{0.80, 0.85, 0.90, 0.95\}$. The performance difference corresponding to different C values in Fig. 7(b) is not as obvious as those in Fig. 6(b). However, the profiling time differs significantly for different C values, as shown in Fig. 7(a).

VI. Conclusion and Future Work

We designed and implemented ProbData, a profiling optimization based data transfer advisor, which is built on top of existing toolkits including TPG and iperf3 to realize data transfer profiling for both TCP and UDT. The advising procedure and performance benefits of ProbData were illustrated using proof-of-concept experiments in real-life networks. ProbData can help end users determine the best suited data transfer method with appropriate control parameter values for big data transfer. It is of our future interest to investigate and improve the convergence speed of SA-based profiling.

References

[1] Energy Sciences Network. http://www.es.net.

[2] ESnet iperf3. https://github.com/esnet/iperf.

[3] ESnet OSCARS Service. http://www.es.net/oscars.

[4] ProbData: Profiling Optimization Based Data Transfer Advisor. https://github.com/daqingyun/probdata/.

[5] TPG: Transport Profile Generator. https://goo.gl/q917E3.

[6] UDT-Powered Projects. http://udt.sourceforge.net/poweredby.html.

[7] UDT: UDP-based Data Transfer. http://udt.sourceforge.net/.

[8] V. Ahuja, M. Farrens, and D. Ghosal. Cache-aware Affinitization on Commodity Multicores for High-speed Network Flows. In *Proc. of the 8th ACM/IEEE Symp. on Architectures for Networking and Communications Systems*, pages 39–48, 2012.

[9] J. Chase, A. Gallatin, and K. Yocum. End System Optimizations for High-speed TCP. *IEEE Communications Magazine*, 39(4):68–74, 2001.

[10] Y. Gu and R. Grossman. SABUL: A Transport Protocol for Grid Computing. *Journal of Grid Computing*, 1(4):377–386, 2003.

[11] Y. Gu and R. Grossman. UDT: UDP-based Data Transfer for High-speed Wide Area Networks. *Computer Networks*, 51(7):1777–1799, 2007.

[12] Y. Gu, X. Hong, and R. Grossman. Experiences in Design and Implementation of a High Performance Transport Protocol. In *Proc. of the ACM/IEEE Conf. on Supercomputing*, pages 22–35, 2004.

[13] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[14] N. Hanford, V. Ahuja, M. Farrens, D. Ghosal, M. Balman, E. Pouyoul, and B. Tierney. Analysis of the Effect of Core Affinity on High-Throughput Flows. In *Proc. of the 4th Int'l Workshop on Network-Aware Data Management*, pages 9–15, 2014.

[15] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *ACM SIGCOMM Computer Communication Review*, 33(2):83–91, 2003.

[16] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of A Regression Function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[17] B. Leitao. Tuning 10Gb Network Cards on Linux. In *Proc. of the Linux Symp.*, pages 169–184, 2009.

[18] J. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[19] J. Spall. Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):817–823, 1998.

[20] D. Yun, C. Wu, N. Rao, Q. Liu, R. Kettimuthu, and E. Jung. Profiling Optimization for Big Data Transfer Over Dedicated Channels. In *Proc. of the 25th Int'l Conf. on Computer Communication and Networks*, 2016.

[21] D. Yun, C. Wu, N. Rao, B. Settlemyer, J. Lothian, R. Kettimuthu, and V. Vishwanath. Profiling Transport Performance for Big Data Transfer over Dedicated Channels. In *Proc. of the Int'l Conf. on Computing, Networking and Communications*, pages 858–862, 2015.