Profiling Optimization for Big Data Transfer Over Dedicated Channels

Daqing Yun and Chase Q. Wu
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ 07102, USA
{dy83,chase.wu}@njit.edu

Nageswara S.V. Rao and Qiang Liu
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
{raons,liuq1}@ornl.gov

Rajkumar Kettimuthu and Eun-Sung Jung
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439, USA
{kettimut,esjung}@anl.gov

***Abstract*—The transfer of big data is increasingly supported by dedicated channels in high-performance networks, where transport protocols play an important role in maximizing application-level throughput and link utilization. The performance of transport protocols largely depend on their control parameter settings, but it is prohibitively time consuming to conduct an exhaustive search in a large parameter space to find the best set of parameter values. We propose `FastProf`, a stochastic approximation-based transport profiler, to quickly determine the optimal operational zone of a given data transfer protocol/method over dedicated channels. We implement and test the proposed method using both emulations based on real-life performance measurements and experiments over physical connections with short (2 ms) and long (380 ms) delays. Both the emulation and experimental results show that `FastProf` significantly reduces the profiling overhead while achieving a comparable level of end-to-end throughput performance with the exhaustive search-based approach.**

***Index Terms*—Big data transfer, stochastic approximation, profiling, dedicated channels, high-performance networks.**

I. Introduction

Next-generation extreme-scale scientific applications are generating colossal amounts of data, now frequently termed as "big data", which must be transferred over long distances for remote distributed processing. Such data transfer is increasingly supported by high-performance networks (HPNs) featuring on-demand dedicated connections with high bandwidth reserved in advance as exemplified by ESnet [1], Internet2 [3], and Google's B4 [15]. Maximizing the performance of data transfer protocols/methods over dedicated channels is challenging mainly because: i) their optimal operational zones are affected by the complex configurations and dynamics of the network segments, end-hosts, and protocol itself; ii) different parameter settings may lead to very different performances and oftentimes the default parameter setting does not yield the best performance; iii) due to the lack of accurate models for high-performance transport protocols such as UDT [12], which is a widely adopted protocol in the HPN community [5], 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 combinations of parameter settings such as socket options, application-specific parameters, and protocol-specific configurations, enables users to determine the "best" set of parameter values for the optimal data transfer performance. However, it is prohibitively time consuming to conduct an exhaustive search when there exists a large parameter space, which is almost always the case in most transport scenarios. In general, users may not be in favor of performing transport profiling if the profiling overhead is comparable with the time needed for their actual data transfer.

To improve the efficiency of transport profiling, we propose a stochastic approximation-based profiling method, referred to as `FastProf`, to quickly determine the optimal operational zone of a given data transfer protocol/method in dedicated network environments. `FastProf` employs the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [19] to accelerate the exploration of the parameter space. We implement the proposed method by leveraging the existing Transport Profile Generator (TPG) [24], and test it using both emulations with real-life performance measurements and experiments over physical connections with short (2 ms) and long (380 ms) delays. Both the emulation and experimental results show that `FastProf` significantly reduces the profiling overhead while achieving a comparable level of end-to-end throughput performance with the exhaustive search-based approach. `FastProf` makes it possible to conduct "on-line" profiling to support time-critical data transfer, and provides an additional level of intelligence to existing profiling-oriented toolkits such as `iperf3` [2] and `xddprof` [6].

The rest of this paper is organized as follows. In Section II, we describe transport profiling and the motivation of our research. In Section III, we present the detailed design of `FastProf`. We implement `FastProf` in Section IV and evaluate its performance using emulations in Section V and using experiments in Section VI, respectively. We conclude our work and sketch a plan for future research in Section VII.

II. Transport Profiling

A. Why is Profiling Needed?

The 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 level. In general, parameter tuning may help achieve a better performance, but it typically requires extensive domain knowledge in networking and systems that many science users lack. Moreover, even if they are able to manually conduct "fine tuning" on different aspects such as core affinities [7], [14] and IRQ balance/conflict [18] at the system level, many application-level control parameters may still significantly affect the end-to-end performance (mainly throughput). For

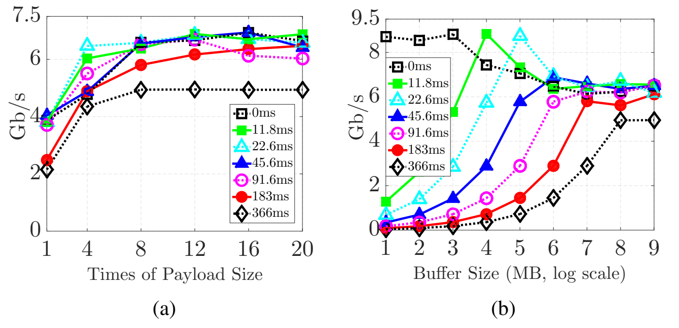
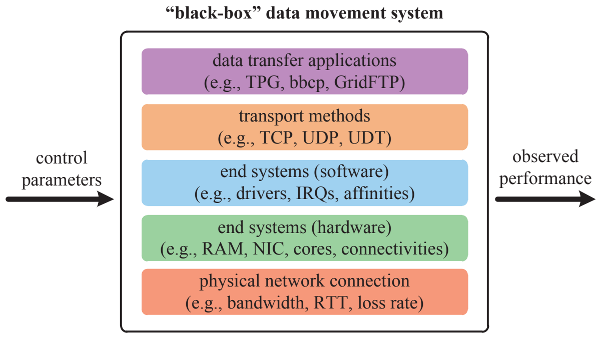


Fig. 1. UDT profiling over ORNL 10 Gb/s emulated connections with various RTTs. **(a)**: throughput vs. block size; **(b)** throughput vs. buffer size.



"black-box" data movement system
data transfer applications
(e.g., TPG, bbcp, GridFTP)
transport methods
(e.g., TCP, UDP, UDT)
end systems (software)
(e.g., drivers, IRQs, affinities)
end systems (hardware)
(e.g., RAM, NIC, cores, connectivities)
physical network connection
(e.g., bandwidth, RTT, loss rate)
control parameters
observed performance

Fig. 2. The "black-box" data movement system.

illustration purposes, we plot in Figure 1 the UDT-based throughput measurements in response to different block sizes and different buffer sizes with different Round Trip Times (RTTs) ranging from 0 ms to 366 ms on an emulated network testbed at Oak Ridge National Laboratory (ORNL). Figure 1(a) shows that different block sizes may result in very different throughput performances when the buffer size is fixed to be 256 MB; and Figure 1(b) shows that different buffer sizes may result in even more significant performance differences with a fixed block size of 89,559 bytes. Although the throughput curves in Figure 1 generally follow a similar pattern, i.e. first increase and then stabilize as the block size increases (Figure 1(a)), and first increase and then decrease as the buffer size increases (Figure 1(b)), the turning points vary with different delays as the block size or the buffer size increases, which makes it difficult to derive an analytical model to determine the optimal parameter settings for block size and buffer size.

A transport profile $TP_t(\langle h_s, h_r\rangle, e, \theta)$ is a control-response plot illustrating how a set θ of control parameters 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 transport profile indicates the qualitative behavior of each component involved in the data transfer process and provides an insight into maximizing the overall transport performance. The transport profile $TP_t(\langle h_s, h_r\rangle, e, \theta)$ can be obtained by exhausting the combinations of the parameter values and collecting the corresponding performance measurements. Each 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

$$G(\theta) = \frac{\int_0^{\Delta T} s(x,\theta)\, dx}{\Delta T}, \tag{1}$$

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

B. Profiling Overhead

The goal of transport profiling is to find the parameter values θ^*, at which the throughput $G(\theta^*)$ reaches its global maximum. An exhaustive search-based transport profiling is able to find the optima, but is very time consuming, and therefore is particularly unsuitable for network environments that are subject to frequent changes (e.g., configurations of the sender or receiver host, connection delay, connection bandwidth, etc.).

As a numerical example, in the Transport Profile Generator (TPG) [24], the selected UDT [12] transport method includes several 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 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 before the actual data transfer. In the emulations in Section V, we fix the packet size m (i.e. $N_m = 1$) and the number p of data streams (i.e. $N_p = 1$), and 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) in total, and hence is impractical in real-life applications. As both the number of control parameters and the profiling resolution increase, the time to obtain a complete profile rapidly increases, making the exhaustive search-based approach practically infeasible.

III. Fast Profiling Based on Stochastic Approximation

To obviate the need of conducting an exhaustive profiling [24], we propose a fast profiling method based on the Simultaneous Perturbation Stochastic Approximation algorithm [19], referred to as `FastProf`, to quickly determine the "best" parameter values prior to actual data transfer.

A. Rationale on the Use of Stochastic Approximation Methods

Figure 2 shows a typical data transfer scenario where a user request, which specifies a sender host and a receiver host, is processed for data transfer in a certain network environment. Since we mainly focus on transport profiling at the application layer rather than system tuning at lower layers, the whole data transfer process could be treated as a "black box" system, where the input is the set of control

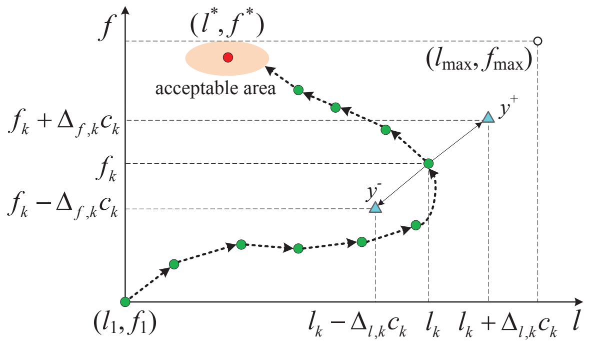


Fig. 3. Visualized profiling process of `FastProf`.

parameters θ and the output is the corresponding throughput measurement $G(\theta)$. Based on this model, the SPSA algorithm is appropriate to be used for quickly determining the optimal parameter values because: i) it does not require an explicit formula of $G(\theta)$ but only its "noise corrupted" measurements $y = G(\theta) + \xi$, which can be obtained by running a "one-time profiling" using existing tools such as `iperf3` [2] and TPG [24] 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 the performance measurements.

For a given data transfer, if the jumbo frame is supported along the path, it is desirable to enable it to minimize per-packet overhead [8]. Thus, the packet size m can be decided by exploring the Path MTU (PMTU) without profiling. Although multiple parallel data streams may improve performance, they typically introduce inter-stream competition that may lead to complex transfer dynamics even over dedicated connections. Furthermore, since many high-performance transport methods including UDT are not best suited for environments with a high level of concurrency [13], we focus our study on one single data stream (i.e. $p = 1$). Similar to TPG [24], where UDT is used in the implementation as a use case, in this paper, we also use UDT as an example transport method and consider block size l and buffer size f as its control parameters (or the input of the black box system in Figure 2), i.e. $\theta = [l, f]^T$.

In the exhaustive search, we need to construct a complete 2D table of performance measurements by running $(N_l \cdot N_f)$ times of one-time profiling, as visualized in Figure 3, where the l-axis and f-axis represent block size and buffer size, respectively. Note that each data point in the graph is actually a 3-tuple $(l, f, G(l, f))$, where $G(l, f)$ is the corresponding observed throughput. The proposed `FastProf` method attempts to explore a path of profiling data points in this 2D table to reach the global optimum within an "acceptable area".

B. Stochastic Approximation (SA) Methods

Based on the model shown in Figure 2, we assume that the average throughput performance is a function $G(\theta)$ of control parameters θ. The goal is to find the control parameters θ^* that maximize $G(\theta)$ within the feasible space Θ, i.e.

$$\max_{\theta \in \Theta} G(\theta). \tag{2}$$

Following the standard Kiefer-Wolfowitz Stochastic Algorithm (KWSA) [16], we have

$$\hat{\theta}_{k+1} = \hat{\theta}_k + a_k \cdot \hat{g}_k(\hat{\theta}_k), \tag{3}$$

where $a_k > 0$ is a scalar gain coefficient, $g(\theta) \equiv \dfrac{\partial G(\theta)}{\partial \theta}$ is the gradient of G, and $\hat{g}(\hat{\theta}_k)$ is an approximation of $g(\theta_k)$.

We assume that the "noise corrupted" observation, denoted as $y(\theta)$, is available at any value of $\theta \in \Theta$, given by

$$y(\theta) = G(\theta) + \xi, \tag{4}$$

where ξ is the noise incurred by the randomness in the network connection and end-host system dynamics. In fact, $y(\theta)$ is the observed average throughput performance of a one-time profiling with a specific θ during a specific time duration $[0, \Delta T]$. Given $\theta = [l, f]^T$, the solution based on the classical KWSA method is a multi-variable recursive optimization procedure, defined as

$$\begin{bmatrix} l_{k+1} \\ f_{k+1} \end{bmatrix} = \begin{bmatrix} l_k \\ f_k \end{bmatrix} + a_k \begin{bmatrix} \hat{g}_{l_k}\left(\begin{bmatrix} l_k \\ f_k \end{bmatrix}\right) \\ \hat{g}_{f_k}\left(\begin{bmatrix} l_k \\ f_k \end{bmatrix}\right) \end{bmatrix}. \tag{5}$$

The gradient $g(\theta)$ of the function $G(\theta)$ is approximated by a "two-sided" finite difference given by

$$\begin{cases} \hat{g}_{l_k}\left(\begin{bmatrix} l_k \\ f_k \end{bmatrix}\right) = \dfrac{\hat{y}\left(\begin{bmatrix} l_k + c_k \\ f_k \end{bmatrix}\right) - \hat{y}\left(\begin{bmatrix} l_k - c_k \\ f_k \end{bmatrix}\right)}{2c_k} \\ \hat{g}_{f_k}\left(\begin{bmatrix} l_k \\ f_k \end{bmatrix}\right) = \dfrac{\hat{y}\left(\begin{bmatrix} l_k \\ f_k + c_k \end{bmatrix}\right) - \hat{y}\left(\begin{bmatrix} l_k \\ f_k - c_k \end{bmatrix}\right)}{2c_k} \end{cases}, \tag{6}$$

where c_k is a small positive number. The coefficients a_k and c_k in the above equations satisfy the following conditions to guarantee the convergence

$$\lim_{k\to\infty} a_k = 0, \ \lim_{k\to\infty} c_k = 0, \ \sum_{k=1}^{\infty} a_k = \infty, \ \sum_{k=1}^{\infty} (\frac{a_k}{c_k})^2 < \infty. \tag{7}$$

We further explore the Simultaneous Perturbation Stochastic Approximation (SPSA) [19], [20] algorithm to further reduce the profiling overhead. Instead of collecting observations along all dimensions of the gradient, SPSA randomly perturbs the control parameter set in two separate directions and collect two corresponding measurements. The gradient approximation of the throughput function based on SPSA is given by

$$\begin{cases} y^+ = y(\theta + \Delta_k c_k) = y\left(\begin{bmatrix} l_k + \Delta_{l,k}c_k \\ f_k + \Delta_{f,k}c_k \end{bmatrix}\right) \\ y^- = y(\theta - \Delta_k c_k) = y\left(\begin{bmatrix} l_k - \Delta_{l,k}c_k \\ f_k - \Delta_{f,k}c_k \end{bmatrix}\right) \end{cases}, \tag{8}$$

$$\begin{cases} \hat{g}_{l_k}(\theta_k) = \hat{g}_{l_k}\left(\begin{bmatrix} l_k \\ f_k \end{bmatrix}\right) = \dfrac{y^+ - y^-}{2\Delta_{l,k}c_k} \\ \\ \hat{g}_{f_k}(\theta_k) = \hat{g}_{f_k}\left(\begin{bmatrix} l_k \\ f_k \end{bmatrix}\right) = \dfrac{y^+ - y^-}{2\Delta_{f,k}c_k} \end{cases}, \tag{9}$$

where the coefficient sequence $\{\Delta_{i,k}\}$ ($i = 1, \cdots, d$ for d dimensional vector, and in this work $d = 2$) are independent and symmetrically distributed around 0 with finite inverse $E|\Delta_{i,k}^{-1}|$ over all parameter components i and time steps k. A simple and effective way to decide each component of $\{\Delta_{i,k}\}$ is to use symmetric Bernoulli ± 1 distribution with a probability of 0.5 for each outcome of either +1 or –1 [20].

C. Convergence of SPSA-based `FastProf`

The convergence of SPSA-based `FastProf` is important as it affects the quality of the profiling results as well as the efficiency of the profiler. To explore the applicability of SPSA in the profiling optimization problem and investigate its convergence property, we justify the conditions that lead to the convergence in the context of `FastProf`. As pointed out by Spall in [21] (pp. 161), the conditions for convergence can hardly be all checked and verified in practice due to the lack of knowledge on $G(\theta)$. We provide some intuitive arguments based on the problem nature, and some empirical verifications based on the extensive experiments to justify the appropriateness of SPSA in the profiling optimization problem.

According to Theorem 7.1 in [21] (pp. 186), if Conditions B.1$''$ – B.6$''$ hold and θ^* is a unique maximum of $G(\theta)$, then for SPSA, $\hat{\theta}_k$ almost surely converges to θ^* as $k \to \infty$.

The coefficient sequences a_k and c_k we choose and the distribution we follow to generate the simultaneous perturbations $\{\Delta_{i,k}\}$ easily validate Conditions B.1$''$ and B.6$''$ (see Section IV-A and [20] for more details).

The main concern of Conditions B.2$''$ and B.3$''$ is to ensure that $\hat{\theta}_k$ is close enough to θ^* such that $\hat{\theta}_k$ has a tendency to converge to θ^*. These two conditions are valid in our problem scenario because: i) the requirement for Condition B.3$''$, i.e. $\sup_{k \geq 0} \left\|\hat{\theta}_k\right\| < \infty$, can be verified since the control parameter values of block size and buffer size are both finite positive numbers; ii) since the feasible regions of the control parameters of `FastProf` are finite and mapped to a limited range (e.g., $[1.0, 25.0]$ in our test cases in Section V and Section VI) of the iterative variables, $\hat{\theta}_k$ (including the starting point) is sufficiently close to θ^*; iii) θ^* is not a single point but an "acceptable area" including a set of adjacent points (see Figure 3); iv) from a practical point of view, `FastProf` attempts to move $\hat{\theta}_k$ to the nearest point within the feasible space if it deviates from the feasible space, and then adjust the step size accordingly to avoid such situations; and v) our extensive emulations and experiments in Section V and Section VI confirm the closeness and natural tendency even if the starting point is randomly selected.

The way we generate the simultaneous perturbations $\{\Delta_{i,k}\}$ (see Section IV-A) ensures that $\{\Delta_{i,k}\}$ is a mutually independent sequence, which is independent of $\hat{\theta}_0, \hat{\theta}_1, \cdots, \hat{\theta}_k$. The observed noise during data transfer is mainly caused by the dynamics of end-hosts and network segments. Since we measure the *average* throughput in time duration $[0, \Delta T]$ (see Equation 1), which captures both the positive noise and negative noise, the long-term conditional expectation of the observed noise is considered to be zero, i.e. $E[(\xi^{(+)} - \xi^{(-)})|\{\hat{\theta}_0, \hat{\theta}_1, \cdots, \hat{\theta}_k\}, \Delta_k] = 0$. In addition, since $\{\Delta_{i,k}\}$ is generated following the symmetric Bernoulli ± 1 distribution with a probability of 0.5 for each outcome of either +1 or –1, $E|\Delta_{i,k}^{-1}|$ is uniformly bounded. The observations $y(\hat{\theta}_k \pm c_k\Delta_k)$ are also bounded by the link capacity, so the ratio of measurement to perturbation $E\left[\left(\dfrac{y(\hat{\theta}_k \pm c_k\Delta_k)}{\Delta_{i,k}}\right)^2\right]$ is uniformly bounded over i and k. Hence, Condition B.4$''$ holds.

As for B.5$''$, it is theoretically unverifiable whether $G(\theta)$ is three-times continuously differentiable and bounded since $G(\theta)$ is practically unknown. However, the smoothness of $G(\theta)$ can be intuitively verified based on the nature of the problem being studied since the throughput $G(\theta)$ as well as the gradient $g(\theta) \equiv \dfrac{\partial G(\theta)}{\partial \theta}$ are at least bounded by the connection capacity and the finite feasible region of θ.

In addition to the above justification, we also would like to point out that although the throughput performance should have a unique theoretical peak over the feasible control parameter space given a specific snapshot of the status of end-hosts and network environment, it has been observed in our experiments that different runs with identical parameter values may yield different throughputs, which makes the uniqueness of θ^* unverifiable. However, the observed performance $y = G(\theta) + \xi$ indeed shows a peak property over the feasible control parameter region. As shown in Figure 1(a), increasing the block size improves the performance if the buffer is sufficiently large; otherwise, increasing the block size would decrease the performance especially when the block size is approaching the buffer size [24]. In Figure 1(b), it is more clearly shown that larger buffer sizes may not always lead to better performance with a fixed block size, and this trend is consistent for different RTTs but with different turning points.

IV. Implementation of `FastProf`

A. An SPSA-based Profiling Process

We present our SPSA-based profiling process as follows.

1) Select an either fixed or random starting point within the feasible space of block size (l) and buffer size (f).
2) Check the termination conditions (see Section IV-C for details) to continue or terminate the profiling process.
3) Calculate $a_k = \dfrac{a}{(A+k+1)^\alpha}$ and $c_k = \dfrac{c}{(k+1)^\gamma}$, where we set $\alpha = 0.602$, $\gamma = 0.101$, and $A = 0.0$ (or other values much less than the expected/allowed number of iterations). The step sizes a and c are determined empirically based on the size of the entire search space.
4) Generate a pair $\Delta_{\{l,f\}} \in \{+1, -1\}$ of perturbations following the symmetric Bernoulli ± 1 distribution with a probability of 0.5 for each outcome of either +1 or –1.

5) Perform one-time profiling twice to collect two corresponding throughput observations $y^{\pm}$, see Equation 8.
6) Generate the simultaneous perturbation approximation to the unknown gradient $g(\theta_k)$ using Equation 9.
7) Apply the standard stochastic approximation form (Equation 3) to update θ_k to a new value θ_{k+1}, increase k by 1 (i.e. $k = k+1$), and go back to Step 2.

B. Profiling Precision

We set the two elements of the control parameter set used in the stochastic approximation model, denoted as $\theta' = [l', f']^T$, as positive numbers within a reasonably selected range to ensure a comparable magnitude of each parameter. We perform a rounding operation in calculating the actual values of the control parameters l and f in the case of fractional results.

The profiling unit of block size, denoted by μ_l, is defined as one payload size, and the block size l ($l \geq 1$) is defined as a multiplicity of the payload size. The profiling process transfers a data block of $\lambda_l(l') \cdot \mu_l$ bytes each time by calling the `appsend()/apprecv()` functions, which may in turn call the `send()/recv()` API functions of the underlying transport protocol multiple times to completely deliver an entire data block. If a UDP-based protocol such as SABUL [11] or UDT [12] is used, 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. For example, UDT's payload size is UDT data packet size (m) minus UDT header length [11], i.e. the profiling unit of block size μ_l is given by

$$\mu_l = m - 16 = (\text{MTU} - 28) - 16 = \text{MTU} - 44, \tag{10}$$

where UDT header is of 16 bytes, and IP and UDP headers are of 20 and 8 bytes, respectively (hence 28 bytes in total). For example, on our testbed in Section V-A, the jumbo frame is enabled and the MTU is 9,000 bytes, then μ_l is 8,956 bytes.

The profiling unit of buffer size, denoted by μ_f, is decided by the specific profiling precision chosen by the end user in any unit within a feasible profiling range, e.g., 1 Byte, 512 KB, 1.0 MB, 2.0 MB, or others.

Based on the above profiling units, we can calculate the actual values of block size (l) and buffer size (f) for performance observations (i.e. the calculations of y^+ and y^- through one-time profilings) given by

$$\begin{cases} l = round\big(\lambda_l(l') \cdot \mu_l\big) \\ f = round\big(\lambda_f(f') \cdot \mu_f\big) \end{cases}, \tag{11}$$

where λ_l and λ_f 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.

C. Termination Conditions

Many efforts (e.g., [9], [22]) have been devoted to the termination conditions of SA methods since the KWSA algorithm was first proposed [16]. In `FastProf`, we consider the following three simple and practical conditions to guarantee the performance and the termination of a profiling process.

1) Early Termination – the $\mathcal{C}$ rule: The best throughput performance y^* of a given data transfer method over a given network connection is unknown until a complete transport profile is obtained. We define the *performance gain ratio* of a one-time profiling as

$$\rho = \frac{y}{y^*}, \tag{12}$$

where y is the observed throughput of a one-time profiling. Over a dedicated connection, we consider bandwidth B as a known constant since it is reserved in advance through services such as OSCARS [4] and set $y^* = B$. When `FastProf` reaches an operational zone that produces a throughput y with a performance gain ratio no less than $\mathcal{C}$, i.e.

$$\rho = \frac{y}{y^*} = \frac{y}{B} \geq \mathcal{C}, \tag{13}$$

`FastProf` terminates. Note that this condition may or may not be satisfied in a certain one-time profiling.

2) Upper Bound – the $\mathcal{M}$ rule: `FastProf` terminates when the number of one-time profilings exceeds a threshold $\mathcal{M}$, which is typically set as $\mathcal{M} \ll (N_l \cdot N_f)$. If $\mathcal{M} = (N_l \cdot N_f)$, `FastProf` rolls back to the exhaustive search as in TPG [24].

3) Impeded Progress – the $\mathcal{R}$ rule: If the number of consecutive iterations that do not produce any performance improvement compared with the best one observed so far exceeds an upper bound ($\mathcal{R}$), `FastProf` terminates.

V. Emulation-based Performance Evaluation Using Real Experimental Results

We conduct profiling emulations using the profiling data collected on a real-life testbed to gain insights into the behaviors of `FastProf` and also compare `FastProf` with other search algorithms including random walk [17] and Tabu search [10].

A. Data Collection

We build a complete profile (a 2D table) in Figure 3 by exhausting $N_l \cdot N_f = 12{,}825$ combinations of block size and buffer size, whose profiling resolutions are set to be one payload size and 2 MB, respectively. These results are collected by running TPG tests over a 10 Gb/s 380 ms connection between a sender host at Argonne National Laboratory (ANL) and a receiver host at University of Chicago (UChicago). We conduct profiling emulations based on this complete profile.

B. Scaling Functions and Parameter Settings

We use Equation 14 to calculate the actual parameter values ($\theta = [l,\ f]^T$) based on the iterative parameter values ($\theta' = [l',\ f']^T$),

$$\begin{cases} l = round\Big(\lambda_l(l') \cdot \mu_l\Big) = \\ round\left\{\left((l' - l'_{\min}) \cdot \frac{(l_{\max} - l_{\min})}{(l'_{\max} - l'_{\min})} + l_{\min}\right) \cdot \mu_l\right\} \\ f = round\Big(\lambda_f(f') \cdot \mu_f\Big) = \\ round\left\{\left((f' - f'_{\min}) \cdot \frac{(f_{\max} - f_{\min})}{(f'_{\max} - f'_{\min})} + f_{\min}\right) \cdot \mu_f\right\} \end{cases}. \tag{14}$$

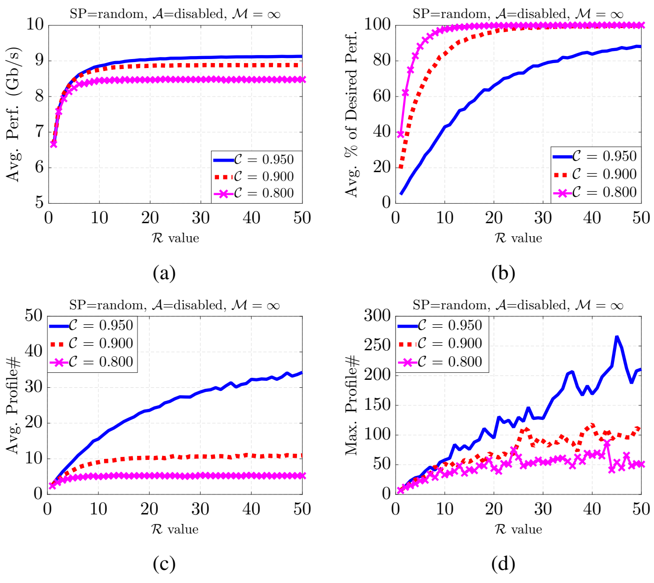


Fig. 4. Overall performance of `FastProf`.

In particular, the profiling range for block size is from 1 payload to 25 payloads (i.e. $l_{\min} = 1$, $l_{\max} = 25$, and $\mu_l = 8956$ bytes), and the profiling range for buffer size is from 1 MB to 1024 MB (i.e. $f_{\min} = 1$, $f_{\max} = 1024$, and $\mu_f = 1{,}048{,}576$ bytes). We set the iterative variables for both block size and buffer size to be from 1.0 to 25.0 (i.e. $l'_{\min} = 1.0$, $l'_{\max} = 25.0$, $f'_{\min} = 1.0$, and $f'_{\max} = 25.0$) to ensure that they are of comparable and consistent magnitudes.

C. Performance Measurements

We consider several different parameters in the profiling emulations: i) the starting point (SP) (either fixed or randomly selected); ii) the $\mathcal{A}$ rule (either disabled or enabled)[1]; iii) the value of $\mathcal{M}$ (either ∞ or $2\mathcal{R}$); iv) the value of $\mathcal{C}$ (selected from $\{0.95, 0.90, 0.80\}$); and v) the value of $\mathcal{R}$ (integers from 1 to 50). We measure four types of performance metrics in 5,000 runs: i) the average throughput; ii) the percentage that leads to a desired performance; iii) the average profiling time as indicated by the average number of one-time profilings; and iv) the longest profiling time as indicated by the maximum number of one-time profilings. The subfigures in each figure in this section correspond to these four types of performance metrics labeled by $\mathcal{A}$, $\mathcal{M}$, $\mathcal{C}$, and starting point (SP), respectively.

1) Overall Performance: We randomly select a starting point within the feasible region of control parameters, disable the $\mathcal{A}$ rule, and plot the results in Figure 4, which shows that `FastProf` is able to find a set of control parameter values with satisfactory performance in a short time.

The actual throughput obtained by `FastProf` is considered as the most important performance metric. Figure 4(a) shows that a satisfactory throughput performance can be achieved as long as a reasonably large $\mathcal{R}$ is specified (e.g., >10 in our test cases). As $\mathcal{R}$ increases, the average throughput performance explored by `FastProf` first increases and then stabilizes at the desired level as specified by the value of $\mathcal{C}$.

We measure the percentage of cases that yield a user-desired performance among all 5,000 runs, as shown in Figure 4(b). As $\mathcal{R}$ increases, the percentage significantly increases up to 100% for $\mathcal{C} \in \{0.80, 0.90\}$. When $\mathcal{C} = 0.95$, the percentage does not reach 100% as $\mathcal{R}$ approaches 50, but we still achieve slightly higher throughput performance near or above 9.0 Gb/s as $\mathcal{R}$ increases, as shown in Figure 4(a).

We measure the profiling speed by calculating the average number of one-time profilings conducted by the profiler among all 5,000 runs. As expected, this average number generally increases as $\mathcal{R}$ increases, as shown in Figure 4(c). For a relatively smaller $\mathcal{C}$ value (e.g., 0.80 and 0.90 in our test cases), it does not always increase as $\mathcal{R}$ increases since the percentage of yielding a desired performance reaches 100% quickly and `FastProf` terminates the profiling process without consuming more profiling time; while for a larger $\mathcal{C}$ value (e.g., 0.95 in our test cases), it takes longer to obtain a higher percentage for a desired performance. As $\mathcal{R}$ increases, there is a higher probability to obtain a desired performance by conducting more one-time profilings, but the actual average throughput performance does not increase as significantly as the percentage of obtaining a desired performance. This observation implies that an appropriate $\mathcal{R}$ is needed: a larger value may lead to a longer profiling process without perceivable performance improvement.

We measure the maximum number of one-time profilings among all 5,000 runs, as shown in Figure 4(d), which reflects the longest profiling time that `FastProf` may take. Since the number of one-time profilings is not limited ($\mathcal{M} = \infty$), the profiling process terminates only when either a desired performance is achieved or the number of consecutive iterations without performance improvement reaches $\mathcal{R}$. As $\mathcal{R}$ increases, there is more space to be explored by `FastProf` for better performance, and the longest profiling time among all runs may either stop increasing (for smaller $\mathcal{C}$) or keep increasing (for larger $\mathcal{C}$).

2) Effects of $\mathcal{M}$: As shown in Figure 4(d), as $\mathcal{R}$ increases, the maximum number of one-time profilings resulted from a larger $\mathcal{C} = 0.95$ is up to 250. Even if a one-time profiling takes only 2 minutes, the profiler would take at most 10 hours to obtain a desired performance with a high probability. In practice, due to the complex system and network dynamics, the profiler may run much longer without getting a desired performance. We avoid this situation by setting the upper bound on the total number of one-time profilings ($\mathcal{M}$) to be a finite value. The results in Figure 5 are based on $\mathcal{C} = 0.95$, the upper bound $\mathcal{M} = 2\mathcal{R}$, and the same other parameters as those in Figure 4. Setting such an upper bound provides a guarantee on the profiling time without significantly affecting the profiling performance. Figure 5(c) shows that the average number of profilings is slightly reduced with $\mathcal{M} = 2\mathcal{R}$, and Figure 5(d) shows the maximal number of one-time profilings

[1]If $\mathcal{A}$ is enabled, `FastProf` only searches the feasible space where the buffer size is larger than Bandwidth Delay Product (BDP), i.e. $f \geq$ BDP; otherwise, it searches the entire feasible space.

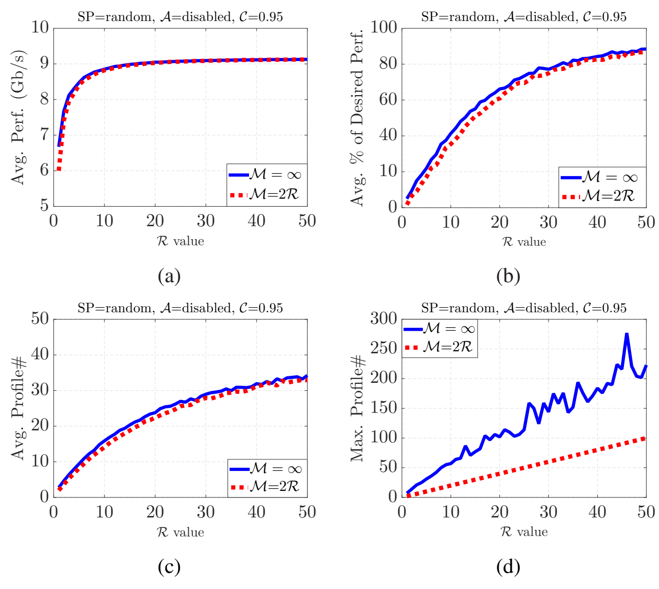


Fig. 5. Effects of $\mathcal{M}$ on profiling performance.

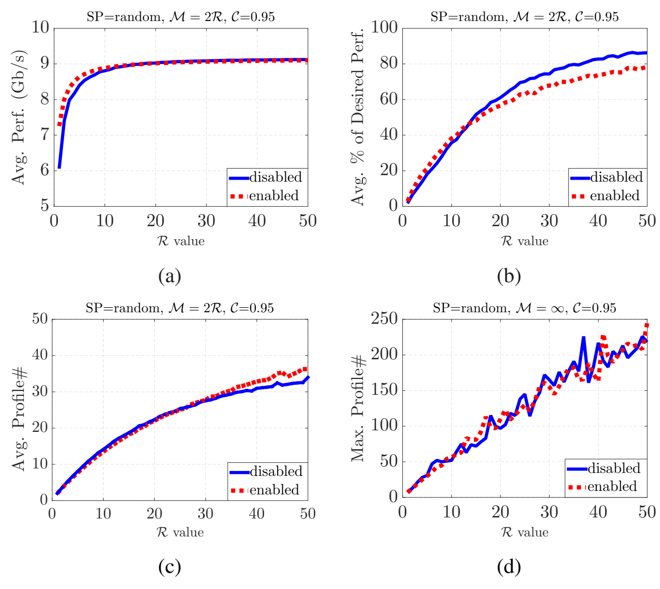


Fig. 6. Effects of $\mathcal{A}$ on profiling performance.

among all 5,000 runs is limited by the finite $\mathcal{M}$. Restricting the total number of one-time profilings does not perceivably affect the average profiling performance, as shown in Figure 5(a). As $\mathcal{R}$ increases, the actual average performance increases and stabilizes between 8.0 Gb/s and 9.5 Gb/s, which is considered to be a satisfactory performance over a 10 Gb/s connection. This observation implies that for one specific run of FastProf, if a desired performance could not be achieved in a reasonable amount of time, simply extending the profiling process may not improve the performance. Therefore, this upper bound $\mathcal{M}$ should be set for FastProf based on the expectation of the tolerable amount of profiling time. In the experiments in Section VI, we set this value to be $\mathcal{M} = 2\mathcal{R}$. Note that since the user-desired performance tends to be achieved before the number of profilings reaches or exceeds $\mathcal{M}$ when $\mathcal{C}$ is relatively small (e.g., 0.80), we choose a relatively larger value $\mathcal{C} = 0.95$ and plot in Figure 5 the effects of $\mathcal{M}$ on the profiling performance. The measurements in Figure 5 suggest that setting $\mathcal{M}$ to be a finite value is more useful and even critical for an aggressive $\mathcal{C}$.

3) *Effects of $\mathcal{A}$:* We plot in Figure 6 the comparisons between the cases where $\mathcal{A}$ is disabled and enabled. Note that when $\mathcal{A}$ is enabled in Figure 6, the profiling range for buffer size is from $f_{\min} = 475$ MB to $f_{\max} = 1{,}024$ MB since a delay of 380 ms over a connection of 10 Gb/s bandwidth yields a bandwidth-delay product of 475 MB. Since the ranges of iterative variables l' and f' are both from 1.0 to 25.0, the buffer size range mapped to the range of iterative variable f' is actually $[475, 1024]$, i.e. $[475, 1024]$ is linearly mapped to $[1.0, 25.0]$. We observe the effects of cutting the search space into nearly half with larger buffer sizes in Figure 6. Firstly, for $\mathcal{R} < 10$, the average performance when $\mathcal{A}$ is enabled is slightly higher than that when $\mathcal{A}$ is disabled, as shown in Figure 6(a). This is because the number of profilings is mainly limited by the value of $\mathcal{R}$ when it is relatively small. The profiling process terminates before FastProf further explores the space for performance improvement, and thus the performance mainly depends on the initial starting point. As $\mathcal{R}$ increases, the differences made by $\mathcal{A}$ become less obvious, and the average throughput performance converges and stabilizes at around 9 Gb/s. Secondly, when $\mathcal{C}$ is a larger value and is hard to achieve (e.g., $\mathcal{C} = 0.95$), as $\mathcal{R}$ increases, enabling $\mathcal{A}$ results in a slightly lower percentage of obtaining a desired performance and a slightly longer profiling time in comparison with the case where $\mathcal{A}$ is disabled (Figure 6(b) and Figure 6(c)). Figure 6(d) shows that enabling $\mathcal{A}$ does not make significant difference on the longest profiling time. Note that in Figure 6(d) we use $\mathcal{M} = \infty$ rather than $\mathcal{M} = 2\mathcal{R}$ to eliminate the effects of $\mathcal{M}$ and show only the effects of $\mathcal{A}$ on the profiling time.

4) *Effects of Starting Point (SP):* To study the effects of the SP, we fix the starting point of the profiling at the "left-bottom" $(l_1, f_1) = (1, 1)$, i.e. one payload size for block size and 1 MB for buffer size (see Equation 14 and Figure 3). We re-run the emulations using the same parameter settings as those in Sections V-C1, V-C2, and V-C3, and compare the results with those with random starting points in Figure 7. We observe that random SP produces: i) either slightly higher (when $\mathcal{R}$ is small) or the same (when $\mathcal{R}$ is large) average performance, as shown in Figure 7(a); ii) a consistently higher average probability (percentage) of obtaining a user-desired performance, as shown in Figure 7(b); iii) a consistently shorter average profiling time, as shown in Figure 7(c); and iv) a more stable status in the worst case, as shown in Figure 7(d).

D. Trace of the Profiling Process

To show the detailed profiling process of FastProf, we set $\mathcal{R} = 35$ and $\mathcal{C} = 0.95$, disable $\mathcal{A}$, and then keep track of each pair of parameter values (l, f) as the profiling process progresses. We plot a trace of control parameter values profiled by

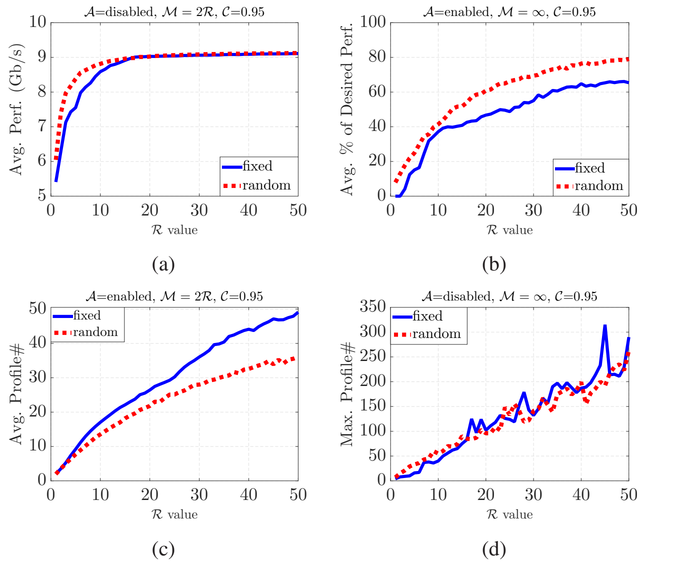


Fig. 7. Effects of starting point (SP) on profiling performance.

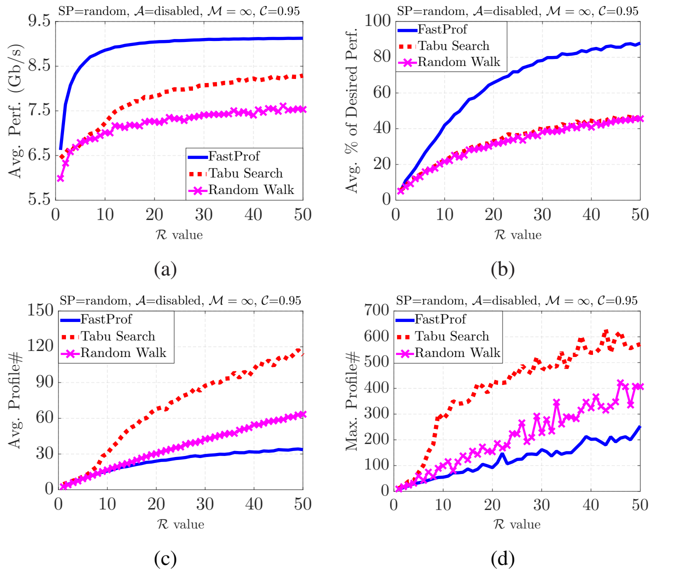


Fig. 9. Comparison of profiling effectiveness and efficiency between `FastProf`, random walk, and Tabu search.

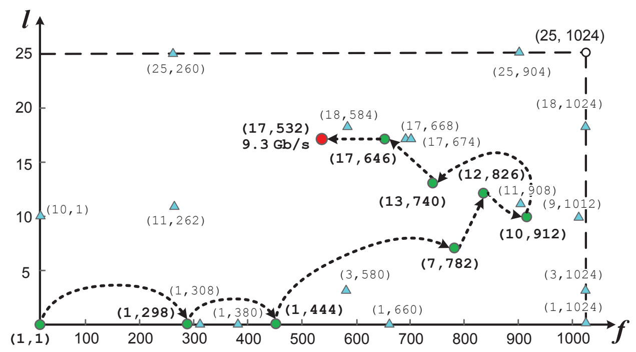


Fig. 8. Profiling process trace with $\mathcal{R} = 35$, $\mathcal{C} = 0.95$, and disabled $\mathcal{A}$.

`FastProf` in Figure 8, where we use (green) circles to indicate the values of $\hat{\theta}_k$, use triangles to indicate the perturbations $\hat{\theta}_k \pm \Delta_k$, i.e. the values used by `FastProf` to measure y^+ and y^-, and use red circles (i.e. the last one on the path) to indicate the control parameter values that produce a desired performance, i.e. the ones in the "acceptable area" (user-desired) as shown in Figure 3. In this tracing experiment, we set $a = 25$ and $c = 9.5$, and fix the starting point at $(1, 1)$. Figure 8 shows that `FastProf` is able to explore a path in the search space from the fixed starting point to an acceptable area.

E. Comparison with Other Search Algorithms

We conduct profiling emulations using two existing heuristics, i.e. random walk [17] and Tabu search [10], and compare their performances with `FastProf`. We measure the same four performance metrics as in Section V-C and plot the results in Figure 9. In comparison with random walk and Tabu search, `FastProf` consistently produces significantly better average performance (Figure 9(a)), has a higher probability to obtain a user-desired performance (Figure 9(b)), takes much less profiling time (Figure 9(c)), and has much better worst cases (Figure 9(d)) among all 5,000 runs.

VI. Experiment-Based Performance Evaluation Over Physical Connections

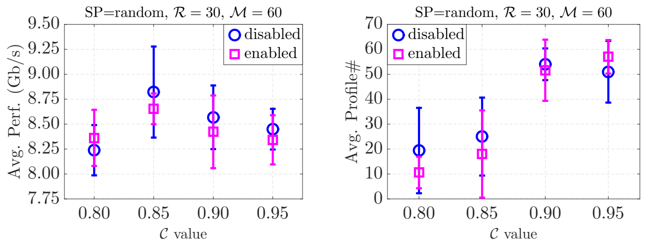
We implement `FastProf` based on TPG and conduct experiments over various physical connections with 2 ms and 380 ms RTTs in real-life network environments.

A. Experimental Results on ANL Testbed

We run `FastProf` over 10 Gb/s physical connections from ANL to UChicago with different delays (2 ms and 380 ms). We set $y^* = 10$ Gb/s (i.e. the connection capacity), $a = 30.0$, $c = 9.5$, $\mathcal{R} = 30$, and $\mathcal{M} = 2\mathcal{R} = 60$, and collect experimental results with profiling precision of 1 Byte[2].

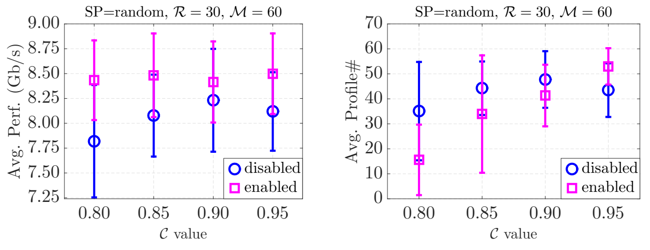
We run each test for 10 times, measure the average performance and profiling time as well as their standard deviations, and plot the results in Figure 10 (2 ms) and Figure 11 (380 ms) for both disabled $\mathcal{A}$ and enabled $\mathcal{A}$ cases. The results show that `FastProf` is able to consistently find a satisfactory set of control parameter values in a short profiling period for both short and long delays. For a short RTT of 2 ms, Figure 10 shows that the throughput explored by `FastProf` is quite stable (>8.0 Gb/s). When $\mathcal{C} \in \{0.80, 0.85\}$, `FastProf` achieves a user-desired performance in all 10 runs with less than 30 one-time profilings on average. When $\mathcal{C} \in \{0.90, 0.95\}$, the desired performance can only be occasionally achieved. Consequently, the early termination rule (i.e. the $\mathcal{C}$ rule) is less frequently triggered and the majority of the runs are terminated by the $\mathcal{R}$ rule or the $\mathcal{M}$ rule. Since `FastProf` attempts to achieve a higher level of performance (due to the larger values of $\mathcal{C}$),

[2]We perform rounding operations to ensure that an integer number of bytes are set for both the block size and the buffer size.



(a) (b)

Fig. 10. Experimental results of `FastProf` on a 10 Gb/s 2 ms connection: (a) average throughput performance; (b) average profiling time.



(a) (b)

Fig. 11. Experimental results of `FastProf` on a 10 Gb/s 380 ms connection: (a) average throughput performance; (b) average profiling time.

it takes a longer time (Figure 10(b)) and the performance is maintained at the same level (Figure 10(a)). For a longer RTT of 380 ms, Figure 11 shows that `FastProf` is able to discover an appropriate set of values for block size and buffer size that result in an average performance between 7.5 Gb/s and 8.5 Gb/s. The performance achieved by `FastProf` is consistent with the complete profile we manually collected for emulations in Section V. We also observe that enabling the $\mathcal{A}$ rule does not necessarily improve the performance, which implies that `FastProf` is not sensitive to the size of the search space.

VII. Conclusion and Future Work

We designed a stochastic approximation-based transport profiler, `FastProf`, to accelerate the profiling process for big data transfer in high-performance networks. We implemented `FastProf` based on the existing Transport Profile Generator, and conducted both extensive profiling emulations in comparison with other search algorithms and profiling experiments on physical connections with short (2 ms) and long (380 ms) delays. The emulation and experimental results showed that `FastProf` significantly reduces the profiling time while achieving a comparable level of end-to-end performance.

We plan to investigate various aspects of the proposed SA-based method such as gradient approximation averaging, step size adaption, and intelligent termination conditions to further improve the profiling performance and accuracy of `FastProf`. Since the end-to-end data transfer is a complex process that involves both network and end-hosts, the parameter selection for each part of the entire process would affect the application level performance observed by end users. Hence, it is also of our interest to explore the possibility of applying such SA-based approaches to storage profiling on end-hosts. `FastProf` makes it possible to conduct "on-line" profiling and we plan to integrate it into existing big data movement solutions such as [23] to improve their time efficiency.

Acknowledgment

This research is sponsored by U.S. National Science Foundation under Grant No. CNS-1560698 and Oak Ridge National Laboratory, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725/4000141963 with New Jersey Institute of Technology. This work was completed in part with resources provided by the University of Chicago Research Computing Center.

References

[1] Energy Sciences Network. `http://www.es.net`.
[2] ESnet iperf3. `https://github.com/esnet/iperf`.
[3] Internet2. `http://www.internet2.edu`.
[4] OSCARS. `http://www.es.net/oscars`.
[5] UDP-based Data Transfer. `http://udt.sourceforge.net/`.
[6] XDD: The eXtreme dd toolset. `https://github.com/bws/xdd`.
[7] V. Ahuja, M. Farrens, and D. Ghosal. Cache-aware Affinitization on Commodity Multicores for High-speed Network Flows. In *Proc. of ANCS '12*, pages 39–48, Austin, TX, Oct. 2012.
[8] J.S. Chase, A.J. Gallatin, and K.G. Yocum. End System Optimizations for High-speed TCP. *IEEE Comm. Mag.*, 39(4):68–74, Apr. 2001.
[9] R.H. Farrell. Bounded Length Confidence Intervals for the Zero of a Regression Function. *Ann. Math. Stat.*, 33(1):237–247, 1962.
[10] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Comput. Oper. Res.*, 13(5):533–549, May 1986.
[11] Y. Gu and R.L. Grossman. SABUL: A Transport Protocol for Grid Computing. *J. of Grid Comput.*, 1(4):377–386, 2003.
[12] Y. Gu and R.L. Grossman. UDT: UDP-based Data Transfer for High-speed Wide Area Networks. *Comput. Netw.*, 51(7):1777–1799, 2007.
[13] Y. Gu, X. Hong, and R.L. Grossman. Experiences in Design and Implementation of a High Performance Transport Protocol. In *Proc. of SC '04*, pages 22–35, Pittsburgh, PA, Nov. 2004.
[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 NDM '14*, pages 9–15, New Orleans, LA, Nov. 2014.
[15] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-Deployed Software Defined WAN. In *Proc. of SIGCOMM '13*, pages 3–14, Hong Kong, China, Aug. 2013.
[16] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of A Regression Function. *Ann. Math. Stat.*, 23(3):462–466, 1952.
[17] G.F. Lawler and V. Limic. *Random Walk: A Modern Introduction*. Cambridge University Press, 2010.
[18] B. H. Leitao. Tuning 10Gb Network Cards on Linux. In *Proc. of the 2009 Linux Symposium*, pages 169–184, Ottawa, Canada, July 2009.
[19] J.C. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Trans. Autom. Control*, 37(3):332–341, March 1992.
[20] J.C. Spall. Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. *IEEE Trans. Aerosp. Electron. Syst.*, 34(3):817–823, July 1998.
[21] J.C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons, Inc., 2003.
[22] T. Wada and Y. Fujisaki. A Stopping Rule for Simultaneous Perturbation Stochastic Approximation. In *Proc. of the 2013 European Control Conference*, pages 644–649, Zurich, July 2013.
[23] D. Yun and C.Q. Wu. An Integrated Transport Solution to Big Data Movement in High-performance Networks. In *Proc. of ICNP '15*, San Francisco, CA, Nov. 2015.
[24] D. Yun, C.Q. Wu, N.S.V. Rao, B.W. Settlemyer, J. Lothian, R. Kettimuthu, and V. Vishwanath. Profiling Transport Performance for Big Data Transfer over Dedicated Channels. In *Proc. of ICNC '15*, pages 858–862, Anaheim, CA, Feb. 2015.