Big Data Transfer Optimization Based on Offline Knowledge Discovery and Adaptive Sampling

MD S Q Zulkar Nine, Kemal Guner, Ziyun Huang, Xiangyu Wang, Jinhui Xu, Tevfik Kosar
Department of Computer Science and Engineering
University at Buffalo (SUNY)
Buffalo, NY, 14260, USA
Email: {mdsqzulk,kemalgne,ziyunhua,xiangyuw,jinhui,tkosar}@buffalo.edu

***Abstract*—The amount of data moved over dedicated and non-dedicated network links increases much faster than the increase in the network capacity, but the current solutions fail to guarantee even the promised achievable transfer throughputs. In this paper, we propose a novel dynamic throughput optimization model based on mathematical modeling with offline knowledge discovery/analysis and adaptive online decision making. In offline analysis, we mine historical transfer logs to perform knowledge discovery about the transfer characteristics. Online phase uses the discovered knowledge from the offline analysis along with real-time investigation of the network condition to optimize the protocol parameters. As real-time investigation is expensive and provides partial knowledge about the current network status, our model uses historical knowledge about the network and data to reduce the real-time investigation overhead while ensuring near optimal throughput for each transfer. Our novel approach is tested over different networks with different datasets and outperformed its closest competitor by 1.7x and the default case by 5x. It also achieved up to 93% accuracy compared with the optimal achievable throughput possible on those networks.**

I. Introduction

Applications in a variety of spaces — scientific, industrial, and personal — now generate more data than ever before. Large scientific experiments, such as high-energy physics simulations [1], [2], climate modeling [3], [4], environmental and coastal hazard prediction [5], [6], genomics [7], [8], and astronomic surveys [9], [10] generate data volumes reaching several Petabytes per year. Data collected from remote sensors and satellites, dynamic data-driven applications, digital libraries and preservations are also producing extremely large datasets for real-time or offline processing [11], [12]. With the emergence of social media, video over IP, and more recently the trend for Internet of Things (IoT), we see a similar trend in the commercial applications as well, and it is estimated that, in 2017, more IP traffic will traverse global networks than all prior "Internet years" combined. The global IP traffic is expected to reach an annual rate of 1.4 zettabytes, which corresponds to nearly 1 billion DVDs of data transfer per day for the entire year [13].

As data becomes more abundant and data resources become more heterogeneous, accessing, sharing and disseminating these data sets become a bigger challenge. Managed file transfer (MFT) services such as Globus [14], PhEDEx [15], Mover.IO [16], and B2SHARE [17] have allowed users to easily move their data, but these services still rely on the users providing specific details to control this process, and they suffer from inefficient utilization of the available network bandwidth and far-from-optimal end-to-end data transfer rates. End-to-end data transfer performance can be significantly improved by tuning the application-layer transfer protocol parameters (such as pipelining, parallelism, and concurrency levels). Sub-optimal choice of these parameters can lead to under-utilization of the network or may introduce link congestion, queuing delays, packet loss, and end-system over-utilization. It is hard for the end users to decide on optimal levels of these parameters statically, since static setting of these parameters might prove sub-optimal due to the dynamic nature of the network which is very common in a shared environment.

In this paper, we propose a novel two-phase dynamic transfer throughput optimization model for big data based on mathematical modeling with offline knowledge discovery/analysis and adaptive online decision making. During the offline analysis phase, we mine historical transfer logs to perform knowledge discovery about the transfer characteristics. During the online phase, we use the discovered knowledge from the offline analysis along with real-time investigation of the network condition to optimize the protocol parameters. As real-time investigation is expensive and provides partial knowledge about the current network status, our model uses historical knowledge about the network and data to reduce the real-time investigation overhead while ensuring near optimal throughput for each transfer. We have tested our network and data agnostic solution over different networks and observed up to 93% accuracy compared with the optimal achievable throughput possible on those networks. Extensive experimentation and comparison with best known existing solutions in this area revealed that our model outperforms existing solutions in terms of accuracy, convergence speed, and achieved end-to-end data transfer throughput.

In summary, the contributions of this paper include: (1) it performs end-to-end big data transfer optimization completely at the application-layer, without any need to chance the existing infrastructure nor to the low-level networking stack; (2) it combines offline knowledge discovery with adaptive real-time sampling to achieve close-to-optimal end-to-end data transfer throughput with very low sampling overhead; (3) it constructs all possible throughput surfaces in the historical transfer logs using cubic spline interpolation, and creates a probabilistic

confidence region with Gaussian distribution to encompass each surface; (4) in real time, it applies adaptive sampling over the pre-computed throughput surfaces to provide faster convergence towards maximally achievable throughput; (5) it outperforms state-of-the-art solutions in this area in terms of accuracy, convergence speed, and achieved throughput.

The rest of the paper is organized as follows: Section II presents the problem formulation; Section III discusses our proposed model; Section IV presents the evaluation of our model; Section V describes the related work in this field; and Section VI concludes the paper with a discussion on the future work.

II. Problem Formulation

Application level data transfer protocol parameters (i.e., concurrency, parallelism, and pipelining) can have different impacts on transfer throughput of files with different sizes and the number of files in the dataset. **Concurrency** (cc) controls the number of server processes which can transfer different files concurrently. It can accelerate the transfer throughput when a large number of files needs to be transferred. **Parallelism** (p) is the number of data connections that each server process can open to transfer the different portions of the same file in parallel. It can be a good option for large or medium files. Therefore, the number of parallel data streams is $(cc \times p)$. **Pipelining** (pp) is useful for small file transfers. It eliminates the delay imposed by the acknowledgment of the previous file before starting the next file transfer. For high latency wide-area networks, this delay might prove highly sub-optimal.

Given a source endpoint e_s and destination endpoint e_d with a link bandwidth b and round trip time rtt; a dataset with a total size f_{all}, average file size f_{avg},and number of files n; and set of protocol parameters $\theta = \{cc, p, pp\}$, the throughput th optimization problem can be defined as:

$$\underset{\{cc,p,pp\}}{\operatorname{argmax}} \int_{t_s}^{t_f} th(e_s, e_d, b, rtt, f_{avg}, n, cc, p, pp, l_{ctd}, l_{ext}) \quad (1)$$

where t_s and t_f are the transfer start and end times respectively. As we are optimizing throughput function in a shared environment, other concurrent transfers can affect the behavior of achievable throughput. We can account the incoming and outgoing transfers happening from the source and destination nodes. Our historical logs contain information of such transfers. We define the load from those contending transfers as l_{ctd}. There might exist other transfers with little-known information. We define the load from those external transfers as l_{ext}.

We have made some assumptions when defining our model, which are expressed below.

Assumption 1: Competing Transfers can achieve aggregate throughput, $T = \sum_{i=1}^{N} th_i$, where N is the number of TCP streams for all competing transfers, and th_i is the throughput of individual transfer i.

Assumption 2: After explaining the effect of known competing transfers, the fluctuation on transfer behavior depends on the intensity of the external load l_{ext}.

Assumption 3: Maximum achievable throughput can be bounded by the end-to-end link bandwidth, disk read speed at the source, or disk write speed at the destination. Given disk read speed v_{read}, disk write speed v_{write}, and the link bandwidth b, the maximally achievable end-to-end throughput th_{max} would be:

$$th_{max} \leqslant \min\{b, v_{write}, v_{read}\} \quad (2)$$

Assumption 4: Our model is for application-level optimization of the network data transfer protocols and it is agnostic of the underlying file systems. Due to the use of concurrency and parallelism, it would provide superior performance when parallel file systems are used at the end nodes. Performance degradation due to hardware misconfiguration, storage access delay, and intermediate network device bottlenecks could limit the achievable throughput. Eliminating such bottlenecks might increase the limit of achievable throughput.

III. Proposed Model

Our model consists of two phases: *(i)* offline knowledge discovery *(ii)* online adaptive sampling. The offline analysis module is an additive model. That means when new logs are generated for a certain period of time, we do not need to combine them with previous logs and perform analysis on the entire log *(old log + new log)* from scratch. Users do not need to perform offline analysis during each transfer. Data transfer logs can be collected for a certain period of time and then the additive offline analysis can be performed on those new logs only. For services like Globus, historical logs can be analyzed by a dedicated server and results can be shared by the users. When a user starts data transfer process, the system initiates online adaptive sampling. Adaptive sampling module queries the results of the offline analysis module which can be answered in constant time. The adaptive sampling guided by offline analysis provides faster convergence towards near-optimal throughput.

A. Offline Analysis

Offline analysis collects useful information from the historical logs so that those information can be used by the online module to converge faster. Offline analysis consists of five phases: *(1)* clustering logs in hierarchy; *(2)* surface construction; *(3)* finding maximal parameter setting; *(4)* accounting for known contending transfers; and *(5)*identifying suitable sampling regions.

1) Clustering Logs: Historical data transfer logs contain information about the verity of transfers performed by the users. Therefore, a natural approach would be cluster the logs based on different matrices. Assuming that we have a historical log, L of n_{log} log entries, We can define our clustering problem as (L, m), where m is the number of target clusters. The clusters of the historical logs are $C = \{C_1, ..., C_m\}$, where $\{n_1, ..., n_m\}$ denote the sizes of the corresponding clusters. We consider a pairwise distance function, $d(x, x')$ where $x, x' \in L$. We have tested clustering algorithm for different pair-wise distance functions. For clustering, we have tested two

well-known approaches: (1) K-means++ [18]; (2) Hierarchical Agglomerative Clustering (HAC) with Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [19]. K-means clustering algorithm suffers from initial centroid selection, and wrong initialization could lead to wrong clustering decision. However, K-means++ provides a theoretical guarantee to find a solution that is $O(log\ m)$ competitive to the optimal K-means solution.

2) Surface Construction: Achievable throughput for a given cluster C_i can be modeled as a polynomial surface which depends on the protocol tuning parameters. We have tried three models to see how accurate those can capture the throughput behavior. The models are: (1) quadratic regression; (2) cubic regression; and (3) piecewise cubic interpolation.

Quadratic regression under-fits the historical log severely. One of the good side of this modeling is that it provides a bitonic surface which is easy to explore.

Cubic regression model also suffers from under-fitting the cluster data from historical logs. One way to resolve this under-fitting problem is by introducing piece-wise polynomials between the data samples with a guarantee of smoothness up to second derivatives.

We model throughput with cubic spline surface interpolation [20]. Before introducing the interpolation method, we should explain the relationship among the parameters briefly. Concurrency and pipelining are responsible for a total number of data streams during the transfer, whereas, pipelining is responsible for removing the delay imposed by small files. Due to their difference in characteristic, we model them separately. At first, we construct a 2-dimension cubic spline interpolation for $th = g(pp)$. Given a group of discrete points in 2-dimension space $\{(pp_i, th_i)\}, i = 0, ..., N$, the cubic spline interpolation is to construct the interpolant $th_i = g(pp_i)$ by using piecewise cubic polynomial $g_i(pp)$ to connect between the consecutive pair of points (pp_i, th_i) and (pp_{i+1}, th_{i+1}). The coefficients of cubic polynomials are constrained to guarantee the smoothness of the reconstructed curve. This is implemented by controlling the second derivatives since each piecewise relaxed cubic polynomial g_i has zero second derivative at the endpoints. Now we can define each cubic polynomial piece as:

$$g_i(pp) = c_{i,0} + c_{i,1}pp + c_{i,2}pp^2 + c_{i,3}pp^3, \forall pp \in [pp_i, pp_{i+1}]. \tag{3}$$

Periodic boundaries can be assumed as $g(pp_{i+1}) = g(pp_i)$. Coefficients $c_{i,j}$, where $j = 1, 2, 3$, of piece-wise polynomial $g_i(pp)$ contains $4(N-1)$ unknowns. We can have:

$$g_i(pp_i) = th_i, \quad i = 1, ..., N \tag{4}$$

Hence, the N continuity constraints of $g(pp)$ are as:

$$g_{i-1}(pp_i) = th_i = g_i(pp_i), \quad i = 2, ..., N. \tag{5}$$

We can get $(N-2)$ constraints from Equation (5) as well. We can impose additional continuity constraints up to second derivatives.

$$\frac{d^2 g_{i-1}}{d^2 pp}(pp_i) = \frac{d^2 g_i}{d^2 pp}(pp_i), \quad i = 2, ..., N \tag{6}$$

We can get $2(N-2)$ constraints from Equation (6). The boundary condition for relaxed spline could be written as:

$$\frac{d^2 g}{d^2 pp}(pp_1) = \frac{d^2 g}{d^2 pp}(pp_n) = 0 \tag{7}$$

So we have $N + (N-2) + 2(N-2) + 2 = 4(N-1)$ constraints in hand. The coefficients can be computed by solving the system of linear equations.

Throughput is also dependent on concurrency and parallelism. The example above can be extended to generate throughput surface with two independent variables - cc and p. Figure 1 shows the resulting throughput surfaces for different kinds of datasets. As we can see that throughput surface of small dataset is more complicated than medium or large files.

Data transfer requests within the same cluster C_i with the same protocol parameter values might have a deviation from one another due to measurement errors and many other network uncertainties such as different packet route in the network layer and minor queuing delay. We define those data points with the same protocol parameter entries as ω. To model this deviation, we have used a Gaussian confidence region around each constructed surface. The probability density function of a Gaussian distribution is:

$$p(\omega; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\omega-\mu)^2}{2\sigma^2}}, \tag{8}$$

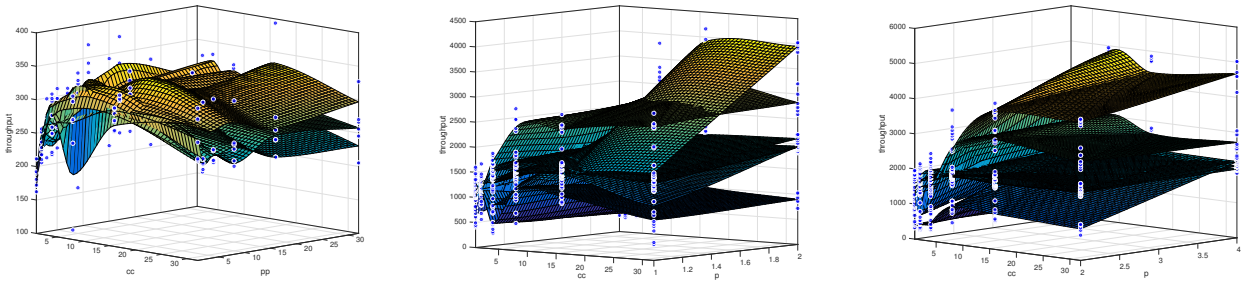
$$\mu = \frac{1}{N}\sum_{i=1}^{N} th_i, \tag{9}$$

$$\delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(th_i - \mu)^2}, \tag{10}$$

where μ is the mean and σ is the standard deviation of the data distribution. Figure 2(a) shows the data model for the Gaussian distribution.

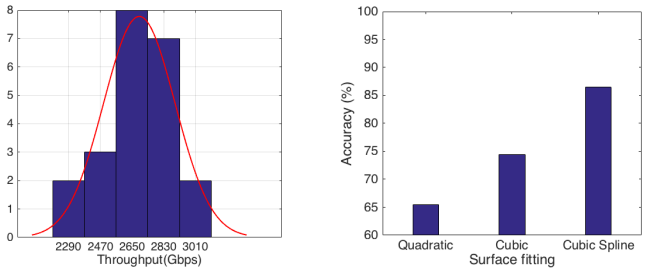
3) Finding Maximal Parameters: Very high protocol parameter values might overburden the system. For this reason, many systems set upper bound on those parameters. Therefore, the parameter search space has a bounded integer domain. Assuming β is the upper bound of the parameters, cubic spline surface functions can be expressed as $f_i : \Psi^3 \Rightarrow \mathbb{R}^+$, where $\Psi = \{1, 2, .., \beta\}$. To find the surface maxima, we need to generate all local maxima of $F = \{f_1, ..., f_p\}$. This is achieved by performing the second partial derivative test on each f_k [20]. The main idea is presented below.

First, we calculate the Hessian matrix of f_k that can be defined as H_k. Then we obtain the coordinates of all local maxima in f_k by calculating the corresponding $\{p, pp, cc\}$'s such that $H_k(p, pp, cc)$ is negative definite. Hence, the set of local maxima of f_k is obtained. Finally, the surface maxima is generated by taking the maximum among all local maxima sets of F.



(a) Small-sized Dataset (b) Medium-sized Dataset (c) Large-sized Dataset

Figure 1: Piece-wise cubic interpolation surface construction.



(a) (b)

Figure 2: (a) Distribution of throughput values under similar external loads; b) Accuracy of different surface construction methods.

4) Accounting for Unknown Contending Transfers: Underlying TCP protocol tries to provide a fair share of bandwidth to all data streams concurrently transferring data. Assume we are analyzing a data transfer log entry t_p, any contending transfer in source or destination can have an impact on transfer request t_p. Known contending transfers are the ones present in the historical log.

In a shared environment, there could be many transfers those are not explicitly logged, however, those unknown transfers could have an impact on the achievable throughput as well. We can define the impact of those uncharted transfers as external load intensity, I_s, and model it with a simple heuristics:

$$I_s = \frac{bw - th_{out}}{bw} \tag{11}$$

5) Identifying Suitable Sampling Regions: Identifying the suitable sampling region is a crucial phase that helps online adaptive sampling module to converge faster. However, not all the regions on a surface are interesting. Many parameter coordinates of a surface are suboptimal. We are interested in regions which have a better possibility of achieving high throughput. The regions containing distinguishable characteristics of the surfaces and containing the local maxima of those surfaces are more compelling. Exploring those regions could lead to a near-optimal solution much faster. Assume the cluster C_i contains η number of the surfaces that can be written as $S = f_1, ...f_\eta$. Now, we can extract the neighborhood with a predefined radius r_d that contains maxima for all the surfaces in S. Assume the set R_m contains all those neighborhood of maxima. We are also interested in regions where surfaces are clearly distinguishable. The goal is to find the regions where surfaces are maximally distant from one another. This problem can be formulated as a max-min problem. Selection can be done by taking the maximum of all pair shortest distance between the surfaces. To achieve that we perform uniform sampling $u = \{u_1, ..., u_\gamma\}$ from surface coordinate (p, cc, pp) for surfaces in S. Therefore, u could be written as:

$$u = \{u_1, ..., u_\gamma\} = \{(p_i, cc_i, pp_i)\}_{i=1}^{\gamma} \tag{12}$$

We define $\Delta_{u_i}^{min}$ as the minimum distance between any two pair of surfaces that can be expressed as:

$$\forall u_k \in u, \quad \Delta_{u_k}^{min} = \min_{\forall i,j \in \{1,...,\eta\}} |f_i(u_k) - f_j(u_k)| \quad \text{where} \quad i \neq j \tag{13}$$

After sorting the list in descending order we choose, λ $(1 < \lambda < k)$, number of the initial samples from the sorted list. Assume the set of points we get after solving the Equation (13) is R_c. We define suitable sampling region as :

$$R_s = R_m \cup R_c \tag{14}$$

During online analysis, we will use the region in R_s to perform the sample transfers.

B. Adaptive Sampling Module

This module is initiated when a user starts a data transfer request. Adaptive sampling is dependent on online measurements of network characteristics. It is essential to assess the dynamic nature of the network that is helpful to find the optimal parameter settings. A sample transfer could be performed to see how much throughput it can achieve. However, a single sample transfer could be error prone and might not provide clear direction towards the optimal solution. Our algorithm adapts as it performs sample transfers by taking guidance from offline surface information. This approach can provide faster convergence. An overview of the module is presented in Algorithm (1). Online module queries the offline analysis module

Algorithm 1: Online Sampling

```
// Source, EP_s, Destination, EP_d, Round trip time, rtt, Bandwidth, bw
input : Data arguments, data_args = {Dataset, avg_file_size, num_files}, network arguments, net_args = {EP_s, EP_d, rtt, bw}, transfer node arguments, node_args = {num_nodes, cores, memory, NIC_speed}
output: Optimal transfer rate, th_opt
1  procedure AdaptiveSampling(F_s, R_s, I_s)
2      D_s ← GetSamples(Dataset)
3      e_{s,median} ← Median(I_s)
4      f_{s,median} ← F_s[e_{s,median}]
5      θ_{s,median}, th_hist ← GetOptimalParam(f_{s,median})
6      th_cur ← DataTransfer(D_{s,1},p_{s,median})
7      Log.append(net_args, D_{s,i},p_{s,median},th_cur)
8      D_s.remove(D_{s,1})
9      for D_{s,i} in D_s do
10         if th_cur ≠ th_hist.confidence_bound then
11             f_{s,cur} ← FindClosestSurface(th_cur)
12             p_{s,cur}, th_hist ← GetOptimalParam(f_{s,curr})
13             th_cur ← DataTransfer(D_{s,i},p_{s,cur})
14             Log.append(net_args, D_{s,i},p_{s,cur},th_cur)
15         end
16     end
17 F_s, R_s, I_s ← QueryDB(data_args,net_args)
18 F'_s ← Sort(F_s,I_s)
   // Set of surfaces, F_s, Sampling region, R_{s,k}, Load intensity, I_s
19 AdaptiveSampling(F'_s, R_{s,k}, I_s)
```

with network and dataset characteristics. Offline module finds the closest cluster and returns the throughput surfaces along with associated external load intensity information and suitable sampling region for each surface.

The online module sorts the surfaces in descending order based on external load intensity value (Lines 17-18). Adaptive sampling module takes the dataset that is needed to be transferred and starts performing sample transfers from the dataset. To perform the first sample transfer, the algorithm chooses the surface with median load intensity, f_{median}, and performs the transfer with:

$$\theta_{s,median} = \{p, cc, pp\} = \text{argmax}(f_{s,median}) \tag{15}$$

which is already precomputed during offline analysis and can be found in the sampling region. Achieved throughput value for the transfer is recorded (Lines 2-6). If the achieved throughput is inside the surface confidence bound at point $\theta_s, median$, then the algorithm continues to transfer rest of the data set chunk by chunk. However, if the achieved throughput is outside the confidence bound, that means the current surface is not representing the external load of the network. If achieved throughput is higher than the surface maxima, that means current network load is lighter than the load associated with the surface. Therefore, the algorithm searches the surfaces with lower load intensity tags and find the closest one and perform second sample transfer with parameters of newly found surface maxima. In this way, the algorithm can get rid of half the surfaces at each transfer. At the point of convergence, our algorithm takes the rest of the dataset and starts the transfer process. Changing parameters in real-time is expensive. For example, if a cc value changes from 2 to 4, this algorithm has to open two more server processes and initialize resources. These new processes have to go through TCP's slow start phase as well. Therefore, the algorithm tries to minimize the initial sampling transfers by the adaptive approach. For very large-scale transfers, when data transfer happens for a long period of time, external traffic could change during the transfer. If the algorithm detects such deviation, it uses most recently achieved throughput value to choose the suitable surface and changes the transfer parameters.

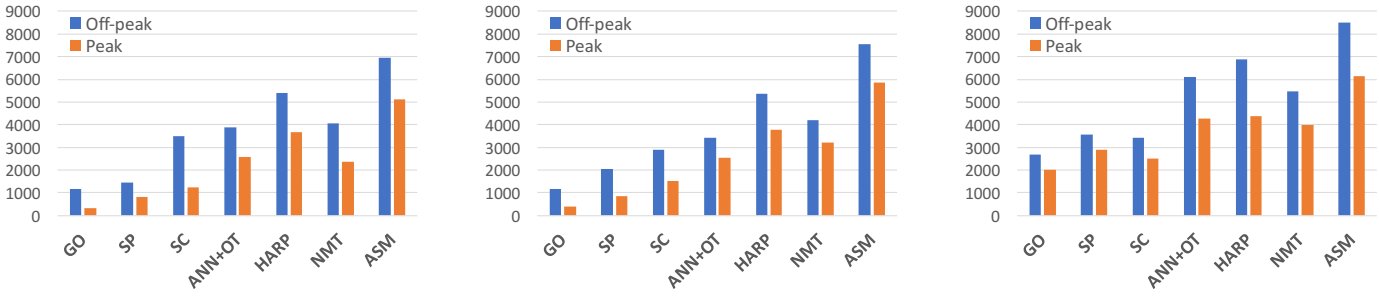
Table I: System specification of our experimental environment

	XSEDE		DIDCLAB	
	Stampede	Gordon	WS-10	Evenstar
Cores	16 per node	16 per node	8	4
Memory	32GB perHost	64GB perHost	10 GB	4 GB
Bandwidth	10 Gbps		1 Gbps	
RTT	40 ms		0.2 ms	
TCP Buffer size	32 MB	32 MB	10 MB	10 MB
Disk Bandwidth	1200 MB/s	1200 MB/s	90 MB/s	90 MB/s

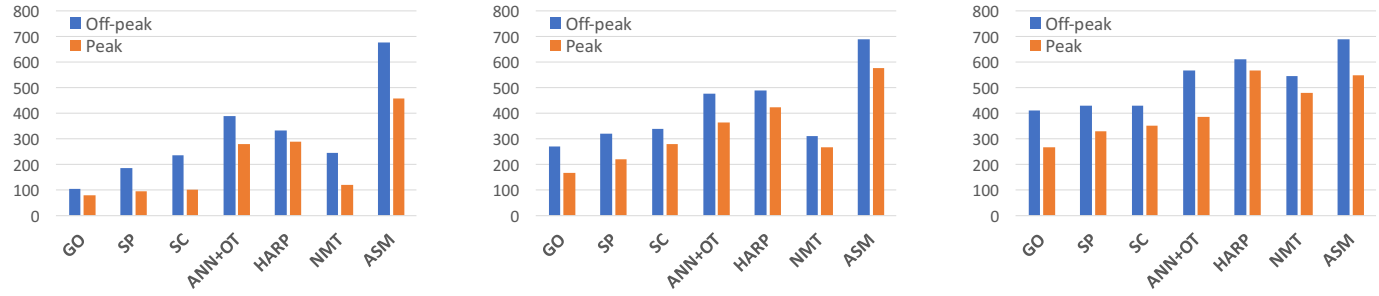
IV. Evaluation

In the evaluation of our model, we used GridFTP [14] data transfer logs generated over a six-week period of time. GridFTP is one of the most widely used data transfer protocols in scientific computing, and it is used to transfer 100s of Petabytes of data every year. As the networking environment, we used XSEDE, a collection of high-performance computing resources connected with high-speed WAN and our DIDCLAB testbed. On XSEDE, we performed data transfers between Stampede at Texas Advanced Computing Center (TACC) and Gordon cluster at San Diego Supercomputing Center (SDSC). Table I shows the system and network specifications of our experimental environment.

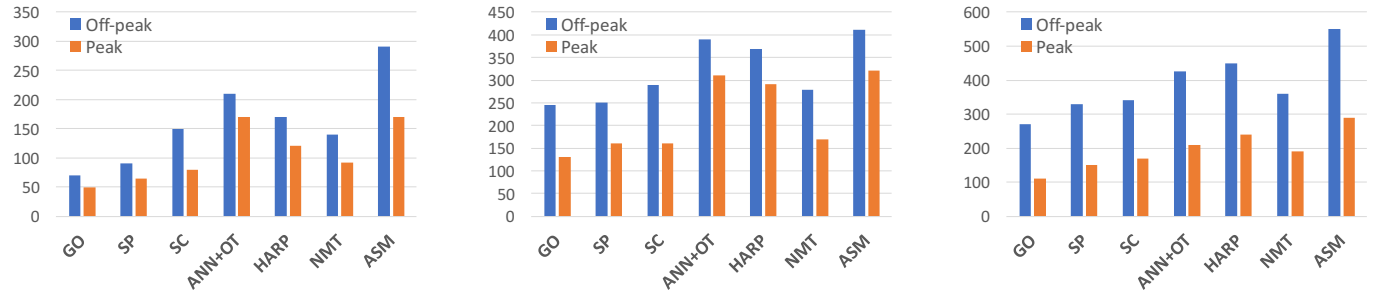
We compared our results with the state-of-the art solutions in this area, such as - (1) Static models: Globus (GO) [21] and Static ANN (SP) [22]; (2) Heuristic models: Single Chunk (SC) [23]; (3) Dynamic models: HARP [24] and ANN+OT [22]; and (4) Mathematical models: Nelder-Mead Tuner (NMT) [25]. Globus uses different static parameter settings for different types of file sizes. SC also makes parameter decision based on dataset characteristics and network matrices. It asks the user to provide an upper limit for concurrency value. SC does not exceed that limit. HARP uses heuristics to perform a sample transfer. Then the model performs online optimization to get suitable parameters and starts transferring the rest of the dataset. Online optimization is expensive and wasteful as it needs to be performed each time, even for similar transfer requests. ANN+OT learns the throughput for each transfer request from the historical logs. When a new transfer request comes, model asks the machine learning module for



(a) Stampede-Gordon SMALL Dataset (b) Stampede-Gordon MEDIUM Dataset (c) Stampede-Gordon LARGE Dataset



(d) WS10-Evenstar SMALL Dataset (e) WS10-Evenstar MEDIUM Dataset (f) WS10-Evenstar LARGE Dataset



(g) Evenstar-Gordon SMALL Dataset (h) Evenstar-Gordon MEDIUM Dataset (i) Evenstar-Gordon LARGE Dataset

Figure 3: Achievable throughput (Gbps) in our experiments performed in various environments for different file sizes.

suitable parameters to perform first sample transfer. Then it uses recent transfer history to model the current load and tune the parameters accordingly. The model only relies on historical data and always tends to choose the local maxima from historical log rather than the global one. Nelder-Mead Tuner implements a direct search optimization which does not consider any historical analysis, rather tries to reach optimal point using reflection and expansion operation. We tested those models three different networks: (1) between two XSEDE nodes; (2) between two DIDCLab nodes; and (3) between DIDCLab and XSEDE nodes.

We tested our model with data transfer requests those are completely different from the historical logs used in the model. To ensure that we computed the list of all unique transfers and split the list as 70% for training the model and 30% for test purpose. We also evaluated our model on both peak and off-peak hours to measure performance under different external load conditions. Achievable throughput is highly dependent on the average file size of the dataset.

In order to evaluate the accuracy of our model for different types of average file sizes, we partitioned transfer requests into three groups - small, medium and large. Then we compared average achievable throughput so that we can evaluate the model in a more fine-grained way. Figure 3 shows the comparison of our proposed Adaptive Sampling Module (ASM) with the other state-of-the-art solutions mentioned above. In all three networks and for all datasets, ASM outperforms all other models. The second best performing model in all of these experiments is HARP [24]. In the XSEDE to XSEDE experiments (Figure 3(a-c)) ASM outperforms HARP by 29% for small datasets, 40% for medium datasets, and 23% for large datasets. Adaptive sampling solves the slow convergence problem with the more accurate pre-constructed representation of throughput surfaces. Our model also gets rid off all the surface regions those proved suboptimal for different background traffic. Moreover, it has a fast online module with adaptive sampling that can converge faster and reduces the suboptimal convergence time. Moreover, our model obtains more impres-

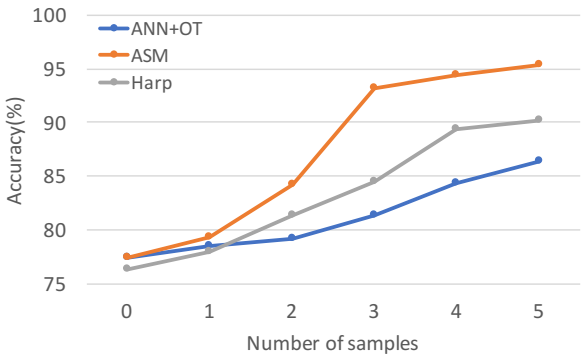


Figure 4: Prediction accuracy of different models with respect to number of sample transfers (those uses online sampling).

sive performance during peak hours. It outperforms HARP by 38%, 55%, and 39% for small, medium, and large datasets respectively. Peak hour periods are challenging to model, and the result shows that our offline analysis is resilient enough to achieve better results in such network environment, with the help of adaptive sampling module.

Figure 3(d-f) shows the performance of different models in our DIDCLAB testbed. Again, our model (ASM) outperforms all the existing models. It achieves 100% performance improvement over HARP during small file transfers during off-peak hours. It outperforms HARP by 41% during medium dataset transfers. However, for large files, the performance improvement is only 13% and during peak hours HARP actually does slightly better than our model. HARP's performance basically depends on its regression accuracy in this case.

In Figure 3 (g-i), we report the performance of these models between DIDCLAB to XSEDE network. This is a quite busy Internet connection which makes it more challenging. In this network too our model performed better than all the mentioned models. For small dataset, our model outperforms its closest competitor ANN+OT by 38%. It outperforms HARP by 22% during large dataset transfers. Our online module needs almost constant time to agree on the parameters. Among the existing models that we have tested so far, only HARP uses the online optimization which could be expensive, however, rest of the models can perform transfers in constant time.

Adaptive Sampling Module(ASM) performs online sampling and uses the network information to query the offline analysis for optimal parameters along with the achievable throughput, $T_{predict}$. The optimal parameters are used for the next sample transfer. Then we measure the actual throughput achieved, $T_{achieved}$. As our model converges $T_{achieved}$ gradually, it gets closer to the $T_predict$. To measure the accuracy of the model we used the following metric:

$$\text{Accuracy} = \frac{|T_{achieved} - T_{predict}|}{T_{predict}} \times 100 \qquad (16)$$

Figure 4 shows a comparison of the accuracy of throughput prediction models. HARP can reach up to 85% with 3 sample transfers along with high online computation overhead. ANN+OT can reach 87.32% accuracy. Our model achieves almost 93% accuracy with three sample transfers for any types of dataset and then it saturates. It shows that our offline cubic spline interpolation can model the network more accurately and adaptive sampling can ensure faster convergence towards the optimal solution.

V. Related Work

Earlier work on application level tuning of transfer parameters mostly proposed static or non-scalable solutions to the problem with some predefined values for some generic cases [14], [26]–[28]. The main problem with such solutions is that they do not consider the dynamic nature of the network links and the background traffic in the intermediate nodes.

Yin et al. [29] proposed a full second order model with at least three real-time sample transfers to find optimal parallelism level. The relationship between parallel streams and throughput along with other parameters are more complex than second order polynomials. Moreover, it does not provide concurrency and pipelining. Yildirim et al. [30] proposed PCP algorithm which clusters the data based on file size and performs sample transfers for each cluster. Sampling overhead could be very high in this model as it does not consider any historical knowledge for optimization.

Engin et al. [24] proposed HARP which uses heuristics to provide initial transfer parameters to collect data about sample transfers. After that model performs the optimization on the fly where it has to perform cosine similarity over the whole dataset which might prove expensive. Even if the optimization and transfer task can be parallelized, it could be wasteful as the same optimization needs to be performed for similar transfers every time a similar transfer request is made.

Prasanna et al. [25] proposed direct search optimization that tune parameters on the fly based on measured throughput for each transferred chunk. However, it is hard to prove the convergence and sometimes hard to predict the rate of convergence. Some cases, it requires 16-20 epochs to converge which could lead to under-utilization. Liu et al. [31] explored Globus historical logs consisting of millions of transfers to analyze the effects of tunable parameters on the transfer characteristics.

Different from the existing work, we address the following issues in this paper: *(i)* Lower order regression model can underfit the data when higher order polynomials can introduce overfitting, in addition, to compute cost and sampling overhead. For small to moderate size of data transfer requests, slow convergence could lead to severe under-utilization. *(ii)* Model free dynamic approaches suffer from convergence issue. And convergence time depends on the location of initial search point. *(iii)* Searching parameters during the transfer could introduce many overheads. Opening a TCP connection in the middle of the transfer introduces a delay due to slow start phase. When initial parameters are far away from optimal solution slow convergence could lead to under-utilization of the network bandwidth which could hurt the overall bandwidth. *(iv)* Optimization based on historical log should not be done

during the transfer, offline analysis can reduce the real-time computing overhead.

VI. Conclusion

In this study, we have explored a novel big data transfer throughput optimization model that relies upon offline mathematical modeling and online adaptive sampling. Existing literature contains different types of throughput optimization models that range from static parameter based systems to dynamic probing based solutions. Our model eliminates online optimization cost by performing the offline analysis which can be done periodically. It also provides accurate modeling of throughput which helps the online phase to reach near optimal solution very quickly. For large scale transfers when external background traffic can change during transfer, our model can detect the harsh changes and can act accordingly. Adaptive sampling module can converge faster than existing solutions. The overall model is resilient to harsh network traffic changes. We performed extensive experimentations and compared our results with best known existing solutions. Our model outperforms its closest competitor by 1.7x and the default case by 5x in terms of the achieved throughput. It also converges faster, and achieves up to 93% accuracy compared with the optimal achievable throughput possible on the tested networks.

As future work, we are planning to increase the achievable throughput further by reducing the impact of TCP slow start phase. Another interesting path is to reduce the overhead introduced by real-time parameter changes. We are also planning to investigate other application-layer protocol parameter sets that can be optimized to achieve even better performance.

Acknowledgments

This project is in part sponsored by the National Science Foundation (NSF) under award number OAC-1724898.

References

[1] CMS, "The US Compact Muon Solenoid Project," https://cms.cern/, 2017.

[2] ATLAS, "A Toroidal LHC ApparatuS Project," https://atlas.cern/, 2017.

[3] J. Kiehl, J. J. Hack, G. B. Bonan, B. A. Boville, D. L. Williamson, and P. J. Rasch, "The national center for atmospheric research community climate model," *J. of Climate*, vol. 11:6, pp. 1131–1149, 1998.

[4] D. R. Easterling, G. A. Meehl, C. Parmesan, S. A. Changnon, T. R. Karl, and L. O. Mearns, "Climate extremes: observations, modeling, and impacts," *science*, vol. 289, no. 5487, pp. 2068–2074, 2000.

[5] R. J. T. Klein, R. J. Nicholls, and F. Thomalla, "Resilience to natural hazards: How useful is this concept?" *Global Environmental Change Part B: Environmental Hazards*, vol. 5, no. 1-2, pp. 35 – 45, 2003.

[6] A. Carrara, F. Guzzetti, M. Cardinali, and P. Reichenbach, "Use of gis technology in the prediction and monitoring of landslide hazard," *Natural hazards*, vol. 20, no. 2-3, pp. 117–135, 1999.

[7] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool," *Journal of Molecular Biology*, vol. 3, no. 215, pp. 403–410, October 1990.

[8] O. Morozova and M. A. Marra, "Applications of next-generation sequencing technologies in functional genomics," *Genomics*, vol. 92, no. 5, pp. 255–264, 2008.

[9] T. J. Loredo, "Analyzing data from astronomical surveys: Issues and directions," in *Statistical Challenges in Modern Astronomy IV*, vol. 371, 2007, p. 121.

[10] D. J. Eisenstein, D. H. Weinberg, E. Agol *et al.*, "Sdss-iii: Massive spectroscopic surveys of the distant universe, the milky way, and extra-solar planetary systems," *The Astronomical Journal*, vol. 142, no. 3, p. 72, 2011.

[11] E. Ceyhan and T. Kosar, "Large scale data management in sensor networking applications," in *In Proceedings of Secure Cyberspace Workshop*, Shreveport, LA, November 2007.

[12] S. Tummala and T. Kosar, "Data management challenges in coastal applications," *Journal of Coastal Research*, vol. special Issue No.50, pp. 1188–1193, 2007.

[13] C. Systems, "Visual networking index: Forecast and methodology, 2015–2020," June 2016.

[14] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, and S. Tuecke, "Software as a service for data scientists," *Communications of the ACM*, vol. 55:2, pp. 81–88, 2012.

[15] R. Egeland, T. Wildish, and C.-H. Huang, "Phedex data service," in *Journal of Physics: Conference Series*, vol. 219, no. 6. IOP Publishing, 2010, p. 062010.

[16] MOVER, https://mover.io/, 2017.

[17] S. B. Ardestani, C. J. Håkansson, E. Laure, I. Livenson, P. Stranák, E. Dima, D. Blommesteijn, and M. van de Sanden, "B2share: An open escience data sharing platform," in *e-Science (e-Science), 2015 IEEE 11th International Conference on*. IEEE, 2015, pp. 448–453.

[18] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[19] I. Gronau and S. Moran, "Optimal implementations of upgma and other common clustering algorithms," *Information Processing Letters*, vol. 104, no. 6, pp. 205–210, 2007.

[20] D. R. Kincaid and E. W. Cheney, *Numerical analysis: mathematics of scientific computing*. American Mathematical Soc., 2009, vol. 3.

[21] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett *et al.*, "Software as a service for data scientists," *Communications of the ACM*, vol. 55, no. 2, pp. 81–88, 2012.

[22] M. S. Q. Z. Nine, K. Guner, and T. Kosar, "Hysteresis-based optimization of data transfer throughput," in *Proceedings of NDM'15*, pp. 5:1–5:9.

[23] E. Arslan, B. Ross, and T. Kosar, "Dynamic protocol tuning algorithms for high performance data transfers," in *Euro-Par 2013 Parallel Processing - 19th International Conference, Aachen, Germany, August 26-30, 2013. Proceedings*, 2013, pp. 725–736.

[24] E. Arslan, K. Guner, and T. Kosar, "Harp: Predictive transfer optimization based on historical analysis and real-time probing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 25:1–25:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014904.3014938

[25] P. Balaprakash, V. Morozov, R. Kettimuthu, K. Kumaran, and I. Foster, "Improving data transfer throughput with direct search optimization," in *2016 45th International Conference on Parallel Processing (ICPP)*, Aug 2016, pp. 248–257.

[26] T. J. Hacker, B. D. Noble, and B. D. Atley, "The end-to-end performance effects of parallel tcp sockets on a lossy wide area network," in *Proceedings of IPDPS '02*. IEEE, April 2002, p. 314.

[27] J. Crowcroft and P. Oechslin, "Differentiated end-to-end internet services using a weighted proportional fair sharing tcp," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 3, pp. 53–69, July 1998.

[28] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante, "Modeling and taming parallel tcp on the wide area network," in *Proceedings of IPDPS '05*. IEEE, April 2005, p. 68.2.

[29] D. Yin, E. Yildirim, and T. Kosar, "A data throughput prediction and optimization service for widely distributed many-task computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22(6), 2011.

[30] E. Yildirim, D. Yin, and T. Kosar, "Balancing tcp buffer vs parallel streams in application level throughput optimization," in *Proc. International Workshop on Data-Aware Distributed Computing (in conjunction with HPDC'09)*, 2009.

[31] Z. Liu, P. Balaprakash, R. Kettimuthu, and I. Foster, "Explaining wide area data transfer performance," in *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '17, 2017, pp. 167–178.