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 network and data agnostic solution is tested over different networks and achieved up to 93% accuracy compared with the optimal achievable throughput possible on those networks.

1. 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 [2,15], climate modeling [17,26], environmental and coastal hazard prediction [13,30], genomics [6,43], and astronomical surveys [20,40] 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 [14,17]. 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 [16].

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 [4], PhEDEx [18], Mover.IO [1], and B2SHARE [7] 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. There is substantial empirical evidence suggesting that performance directly impacts revenue. As two well known examples for this, Google reported 20% revenue loss due to a specific experiment that increased the time to display search results by as little as 500 milliseconds; and Amazon reported a 1% sales decrease for an additional delay of as little as 100 milliseconds [31].

End-to-end data transfer performance can be significantly improved by tuning the application-layer transfer protocol parameters (such as pipelining, parallelism, concurrency, and TCP buffer size). Sub-optimal choice of these parameters can lead to under-utilization of the network or may introduce congestion which would lead to queuing delay, packet loss, end-system over-utilization, extra power consumption, and other factors. 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 end-to-end data transfer throughput optimization model 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. Instead of performing solely mathematical optimization on the fly, we use historical log analysis to construct the throughput surface for different parameters and external loads. We pre-compute the optimal solution for each during the offline analysis phase.

2. We construct all possible throughput surfaces in the historical data using cubic spline interpolation, and create a probabilistic confidence region with Gaussian distribution to encompass each surface.

3. Real production level data transfer logs are used in our experiment and to compute static parameter settings for different types of transfers.

4. In real time, adaptive sampling technique is used over the pre-computed throughput surfaces to provide faster convergence towards maximally achievable throughput.

5. We provide highly accurate prediction of the optimal transfer parameter combinations with minimal sampling overhead.

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.

2. PROBLEM FORMULATION

Application level data transfer parameters can have different impact on 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 [33, 35, 54]. 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 portion of the same file in parallel [11, 19, 24, 25, 39, 41, 45, 51, 53]. 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 [3, 21, 28, 50]. 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 \( b \), round trip time \( rtt \), and a dataset of size \( f_{all} \), average file size \( f_{avg} \), number of files \( n \) and set of protocol parameters \( \theta = \{cc, p, pp\} \), the throughput, \( th \) optimization problem can be defined as:

\[
\arg\max_{\{cc,p,pp\}} \int_{t_s}^{t_f} th(e_s, e_d, b, rtt, f_{avg}, n, cc, p, pp, l_{ctd}, l_{ext})
\]

(1)

Where \( l_{ctd} \) and \( l_{ext} \) are the external load from other transfers. 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 those contending transfers as \( t_{ctd} \). There might exist other transfers with little-known information. We define those transfers as external load \( t_{ext} \).

We have made couple of assumption to define our model. Those are expressed here.

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.

Assumption 2. After explaining away 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 bandwidth, disk read or disk write bottleneck.

Given disk read speed, \( v_{read} \) and disk write speed \( v_{write} \) and the link bandwidth \( b \), the maximally achievable throughput,

Assumption 4. Our model is a network protocol optimization and it is underlying file system agnostic. It can provide superior performance with a parallel file system.

Performance degradation due to hardware misconfiguration, storage access delay, intermediate network devices bottleneck, etc. could limit the achievable throughput. Eliminating such bottlenecks might increase the limit of achievable throughput.
3. PROPOSED MODEL

Our model consists of two phases: (i) Offline knowledge discovery (ii) Online Adaptive sampling. 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 it with previous logs and perform analysis on whole log \((\text{old log} + \text{new log})\). 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 offline analysis can be performed on those new logs. 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, system initiates online adaptive sampling. Adaptive sampling queries the results of offline analysis which can be answered in constant time. Then adaptive sampling guided by offline analysis provides faster convergence towards near optimal throughput.

3.1 Offline Analysis

Offline analysis collects useful information from the historical logs so that those information can be used by online module to converge faster. Offline analysis consist of five phases - (i) Clustering logs in hierarchy, (ii) Surface construction, (iii) Find maximal parameter setting, (iv) Accounting for known contending transfers, (v) Identify suitable sampling regions.

Historical 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 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 pair-wise 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++ \[10\], (2) Hierarchical Agglomerative Clustering (HAC) with Unweighted Pair Group Method with Arithmetic Mean (UPGMA) \[22\]. K-means clustering algorithm suffers from initial centroid selection. Wrong initialization could lead to wrong clustering decision. However, K-means++ provides theoretical guarantee to find a solution that is \(O(\log m)\) competitive to the optimal k-means solution. For HAC, distance between two clusters \(C_i\) and \(C_j\) is defined as \(D(C_i, C_j)\) in Equation \(2\):

\[
D(C_i, C_j) = d(c_i, c_j) = \sqrt{(c_i - c_j)^2}
\]  

(2)

Where \(c_i\) and \(c_j\) are the corresponding cluster centroid of \(C_i\) and \(C_j\). HAC computes proximity matrix with minimum distance \(D(C_i, C_j)\). Then it updates the rows and columns of proximity matrix with new clusters and fills out the matrix with new \(D(C_i, C_j)\). This process is repeated until all clusters are merged into a single cluster.

Clustering accuracy depends on the appropriate number of clusters \(k\). In this work, we have used Calinski and Harabasz index (CH index) to identify the appropriate number of clusters. CH index can be computed as:

\[
CH(m) = \frac{\Phi_{\text{inter}}(m)/(m - 1)}{\Phi_{\text{intra}}(m)/(n - m)}
\]  

(3)

Where \(\Phi_{\text{inter}}\) is the Between-cluster variation and \(\Phi_{\text{intra}}\) is the Within-cluster variation. Both can be defined as sum of Euclidean distance as bellow:

\[
\Phi_{\text{inter}}(m) = \sum_{i=1}^{m} \sum_{x \in c_i} (x - c_i)^2
\]  

(4)

\[
\Phi_{\text{intra}}(m) = \sum_{i=1}^{m} n_k (\bar{C}_k - \bar{x})^2
\]  

(5)

Where \(\bar{C}_k\) is the mean of points in cluster \(k\) and \(\bar{x}\) is the overall mean. Largest \(CH(m)\) score is preferable.

3.1.1 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.

(1) Quadratic regression: A quadratic surface can be represented as:

\[
f(p, cc, pp) = t_1 p^2 + t_2 cc^2 + t_3 pp^2 + ... + t_{k-2} cc + t_{k-1} pp + t_k
\]  

(6)

Where \(t_1, t_2, ..., t_k\) are the parameters of the surface which can be estimated by minimizing the least squared error.

\[
\min_{t_1, ..., t_k} \sum_{i=1}^{n} [f(p_i, cc_i, pp_i) - th_i]^2
\]  

(7)

However, it 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.

(ii) Cubic regression: A multivariate cubic surface can be represented as:

\[
f(p, cc, pp) = t_1 p^3 + t_2 cc^3 + t_3 pp^3 + ... + t_{k-1} pp + t_k
\]  

(8)

Corresponding least squared minimization problem can be defined as:

\[
\min_{t_1, ..., t_k} \sum_{i=1}^{n} [f(p_i, cc_i, pp_i) - th_i]^2 \quad \text{s.t.} \quad f(p, cc, pp) > 0
\]  

(9)
This model also suffers from under-fitting the cluster data from historical logs. One way to resolve this under-fitting problem by introducing piecewise polynomials between the data samples with a guarantee of smoothness up to second derivatives.

(iii) **Piecewise cubic spline interpolation:** We modeled throughput with cubic spline surface interpolation [29]. Before introducing the interpolation method, we should explain the relationship among the parameters briefly. Concurrency and pipelining 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 constructed a 2-dimension cubic spline interpolation for $g(pp) = th$. 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 $g(pp) = th$ 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

$$g_i(pp) = c_i,0 + c_i,1pp + c_i,2pp^2 + c_i,3pp^3, \forall pp \in [pp_i, pp_{i+1}].$$  \hspace{1cm} (10)

Periodic boundaries can be assumed as $g(pp_{N+1}) = g(pp_1)$. Coefficients $c_{i,j}$, where $j = 1, 2, 3$, of piecewise polynomial $g_i(pp)$ contains $4(N-1)$ unknowns. We can have

$$g_i(pp_i) = th_i, \hspace{0.5cm} i = 1, ..., N$$  \hspace{1cm} (11)

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

$$g_{i-1}(pp_i) = th_i = g_i(pp_i), \hspace{0.5cm} i = 2, ..., N. \hspace{1cm} (12)$$

We can get $(N-2)$ constraints from Equation (12) as well. We can impose additional continuity constraints up to second derivatives.

$$\frac{d^2g_{i-1}}{dp^2} (pp_i) = \frac{d^2g_i}{dp^2} (pp_i), \hspace{0.5cm} i = 2, ..., N \hspace{1cm} (13)$$

We can get $2(N-2)$ constraints from Equation (13). The boundary condition for relaxed spline could be written as,

$$\frac{d^2g}{dp^2} (pp_1) = \frac{d^2g}{dp^2} (pp_n) = 0 \hspace{1cm} (14)$$

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. This example is extended to generate surface with two independent variables.

Then we modeled throughput as a piecewise cubic spline surface. It can be done by extending the above 2-dimension cubic interpolation scheme to model the function of throughput. A short overview is given below: We first fix the value of $pp$. The throughput $f(p, pp, cc)$ then becomes $f_{pp}(p, cc)$ which is a surface in
Data points can be represented as \( P = \{ (p_i, cc_j, th_{i,j}) | i = 1, 2, \ldots, N, j = 1, 2, \ldots, M, (x_i, y_j) \in G \} \), where \( G \) is an \( N \times M \) rectangle grid, and each \((p_i, cc_j)\) denotes the grid point at \( i\)-th row and \( j\)-th column. The piecewise cubic interpolation method will construct an interpolated cubic function \( f_{r(i,j)}(p, cc) \) for each rectangle \( r(i,j) \) from the grid \( G \), where \( r(i,j) \) rectangle can be defined by \([p_i, p_{i+1}] \times [cc_j, cc_{j+1}]\).

\( f_{r(i,j)}(p, cc) \) should fit the \( th\)-values of \( P \) at the vertices of \( r(i,j) \), i.e.

\[
\begin{align*}
    f_{r(i,j)}(p_i, cc_j) &= th_{i,j} \\
    f_{r(i,j)}(p_{i+1}, cc_j) &= th_{i+1,j} \\
    f_{r(i,j)}(p_i, cc_{j+1}) &= th_{i,j+1} \\
    f_{r(i,j)}(p_{i+1}, cc_{j+1}) &= th_{i+1,j+1}
\end{align*}
\]

The interpolated cubic functions should also maintain smoothness at grid points. The continuity constraints are computed as an extension of Equation (13) and (14).

After solving the constrains (which is a linear system), all the functions, \( f_{rect(i,j)}(x, y) \), \( i = 1, 2, \ldots, N - 1, j = 1, 2, \ldots, M - 1 \) form a piecewise smooth function which fixes data point set \( P \). Figure 1 shows the constructed piecewise cubic surfaces of throughput for different \( cc \) and \( p \) value. We can see from the graph surfaces for small files are more complex than the medium and large file. Figure 2 shows interpolated curve of throughput for different \( pp \) values. Figure 3(b) shows the accuracy of different surface construction models. It can be seen that piecewise cubic spline outperforms all other models and achieves almost 85% accuracy.

Data transfer requests within the same cluster \( C_i \) with 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, minor queuing delay etc. We define those data points with 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, minor queuing delay etc. We define those data points with 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, minor queuing delay etc.

The probability density function of a Gaussian distribution is:

\[
p(\omega; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(\omega-\mu)^2}{2\sigma^2}}, \quad (15)
\]

\[
\mu = \frac{1}{N} \sum_{i=1}^{N} th_{i,i}, \quad (16)
\]

\[
\delta = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (th_{i,i} - \mu)^2}, \quad (17)
\]

where \( \mu \) is the mean and \( \sigma \) is the standard deviation of the data distribution. Figure 3(a) shows the data model for Gaussian distribution.

### 3.1.2 Find 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 \( \beta \) is the upper bound of the parameters, cubic spline surface functions can be expressed as \( f_i : \Psi^3 \Rightarrow R^+, \) where \( \Psi = \{1, 2, \ldots, \beta\} \). To find the surface maxima, we need to generate all local maxima of \( F = \{ f_1, \ldots, f_p \} \). This is achieved by performing the second partial derivative test on each \( f_k \) [29]. The main idea is shown as below.

First, we calculate the Hessian matrix of \( f_k \) as

\[
H_k = J(\nabla f_k) = \begin{bmatrix}
    \frac{\partial^2 f_k}{\partial p^2} & \frac{\partial^2 f_k}{\partial p \partial cc} & \frac{\partial^2 f_k}{\partial p \partial pp} \\
    \frac{\partial^2 f_k}{\partial pp \partial p} & \frac{\partial^2 f_k}{\partial pp \partial cc} & \frac{\partial^2 f_k}{\partial pp^2}
\end{bmatrix},
\]

where \( J \) stands for the Jacobian matrix

\[
J(F) = \begin{bmatrix}
    \frac{\partial f_1}{\partial p} & \frac{\partial f_1}{\partial cc} & \frac{\partial f_1}{\partial pp} \\
    \frac{\partial f_p}{\partial p} & \frac{\partial f_p}{\partial cc} & \frac{\partial f_p}{\partial pp}
\end{bmatrix}.
\]

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 \( F \).

3.1.3 Accounting for known 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 and can be classified into five types:

(i) **Same source and destination**: Contending transfers with same source and destination as \( t_p \). Figure 4 describes the known contending transfers in \( S \).

(ii) **Source outgoing**: Outgoing transfers from the source node of \( t_p \) with aggregate transfer rate \( r_{out} \). Any contending transfer in same source and destination is given as

\[
\begin{align*}
\text{(iii) Source incoming:} & \quad \text{Incoming transfers to the source} t_{in} \text{with aggregate transfer rate} r_{in} \text{and parameters settings} \, \text{param}_{in}^f. \\
\text{(ii) Destination outgoing:} & \quad \text{Outgoing transfers from the destination node of} t_p \text{with different destination. We defined those transfers as} t_{out} \text{with aggregate transfer rate} r_{out} \text{and parameter settings} \, \text{param}_{out}^f. \\
\text{(iii) Source incoming:} & \quad \text{Incoming transfers to the destination of} t_p \text{named as} t_{in} \text{with aggregate transfer rate} r_{in} \text{and parameters settings} \, \text{param}_{in}^f. \\
\end{align*}
\]

(iii) **Destination incoming**: Outgoing transfers from the destination node of \( t_p \) with different destination. We defined those transfers as \( t_{in}^f \) with aggregate transfer rate \( r_{in} \) and parameter settings \( \text{param}_{in}^f \).

After sorting the list in descending order we choose \( \lambda \) number of initial samples from sorted list. Assume the set of points we get after solving the equation (22) is \( R_c \). We define suitable sampling region as

\[
R_s = R_m \cup R_c
\]

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

3.2 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 with network information, such as bandwidth, RTT, System information and the information about the dataset that user requested to transfer. 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 (Line 17, 18). Then Online module sorts the surfaces in descending order based on external load intensity value. Adaptive sampling module takes the dataset that is needed to be transferred and starts performing sample transfers from the dataset. To perform first sample transfer, the algo-

\[
\begin{align*}
\text{To perform first sample transfer, the algo-}
\end{align*}
\]
Algorithm 1: Online Sampling

// Source, EP<sub>s</sub>, Destination, EP<sub>d</sub>, Round
trip time, rtt, Bandwidth, bw

input : Data arguments, data<sub>args</sub> =
{Dataset, avg_file_size, num_files},
network arguments,
net<sub>args</sub> = {EP<sub>s</sub>, EP<sub>d</sub>, rtt, bw}, transfer
node arguments, node<sub>args</sub> =
{num_nodes, cores, memory, NIC_speed}

output: Optimal transfer rate, th<sub>opt</sub>

procedure AdaptiveSampling(F<sub>s</sub>, R<sub>s</sub>, I<sub>s</sub>)

D<sub>s</sub> ← GetSamples(Dataset)
ε<sub>s,median</sub> ← Median(I<sub>s</sub>)
I<sub>s,median</sub> ← F<sub>s</sub>[ε<sub>s,median</sub>]
θ<sub>s,median</sub>, th<sub>hist</sub> ← GetOptimalParam(I<sub>s,median</sub>)

th<sub>cur</sub> ← DataTransfer(D<sub>s,i</sub>, p<sub>s,median</sub>, th<sub>cur</th>)

D<sub>s</sub>, remove(D<sub>s,i</sub>)

for D<sub>si</sub> in D<sub>s</sub> do

if th<sub>cur</sub> ≠ th<sub>hist</sub>.confidence_bound then

I<sub>s,cur</sub> ← FindClosestSurface(th<sub>cur</th>)
p<sub>s,cur</sub>, th<sub>hist</sub> ← GetOptimalParam(I<sub>s,cur</th>)

th<sub>cur</sub> ← DataTransfer(D<sub>s,i</sub>, p<sub>s,cur</sub>)

Log.append(net<sub>args</sub>, D<sub>s,i</sub>, p<sub>s,cur</sub>, th<sub>cur</th>)

end

end

F<sub>s</sub>, R<sub>s</sub>, I<sub>s</sub> ← QueryDB(data<sub>args</sub>, net<sub>args</sub>)

// Set of surfaces, F<sub>s</sub>, Sampling region,
R<sub>s,k</sub>, Load intensity, I<sub>s</sub>

AdaptiveSampling(F<sub>s</sub>, R<sub>s,k</sub>, I<sub>s</sub>)

end

\[
\theta_{s,median} = \{p, cc, pp\} = \arg\max(I_{s,median}) \tag{24}
\]

which is already precomputed during offline analysis and can be found in sampling region. Achieved throughput value for the transfer is recorded (Line 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 surface maxima, 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, initialize resources. These new processes have to go through TCP 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 algorithm detects such deviation, it uses most recently achieved throughput value to choose the suitable surface and changes the transfer parameters.

4. EXPERIMENTS

In our study, we have used real production level Globus data transfer logs \[4\] to analyze network and throughput behaviors, end system characteristics, peak and off-peak hour transfer loads. For our experiments, 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 1 shows the system and network specifications of our experimental environment.

![Figure 4: Different types of known conflicting transfers.]

| System specification of our experimental environment | XSEDE | DIDCLAB |
|-----------------------------------------------------|-------|---------|
| Cores                                               | 8     | 4       |
| Memory                                              | 10 GB | 4 GB    |
| Bandwidth                                           | 10 Gbps| 1 Gbps  |
| RTT                                                 | 40 ms | 0.2 ms  |
| TCP Buffer size                                     | 48 MB | 10 MB   |
| Disk Bandwidth                                      | 1200 MB| 90 MB   |

We compared our results with many other existing
models, such as - (1) Static models - Globus (GO) [5], Static Parameters (SP) [44], (2) Heuristic Model - Single Chunk (SC) [9], (3) Dynamic model - HARP [8], ANN+OT [14], (4) Mathematical model - Nelder-Mead Tuner (NMT) [12]. 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 requests from the historical log. When a new transfer request comes, model asks the machine learning module for suitable parameters to perform first sample transfer. Then it uses recent transfer history to model current load and tune parameter accordingly. The model only relies on historical data and always tends to choose the 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 real production level networks - (1) Between XSEDE nodes, (2) DDCLab testbed, and (3) Between DIDCLab and XSEDE nodes.

4.1 Transfers between 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 also highly dependent on the average file sizes of the dataset. For example, a dataset with average file size 2 MB and 4 MB achieve two different level of throughputs than dataset with average file size 100MB or 200MB. To see how accurate our model is for different types of average file sizes, we partitioned test data 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 5(a-c) shows the comparison among above mentioned models and our proposed models. For XSEDE transfers, Globus can achieve up to 2700 Mbps throughput for large file transfers during off-peak hours, however, achieved throughputs are significantly lower for the medium and small dataset. Static parameters from [44] can achieve almost 100% increase in throughput compared to Globus for medium files. It also outperforms GO during small and large dataset transfers. It uses the knowledge from historical data to set those parameters. For example, consider two identical transfers are performed over similar external loads. First transfer uses cc = 4 and pp = 4 and the second one uses cc = 8 and pp = 2. Even though in both cases, the total number of TCP streams is 16, the later one achieves significantly higher throughput than the first one. This is because in the later case, eight GridFTP server processes are pushing data through the network instead of four. For small files, higher cc and pp value up to a certain extent work well. However, in a dynamic environment, such strict static settings might lead to under-utilization or very high value could lead to severe packet loss and queuing delay. SC provides around 140% throughput increase for small files compare to Static parameters. The user-provided upper limit for concurrency is set to 10. We can see it performs better than static parameters for all three types of dataset groups. SC uses information like average file sizes, the number of files, network round trip time, TCP Buffer size, network bandwidth etc. All these information help SC to adapt to the network environment. However, for large files, the performance increase is negligible. This is because SC sets concurrency to user defined value 10 which is close to Static parameter concurrency level 8. ANN+OT outperforms SC in every single file category during off-peak and peak hours. However, for small files, the throughput increase is only 11% during the off-peak hour. This is the effect of similar parameters choice of two models. During peak hour, ANN provides almost 38% throughput increase for small files, 65% increase for medium files and 70% increase. That is because, ANN+OT performs online transfers to model current network load, where SC is agnostic towards network traffic. However, HARP outperforms ANN+OT and NMT for all three types of datasets. HARP’s better performance relies on expensive online optimization. ANN+OT excessively dependent on sample maxima. However, for large datasets, the performance of HARP during peak hours is pretty similar to ANN+OT. Even though NMT ultimately reaches to global maxima, it suffers during peak period due to its slow convergence. Each time this algorithm changes the parameters, it has to stop the globus-url-copy command and has to start the command with new parameters, which comes with expensive initialization and TCP slow start phase. Sub-optimal parameter settings during convergence period hurt its overall performance, yet it is free from expensive online optimization. We can observe its degraded performance during the peak hour period. However, it outperforms all the static models along with SC. It actually proves the power of online sampling over the heuristic models. Adaptive Sampling Module (ASM) outper-
forms all other models, even the nearest one (HARP) by almost 40% for medium datasets. For the large dataset, our model outperforms HARP by 23%. In case of the small dataset, the performance increase is 29%. 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 impressive performance during peak hours. It outperforms HARP by almost 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 environments, with the help of adaptive sampling module.

4.2 DIDCLab testbed

Figure 5(d-f) shows the throughput performance of different models in our DIDCLAB testbed. As it is connected with University LAN, peak hour is experienced from 11 am to 3 pm. And night time is mostly the off-peak hour. Globus online better results on the large dataset, however, performs poorly during small and medium transfers. Static Parameters (SP) outper-
forms GO by almost 100% during small file transfers. The performance difference is negligible during large file transfers. In this experimental setting, achievable throughput is actually bounded by disk speed. As single chunk is unaware of disk bottleneck, its parameters become suboptimal and exhibit similar results than SP during peak hours. Because of the historical analysis, ANN+OT can outperform all the static and heuristic models. For small files during peak hour ANN+OT can achieve 3× throughput improvement. On the other hand, during the small dataset transfers, HARP shows performance degradation compare to ANN+OT. After the investigation, we figured out that some cases sample transfer finished during the TCP slow start phase with low throughput, which could mislead the online optimizer and eventually lead to performance degradation. However, HARP outperforms slightly ANN+OT during medium and large dataset transfers. In this environment, direct search shows interesting results. As it is unaware of disk bottleneck, convergence time takes longer than expected. The performance of medium and small dataset hurts the most, as a big portion of the data gets transferred during convergence period. That is why it shows similar results to static models during both peak and off-peak hours. 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, we observed high regression accuracy which leads to better performance. However, the rest of the experiments suggest that in this case, HARP gets "lucky".

4.3 Between DIDCLab and XSEDE

In Figure 5 (g-i), we reported the transfer performance among the models on DIDCLAB to Gordon network. This is over the Internet connection which makes it more challenging. During off-peak period, GO performs better during medium and large data transfers rather than small transfers. Static parameters do better than GO during all the cases, except offline medium dataset transfers. Single chunk outperforms Static Parameters (SP) by 50% during off-peak hours. As expected ANN+OT does better than heuristic and static models. Surprisingly, it performs better than HARP during small and medium file transfers. For medium file transfers its performance is very close to our model. We observed high neural network accuracy for predicting parameters that lead to better performance. This network environment is proved challenging for NMT, as unpredictable peak hour leads to slow convergence. 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.

Among the above-mentioned models, static, heuristics, and mathematical optimization models do not require any historical analysis, however, our model requires extra historical analysis. Therefore, a natural question would be, how often do we have to perform the offline analysis? The answer is, we do not need to perform offline analysis before every single data transfer requests, rather it can be done periodically. Figure 7 shows the impact of offline analysis frequency on the accuracy of the model. Offline analysis performed once a day is enough to reach 92% accuracy. Model accuracy decreases slightly to 87% even for cases, where offline analysis is performed once in 10 days. This shows the model could converge faster, even when offline analysis are performed 10 days apart.

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

\[
\text{Accuracy} = \frac{|T_{\text{achieved}} - T_{\text{predict}}|}{T_{\text{predict}}} \times 100
\]

Figure 6 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.

5. 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 \cite{4,16,23,42,52}. 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. Managed File Transfer (MFT) systems were proposed which used a subset of these parameters in an effort to improve the end-to-end data transfer throughput \cite{4,32,36–38}.

Yin et al. \cite{55} 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. \cite{49}, Kim et al. \cite{27} proposed similar approaches with higher order polynomials.

Yildirim et al. \cite{48} 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. \cite{8} 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. \cite{12} 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 severe under-utilization.

Different from the existing work, we address the following issues in this paper:

1. 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.

2. Model free dynamic approaches suffer from convergence issue. And convergence time depends on the location of initial search point.

3. 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.

4. Optimization based on historical log should not be done during the transfer, offline analysis can reduce the real-time computing overhead.

6. CONCLUSION

In this study, we have explored a novel 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 existing models in terms of accuracy, convergence speed, and achieved throughput. The throughput prediction accuracy of our model reaches 93%.

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.

7. REFERENCES

\begin{enumerate}
\item [1] https://mover.io/.
\item [2] A Toroidal LHC ApparatuS Project (ATLAS). http://atlas.web.cern.ch/.
\item [3] SMTP service extension for command pipelining. http://tools.ietf.org/html/rfc2920, 2015.
\item [4] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, and S. Tuecke.
\end{enumerate}
Software as a service for data scientists. 
*Communications of the ACM*, 55:2:81–88, 2012.

[5] Bryce Allen, John Bresnahan, Lisa Childers, Ian Foster, Gopi Kandaswamy, Raj Kettimuthu, Jack Kordas, Mike Link, Stuart Martin, Karl Pickett, et al. Software as a service for data scientists. *Communications of the ACM*, 55(2):81–88, 2012.

[6] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 3(215):403–410, October 1990.

[7] Sarah Berenji Ardestani, Carl Johan Håkansson, Erwin Laure, Ilja Livenson, Pavel Stranák, Emanuel Dima, Dennis Blommesteijn, and Mark van de Sanden. B2share: An open escience data sharing platform. In *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pages 448–453. IEEE, 2015.

[8] Engin Arslan, Kemal Guner, and Tevfik 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, SC ’16*, pages 25:1–25:12, Piscataway, NJ, USA, 2016. IEEE Press.

[9] Engin Arslan, Brandon Ross, and Tevfik 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*, pages 725–736, 2013.

[10] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[11] H. Balakrishman, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. Tcp behavior of a busy internet server: Analysis and improvements. In *Proceedings of INFOCOM ’98*, pages 252–262. IEEE, March 1998.

[12] 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)*, pages 248–257, Aug 2016.

[13] Antonini Carrara, Fausto Guzzetti, Mauro Cardinali, and Paola Reichenbach. Use of gis technology in the prediction and monitoring of landslide hazard. *Natural hazards*, 20(2-3):117–135, 1999.

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

[15] CMS. The US Compact Muon Solenoid Project. [http://uscms.fnal.gov/](http://uscms.fnal.gov/)

[16] J. Crowcroft and P. Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. *ACM SIGCOMM Computer Communication Review*, 28(3):53–69, July 1998.

[17] David R Easterling, Gerald A Meehl, Camille Parmesan, Stanley A Changnon, Thomas R Karl, and Linda O Mears. Climate extremes: observations, modeling, and impacts. *science*, 289(5487):2068–2074, 2000.

[18] Ricky Egeland, Tony Wildish, and Chih-Hao Huang. Phedex data service. In *Journal of Physics: Conference Series*, volume 219, page 062010. IOP Publishing, 2010.

[19] L. Eggert, J. Heidemann, and J. Touch. Effects of ensemble-tcp. *ACM SIGCOMM Computer Communication Review*, 30(1):15–29, January 2000.

[20] Daniel J Eisenstein, David H Weinberg, Eric Agol, Hiroaki Aihara, Carlos Allende Prieto, Scott F Anderson, James A Arns, Éric Aubourg, Stephen Bailey, Eduardo Balbinot, et al. Sdss-iii: Massive spectroscopic surveys of the distant universe, the milky way, and extra-solar planetary systems. *The Astronomical Journal*, 142(3):72, 2011.

[21] K. Farkas, P. Huang, B. Krishnamurthy, Y. Zhang, and J. Padhye. Impact of tcp variants on http performance. *Proceedings of High Speed Networking*, 2, 2002.

[22] Ilan Gronau and Shlomo Moran. Optimal implementations of upgma and other common clustering algorithms. *Information Processing Letters*, 104(6):205–210, 2007.

[23] 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*, page 314. IEEE, April 2002.

[24] T. J. Hacker, B. D. Noble, and B. D. Atley. Adaptive data block scheduling for parallel streams. In *Proceedings of HPDC ’05*, pages 265–275. ACM/IEEE, July 2005.

[25] R. P. Karrer, J. Park, and J. Kim. Tcp-rome:performance and fairness in parallel downloads for web and real time multimedia streaming applications. In *Technical Report*. Deutsche Telekom Laboratories, September 2006.

[26] J.T. 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: Ccm3. *Journal of
Climate, 11:6:1131–1149, 1998.

[27] J. Kim, E. Yildirim, and T. Kosar. A highly-accurate and low-overhead prediction model for transfer throughput optimization. In Proc. of DISCS Workshop, November 2012.

[28] JangYoung Kim, Esma Yildirim, and Tevfik Kosar. A highly-accurate and low-overhead prediction model for transfer throughput optimization. Cluster Computing, 18(1):41–59, 2015.

[29] David Ronald Kincaid and Elliott Ward Cheney. Numerical analysis: mathematics of scientific computing, volume 3. American Mathematical Soc., 2009.

[30] Richard J. T. Klein, Robert J. Nicholls, and Frank Thomalla. Resilience to natural hazards: How useful is this concept? Global Environmental Change Part B: Environmental Hazards, 5(1-2):35–45, 2003.

[31] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: Listen to your customers not to the hippo. In Proceedings of KDD 2007.

[32] George Kola, Tefik Kosar, Jaime Frey, Miron Livny, Robert Brunner, and Michael Remijan. Disc: A system for distributed data intensive scientific computing, In WORLDS, 2004.

[33] T. Kosar and M. Balman. A new paradigm: Data-aware scheduling in grid computing. Future Generation Computing Systems, 25(4):406–413, 2009.

[34] T. Kosar and M. Livny. Stork: Making data placement a first class citizen in the grid. In Proceedings of ICDCS’04, pages 342–349, March 2004.

[35] Tefik Kosar. Data Placement in Widely Distributed Systems. PhD thesis, University of Wisconsin–Madison, 2005.

[36] Tefik Kosar. Data intensive distributed computing: Challenges and solutions for large-scale information management, 2012.

[37] Tefik Kosar, Engin Arslan, Brandon Ross, and Bing Zhang. Storkcloud: Data transfer scheduling and optimization as a service. In Proceedings of the 4th ACM workshop on Scientific cloud computing, pages 29–36. ACM, 2013.

[38] Tefik Kosar, Mehmet Balman, Esma Yildirim, Sivakumar Kulasekaran, and Brandon Ross. Stork data scheduler: Mitigating the data bottleneck in e-science. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 369(1949):3254–3267, 2011.

[39] J. Lee, D. Gunter, B. Tierney, B. Alcock, J. Bester, J. Bresnahan, and S. Tuecke. Applied techniques for high bandwidth data transfers across wide area networks. In International Conference on Computing in High Energy and Nuclear Physics, April 2001.

[40] Thomas J Loredo. Analyzing data from astronomical surveys: Issues and directions. In Statistical Challenges in Modern Astronomy IV, volume 371, page 121, 2007.

[41] D. Lu, Y. Qiao, and P. A. Dinda. Characterizing and predicting tcp throughput on the wide area network. In Proceedings of ICDCS ’05, pages 414–424. IEEE, June 2005.

[42] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante. Modeling and tampering tcp on the wide area network. In Proceedings of IPDPS ’05, page 68.2. IEEE, April 2005.

[43] Olena Morozova and Marco A Marra. Applications of next-generation sequencing technologies in functional genomics. Genomics, 92(5):255–264, 2008.

[44] MD S. Q. Zulkar Nine, Kemal Guner, and Tevfik Kosar. Hysteresis-based optimization of data transfer throughput. In Proceedings of the Fifth International Workshop on Network-Aware Data Management, NDM ’15, pages 5:1–5:9, New York, NY, USA, 2015. ACM.

[45] H. Sivakumar, S. Bailey, and R. L. Grossman. Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In Proceedings of SC’00 ACM/IEEE conference on Supercomputing, pages 37–es. ACM/IEEE, September 2001.

[46] Cisco Systems. Visual networking index: Forecast and methodology, 2015–2020, June 2016.

[47] S. Tummala and T. Kosar. Data management challenges in coastal applications. Journal of Coastal Research, special Issue No.50:1188–1193, 2007.

[48] 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.

[49] E. Yildirim, D. Yin, and T. Kosar. Prediction of optimal parallelism level in wide area data transfers. IEEE Transactions on Parallel and Distributed Systems, 22(12), 2011.

[50] Esma Yildirim, Engin Arslan, Jangyoung Kim, and Tevfik Kosar. Application-level optimization of big data transfers through pipelining, parallelism and concurrency. IEEE Transactions on Cloud Computing, 4(1):63–75, 2016.

[51] Esma Yildirim, Mehmet Balman, and Tevfik Kosar. Dynamically tuning level of parallelism in wide area data transfers. In Proceedings of the 2008 International Workshop on Data-aware
[52] Esma Yildirim, JangYoung Kim, and Tevfik Kosar. How gridftp pipelining, parallelism and concurrency work: A guide for optimizing large dataset transfers. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion*, pages 506–515. IEEE, 2012.

[53] Esma Yildirim and Tevfik Kosar. Network-aware end-to-end data throughput optimization. In *Proceedings of the first international workshop on Network-aware data management*, pages 21–30. ACM, 2011.

[54] Esma Yildirim and Tevfik Kosar. End-to-end data-flow parallelism for throughput optimization in high-speed networks. *Journal of Grid Computing*, pages 1–24, 2012.

[55] 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*, 22(6), 2011.