

Toward optimizing disk-to-disk transfer on 100G networks

Eun-Sung Jung, Rajkumar Kettimuthu, Venkatram Vishwanath
Mathematics and Computer Science Division
Argonne National Laboratory
Email: {esjung,kettimut,venkatv}@mcs.anl.gov

Abstract—The recent emergence of ultra high-speed networks up to 100 Gbps has posed numerous challenges and has led to many investigations on efficient protocols to saturate 100 Gbps links. However, end-to-end data transfers involve many components, not only protocols, affecting overall transfer performance. These components include a disk I/O subsystem, additional computation associated with data streams, and network adapter capacities. For example, achievable bandwidth by TCP may not be implementable if disk I/O or CPU becomes a bottleneck in end-to-end data transfer. In this paper, we first model all the system components involved in end-to-end data transfer as a graph. We then formulate the problem whose goal is to achieve maximum data transfer throughput using parallel data flows. Our contributions lie in how to optimize data transfers considering all involving system components rather than in accurately modeling involving system components. Our proposed formulations and solutions are evaluated through experiments on the ESnet 100G testbed. The experimental results show that our approach is several times faster than Globus Online – 8x faster for datasets with many 10MB files and 3-4x faster for other datasets of larger size files.

I. INTRODUCTION

Scientific workflows are getting more data-intensive as technology advances in sensors, sequencers, detectors, etc. make abundant data available for analysis. In addition, distributed high-performance computing resources, such as supercomputers, make data movement among geographically distributed sites a major factor that should be taken into account for efficient and reliable scientific workflow management. End-to-end data transfers involve many components affecting the overall transfer performance. Disk-to-disk data transfers start with disk reads, go through data processing and data transmission over network, and end up with disk writes. But the process is not simple. For example, disk reads may involve multiple disks on which data are distributed randomly or with some rules.

The recent emergence of high-speed network up to 100 Gbps has posed considerable challenges and many studies have been conducted on new 100G high-speed networks. In [1], various data transfer middleware such as GridFTP [2] and SRM [3] has been evaluated to determine whether they can

saturate a 100G network link. The results in [1] show that they can achieve 80-90 Gbps in case of memory-to-memory data transfer, where the system’s RAM buffer cache is big enough to hold the entire dataset, so the dataset is loaded into memory before data transfer.

Such performance improvements have resulted from several research areas. First, the attempts to optimize network protocols have brought enhanced network throughput. Globus eXtensible Input/Output System (XIO) [4] provides a framework on which applications do not have to care about what protocols are best for the data transfer over the networks. RDMA-based protocols have been evaluated compared with common protocols such as TCP for high-performance data transfers [5]. The results show that RDMA-based protocols can achieve 10 Gbps data transfer with much lower operating system overheads. Another research area focuses on exploiting multiple flows to achieve high-performance data transfer. For example, GridFTP utilizes pipelining [6] and concurrency [7], to offset protocol overhead for small files.

However, because of the lack of a holistic approach to end-to-end data transfer, achieving high-performance data transfer is difficult in varying hardware and software environments. End systems are becoming more and more complex and heterogeneous. System hierarchy is becoming deep and complex with multi-dimensional topologies. Applications must be smart enough to take advantage of parallelism in various subsystems. So far, manual hardware and software tuning have been needed in order to figure out what configurations are to be set to meet the required data transfer rate. In this paper, we address this problem by modeling system components involved in data transfer and solving mathematically formulated problems.

In this paper, we focus on optimizing parallel flows and CPU loads in end-to-end data transfers. We show how the throughput for datasets with many files can be improved through optimizing the number of parallel flows under constraints of CPU, disk I/O, and so on. For many applications, the individual file sizes in the data set are still small with respect to increasing bandwidth-delay products even though the total volume of the datasets have increased significantly in the past decade. For large files, the approach of splitting a file into multiple chunks and transferring the chunks simultaneously improves the performance. However, the same approach does not work with small files, or even hurt the performance. In this paper, we show that our approach improves the performance significantly by optimizing parallel data flows.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

The remainder of this paper is organized as follows. In Section II, we describe the system modeling for the optimization problem formulations and we formulate the problem. In Section III, we evaluate proposed formulations and solutions through extensive experiments. In Section IV, we summarize our work and briefly discuss future work.

II. END-TO-END TRANSFER OPTIMIZATION

In this section, we describe how to model system components relevant to end-to-end data transfer and we formulate the problem mathematically based on models.

A. System Modeling

Our target system is composed of two clusters of hosts. Two clusters are connected through networks consisting of multiple domains. Any host in one cluster can send data to any host in the other cluster over the networks. One host has multiple CPU cores and is connected to multiple disks. In addition, more than one network interface controller (NIC) may exist for achieving higher bandwidth or utilizing different network paths.

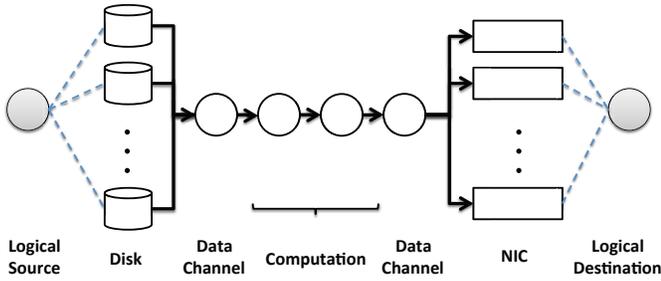


Fig. 1: Data flow graph model

One host can be modeled as a graph as shown in Figure 1. In the graph, there are five classes of nodes, and edges that link adjacent nodes. The five classes of nodes are disk node, data channel node, computation node, NIC node, and logical node. A node is not associated with any attribute, but an edge is associated with attributes describing a node's characteristics. Data channel nodes reflect contention among data flows. For example, if all disks are connected to only one disk interface adapter, maximum disk throughput may not scale linearly as the number of disks increases due to data contention. Logical nodes are inserted for explicit data flow start and end in a graph model. The CPU cores are not expressed explicitly as a node but are put implicitly as costs on edges and constraints in the resulting formulations.

Two attributes are assigned on an edge. One is *capacity/bandwidth* of a source node. The other is *cost* of a data flow on the edge. Both attributes can be either a constant value or a function of some parameters originating from underlying system behaviors. Depending on two end nodes linked by an edge, the edge has different attributes. First, the edge linking from a logical start node to a disk node, *logical edge*, is a logical link with unlimited bandwidth and zero cost function. Second, the edge linking from a disk or data channel node to any other node, *disk edge*, represents a disk I/O path from a disk or data channel. Third, the edge linking from a

computation node to another computation node or a NIC node, *compute edge*, represents a data flow going through computations such as GridFTP and compression computation. Fourth, the edge linking from a NIC node to a logical end node, *network edge*, represents a network path from a source node to a destination node. Each edge is associated with a bandwidth function and a cost function. A bandwidth function and a cost function of an edge describe the performance throughput and CPU resource consumption of a source node, respectively. In the following subsections, we describe each edge's attributes and associated modeling in details.

1) Disk modeling: Disk edge

A disk edge is associated with disk capacity/bandwidth and CPU load related to disk I/O operations. Even though many parameters such as disk cache size are involved in disk I/O bandwidth, the number of data flows per disk is the most important variable assuming that other parameters are fixed and not adjustable.

Equation (1) computes utilization of a disk as a function of number of processes and disk access probability [8]. Here p is a ratio of request data size and the stripe size of a RAID disk. If we assume that file size or request data is bigger than the stripe size of a disk, p can be substituted by 1. The resulting equation is $U \simeq \frac{1}{1+\frac{\gamma}{L}}$, which means the disk utilization increases to some extent as the number of processes increases. γ is a constant to take into account other factors in disk performance such as block size and disk cache.

$$U \simeq \frac{1}{1+\frac{\gamma}{L}(\frac{1}{p}-1+\gamma)}$$

U : Utilization
 L : Number of processes issuing requests
 p : Probability that a request will access a given disk
 γ : Empirically calibrated value

(1)

The disk throughput can be determined by Equation (2) in which the disk utilization in Equation 1 is multiplied by $\frac{N \cdot SU}{E(S)}$. The equation can be rearranged as Equation (3) after substituting $\frac{N \cdot SU}{E(S)}$ by $\frac{\alpha}{L}$, where $\alpha = N \cdot SU$, since $E(S)$, the expected service time of a given disk request, is proportional to L . We can determine α and γ in Equation (3) through experimental values. The cost function for the edge is assumed to be zero since CPUs sit idle until a read/write operation is done and any file operation overhead can be attributed to computation nodes.

$$T = \frac{U \cdot N \cdot SU}{E(S)}$$

T : Throughput
 U : Utilization
 N : Number of disks in a RAID disk
 SU : Stripe size
 S : Service time of a given disk request

(2)

$$T = \frac{1}{1+\frac{\gamma}{L}} \cdot \frac{\alpha}{L} = \frac{\alpha}{L+\gamma}$$

α, γ : Empirically calibrated value

(3)

If the source node is a data channel node, the disk edge is associated with the above bandwidth function when the data channel node has fan-in disk nodes, or the disk edge is associated with infinite bandwidth when the data channel node has

fan-out nodes. Equation (3) will be used as bandwidth function $B_{lk}^d()$ in Section II-B and approximated by a linear/quadratic function for linear programming solver such as cplex [9].

2) Computation modeling: Compute edge

The edge whose source node is a computation node has attributes of linear bandwidth and cost functions. The bandwidth function is a function of the number of flows as in Equation (4), and the cost function can be defined as in Equation (5).

$$T = \alpha n_s + \beta$$

n_s : Number of parallel data transfer streams
 α, β : Empirically calibrated value

(4)

$$C(r) = \alpha r$$

C : CPU load
 r : Data flow rate
 α : Empirically calibrated value

(5)

Equation (4) and (5) will be used as bandwidth function $B_{lk}^c()$ and cost function $C_{lk}^c()$, respectively, in Section II-B.

3) Network modeling: Network edge

The edge linking a NIC node and a logical destination node has attributes of a throughput function and a cost function. In order to simplify the problem, only TCP is considered and a NIC is assumed to have a preassigned protocol property associated with corresponding throughput function.

Several throughput models for parallel TCP streams have been proposed to predict the performance. The simplest model is proposed in [10] and given by Equation (6).

$$T \leq \min\left\{NC, \frac{MSS \times c}{RTT} \cdot \frac{n_t}{\sqrt{p}}\right\}$$

T : Achievable throughput
 NC : Capacity of NIC
 MSS : Maximum segment size
 RTT : Round trip time
 p : Packet loss rate
 n_t : Number of parallel data transfer streams

(6)

If $\frac{MSS \times c}{RTT} \cdot \frac{1}{\sqrt{p}}$ is regarded as a constant, Equation (6) can be rearranged as $T \leq \min\{NC, \alpha \cdot n_t\}$ where $\alpha = \frac{MSS \times c}{RTT} \cdot \frac{1}{\sqrt{p}}$.

The cost function for TCP is given by Equation (7).

$$C(r) = \alpha r$$

C : CPU load
 r : Data flow rate
 α : Empirically calibrated value

(7)

Equation (6) and (7) will be used as bandwidth function $B_{lk}^n()$ and cost function $C_{lk}^n()$, respectively, in Section II-B.

B. Problem Formulation

In order to simplify the problem, we assume that a sender and a receiver have similar hardware such that optimization on the sender side can be applied to the receiver for end-to-end data transfer optimization.

The overall problem-solving procedure is as follows.

- Compute parameters of capacity functions based on empirical data.
- Formulate the modified multicommodity flow problem based on the capacity/cost functions on edges.
- Find a solution including the number of parallel flows, the number of required CPUs, and the number of NICs using linear programming solver, cplex [9].

The graph model as in Figure 2b can be formally represented by a graph $G = (V, E)$, where V is a set of vertices, and E is a set of edges.

Objective

$$\text{maximize } T$$
(8)

Subject to:

$$r_{lk} \geq 0, (l, k) \in E$$
(9)

$$0 \leq n_s \leq M_s, M_s \text{ is maximum number of data streams.}$$
(10)

$$r_{lk} \leq \begin{cases} B_{lk}^d(n_s), (l, k) \in E, & \text{if } V_l \text{ is a disk node} \\ B_{lk}^c(n_s), (l, k) \in E, & \text{if } V_l \text{ is a computation node} \\ B_{lk}^n(n_t), (l, k) \in E, & \text{if } V_l \text{ is a NIC node} \end{cases}$$
(11)

$$\sum_{k:(l,k) \in E} r_{lk} - c \sum_{k:(k,l) \in E} r_{kl} = 0,$$

$$l \neq s_j, l \neq d_j, c = \begin{cases} \text{compression ratio,} & \text{if } l \text{ is compression node} \\ 1, & \text{otherwise} \end{cases}$$
(12)

$$\sum_{k:(s,k) \in E} r_{sk} - \sum_{k:(k,s) \in E} r_{ks} \geq T$$
(13)

$$\sum_{k:(k,d) \in E} r_{ks} - \sum_{k:(d,k) \in E} r_{dk} \geq T$$
(14)

$$\sum_{k:(l,k) \in E} C(r_{lk}) \leq \min(N_c \times 100, n_s \times N_d \times 100)$$
(15)

Fig. 3: Multiple flow determination algorithm

Table I gives a list of notations for mathematical formulations, and the complete formulation is described in Figure 3. The formulation in Figure 3 is mixed-integer convex programming (MICP) since n_s , the number of data streams, is integer and bandwidth function B_{lk}^d is approximated by quadratic functions. The objective function is given in Expression (8), which is to maximize overall throughput of data transfer. Expression (10) set the range of the number of data streams per disk. Expression (11) is a bandwidth/capacity constraint on the edges, where r_{lk} denotes data rate on an edge (l, k) and bandwidth functions is chosen depending on the edge type. The flow conservation constraint given by Equation (12) ensures that the sum of incoming data rates should be same as that of outgoing data rates at every node. In special cases such as compression computation, the sum of outgoing data rates can be a fraction of that of incoming data rates. Expression (13) and (14) constrain the total outgoing data rates from the logical

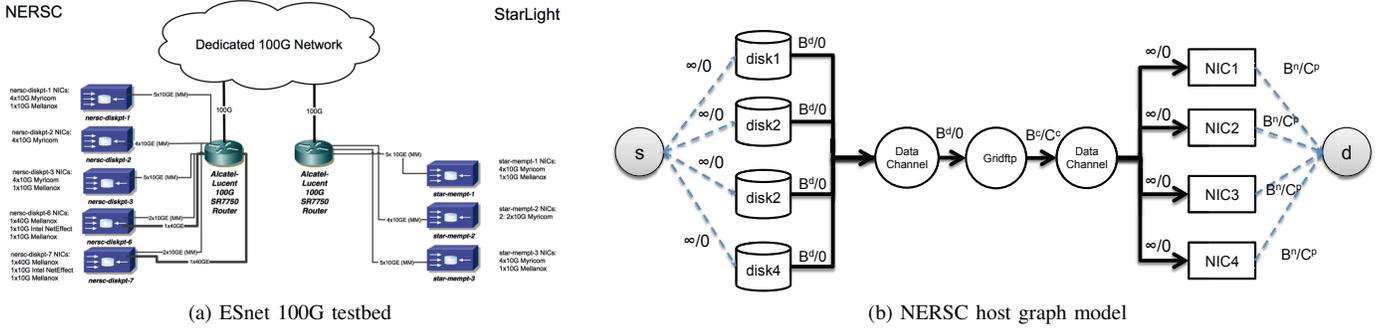


Fig. 2: ESnet 100G testbed and a corresponding graph model

TABLE I: Notation for problem formulation

Notation	Description
V_s	Logical source node
V_d	Logical destination node
N_d	Number of disks
N_c	Number of CPU cores
n_s	Number of data streams per each disk; integer variable
n_{lk}^t	Number of parallel TCP streams on an edge (l, k)
r_{lk}	Data rate on an edge (l, k)
$B_{lk}^d(n_s)$	Disk capacity/bandwidth of V_l , a disk node, associated with an edge (l, k)
$B_{lk}^c(n_s)$	Computation capacity of V_l , a computation node, associated with an edge (l, k)
$B_{lk}^p(n_t)$	Maximum network capacity/bandwidth of V_l , a NIC node, associated with an edge (l, k)
$C_{lk}^c(r_{lk})$	CPU/Computation cost of V_l , a computation node, associated with an edge (l, k)
$C_{lk}^p(r_{lk})$	CPU/Computation cost related to network protocol on V_l , a NIC node, associated with an edge (l, k)

source and to the logical destination node to be greater than or equal to T , which is to be maximized. In this way, we can get the solution that maximizes the overall data throughput. The computation constraints by the number of CPU cores in the system and the number of data flows is given by Expression (15). Note that the formulation assume a circumstance where the number of data flows per disk is same, but the formulation can be easily extended to reflect different number of data flows per disk by assigning separate variables per disk.

III. EXPERIMENTAL EVALUATION

We have conducted experiments on an ESnet 100G testbed in two locations: NERSC (Oakland, CA) and StarLight (Chicago, IL). Figure 2a shows the detailed configuration of the testbed. At NERSC, there are 5 hosts of three different hardware configurations. Three hosts, nersc-diskpt-1, nersc-diskpt-2, and nersc-diskpt-3, have Intel Xeon Nehalem E5650 (2 x 6 = 12 cores), multiple 10G NICs, and 4 RAID 0 sets of 4 drives. Other two hosts do not have RAID drives but have only a local disk. On the other hand, there are 3 hosts without disk arrays at StarLight. These hosts have 2 AMD 6140 (2 x 8 = 16 cores) and multiple 10G NICs, but do not have RAID disks. The hosts at StarLight have only local disks, which are slow (i.e. 300MB/s) and thus can not saturate even a 10G link. For such reasons, we conducted disk-to-memory tests where all data flows departing from hosts at NERSC are directed to `/dev/null` on hosts at StarLight so that we can assume that the hosts at StarLight has same disks as those of the hosts at NERSC.

We have chosen various size of datasets including lots of small files (LOSF) dataset for evaluation of optimizing the end-to-end data transfer rates. We use four different datasets

– ten thousands of 1 MB files, one thousand 10 MB files, one hundred 100 MB files, and ten 1 GB files such that total amount of each dataset would be around 10 GB. The files were synthetically generated using `/dev/urandom` in Linux.

To measure the disk performance, we use `dd` and `iozone` as disk I/O benchmark tools. In addition, we use `nmon` and `netperf` as benchmark tools to measure CPU load and network performance, respectively.

A. Subsystem Tests for Model Parameter Setting

We first conducted basic disk I/O performance tests using `dd` disk utility to obtain baseline performance of disk throughputs. Figure 4 shows the disk read throughputs ($\sim 500\text{MB/s}$) of 4 RAID sets attached to hosts at NERSC. The theoretical upper limits of each RAID disk is around 1.2 GB/s since the RAID disk is composed of four disks with 300 MB/s read performance. Even though there are performance variances among disks, we ignore the variances for simplicity in this paper.

Next, we measured the multithread disk read performance depending on file size and the number of threads to determine the value of α, γ in Equation (3). Figure 5 shows that disk throughput decreases as the number of streams increases regardless of applications' read unit sizes (i.e., 1MB and 10MB). We conducted experiments in case of sequential disk read. Equation (3) determined by these results would be $B_{lk}^d(\cdot)$ in Figure 3 where n_s equals L , and l is a disk node.

However, as Figure 6 shows, the aggregate disk throughput using multiple disks does not scale linearly due to channel contention. We model the channel contention using a data channel node and associated bandwidth function as in Equation (3).

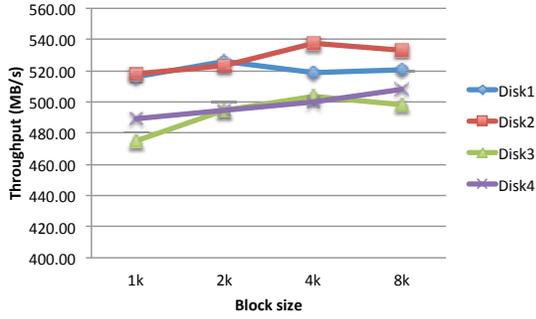


Fig. 4: Disk throughput at NERSC using *dd*: similar disk throughput with varying block size

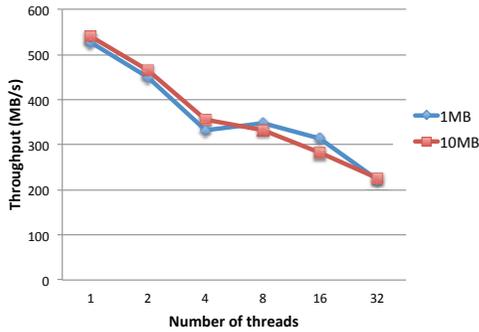


Fig. 5: Disk throughput at NERSC using *iozone*: decreasing disk throughput with increasing number of data streams

We measured the application (i.e., GridFTP) throughput while varying the number of data streams as in Figure 7. We can model GridFTP throughput through Equation (4) by ignoring the decreasing throughput after hitting the peak because that is due to disk bottleneck which is already modeled by *disk edges*. Even though the data movement tool GridFTP is the only application used for the end-to-end data transfers in this paper, we can model any applications such as compression in a similar way through Equation (4) and (5).

Regarding network edges, Figure 8a and 8b show the throughput and CPU load of TCP protocol, respectively. These TCP performance results are measured by *netperf*, and memory-to-memory transfer over a 10G NIC. Figure 8a shows that network transfer throughput is saturated with 3 TCP streams and is near the full capacity of the 10G NIC. Obviously, these protocol behaviors can be modeled by Equation (6) and (7).

B. Results and Discussion

We have compared our model-based optimization approach with two cases: (1) GridFTP with only *-fast option*, (2) GridFTP with auto-tuning optimizations currently used by Globus Online [11]. Globus Online’s auto-tuning algorithm uses different GridFTP optimization options depending on file size. If the number of files is more than 100 and an average file size smaller than 50 MB, it uses GridFTP with concurrency=2 files, parallelism=2 sockets per file, and pipelining=20 requests outstanding at once. If file size is larger than 250 MB, Globus

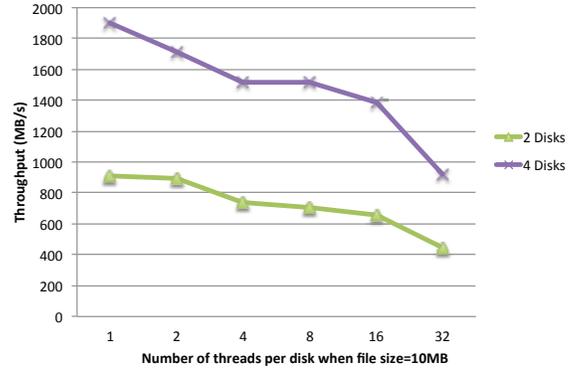


Fig. 6: Multiple disk throughput at NERSC using *iozone*: increased throughput using multiple disks, but slightly under the total sum of individual disks

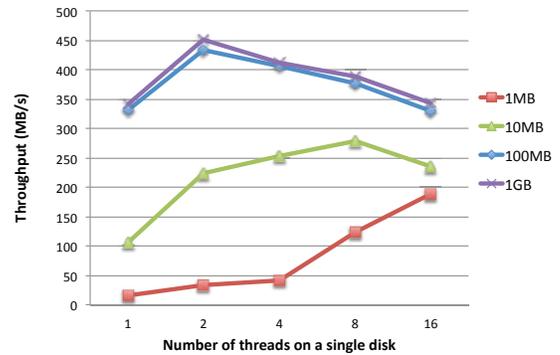
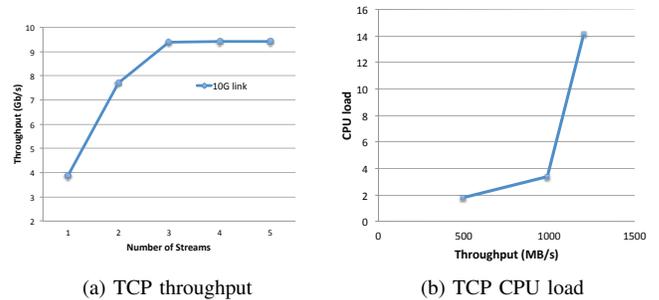


Fig. 7: GridFTP throughput: increasing throughput until disk throughput or data contention among multiple data streams becomes a bottleneck



(a) TCP throughput (b) TCP CPU load

Fig. 8: TCP protocol characteristics

Online uses options of concurrency=2, parallelism=8, and pipelining=5. In all other cases, the default setting is used: concurrency=2, parallelism=4, and pipelining=10.

Figure 9 shows the experimental results of all three cases. The data transfer experiments have been done from NERSC to StarLight by varying the number of hosts at NERSC from 1 to 5 and the number of hosts at StarLight from 1 to 3. The numbers of hosts at NERSC and StarLight are kept same except when the number of hosts at NERSC is greater than 3. In such cases, the number of hosts at StarLight is set to 3. We compute

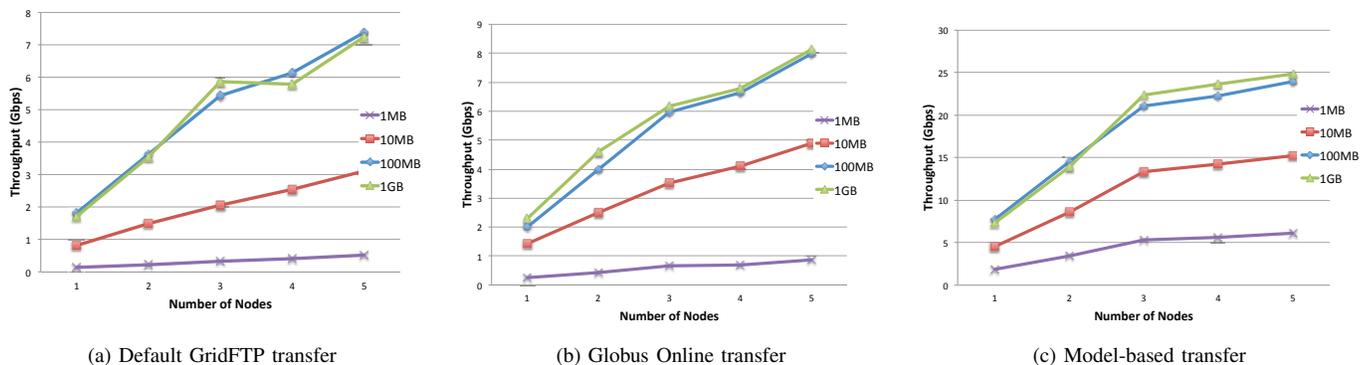


Fig. 9: Data transfer throughput comparison

the data transfer throughput by measuring the total time taken for transferring certain datasets. Globus Online outperforms the naive GridFTP especially in the cases of 1 MB and 10 MB datasets. Our model-based optimizations are 3-4 times faster than Globus Online in most cases, and 8 times faster than Globus Online, particularly, in the case of 1 MB datasets. It is mainly because our model can effectively identify the number of data flows based on disk throughput performance models and utilize data flow parallelism through multiple disks. For instance, with $-cc=2$ options, Globus Online can only utilize only two data streams from disks, which has a lot room for improvement, and cannot utilize the advantages of multiple disks. Based on models, our formulation in Figure 3 could find the proper number of data flows, 8, 6, 3, 2 in the case of 1 MB, 10 MB, 100 MB, 1 GB files, respectively. The data transfer throughput scales well as the number of hosts at NERSC increases up to 3 hosts. In case of 4 and 5 hosts at NERSC, the increasing rate slows down because those hosts have only local disks. In addition, our formulation could find a solution suggesting using multiple NICs in case that the aggregate throughput is beyond the capability of a 10G NIC.

The advantages of using model-based optimization formulations are as follows: (1) it can suggest the future hardware plan optimized for overall data transfer throughput just by simulating different configurations of hardware as well as software, (2) it can be used by systems such as Globus Online and other intelligent data transfer managers to adaptively optimize transfers for varying CPU resource availability and network status, and (3) it can provide basic models for simulating bulk data movement in next generation networks.

IV. CONCLUSIONS

We first model all the system components involved in end-to-end data transfer as a graph. We then formulate the problem whose goal is to achieve maximum data transfer throughput using parallel data flows. Our proposed formulations and solutions are evaluated through experiments on the ESnet testbed. The experimental results show that our approach is around four times faster than Globus Online in most datasets. Our models and formulations are extensible to more complex cases such as more deep software stacks and more complex system architectures (e.g., cluster). Accordingly, we will continue our research toward cluster-wide session

control and weighted CPU scheduler based on parameters determined by optimization formulation.

ACKNOWLEDGMENT

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

REFERENCES

- [1] A. Rajendran, P. Mhashilkar, H. Kim, D. Dykstra, and I. Raicu, "Optimizing large data transfers over 100Gbps wide area networks," Nov. 2012.
- [2] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The globus striped GridFTP framework and server," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, ser. SC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 54–64.
- [3] A. Shoshani, A. Sim, and J. Gu, "Grid resource management," J. Nabrzyski, J. M. Schopf, and J. Weglarz, Eds. Norwell, MA, USA: Kluwer Academic Publishers, 2004, pp. 321–340.
- [4] W. Allcock, J. Bresnahan, K. Kettimuthu, and J. Link, "The globus extensible input/output system (XIO): a protocol independent IO system for the grid," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, Apr. 2005, pp. 8–15.
- [5] E. Kissel and M. Swamy, "Evaluating high performance data transfer with RDMA-based protocols in wide-area networks," in *Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication*, ser. HPCC '12, Washington, DC, USA, 2012, pp. 802–811.
- [6] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, and I. Foster, "Gridftp pipelining," 2007.
- [7] R. K. et. al, "Lessons learned from moving earth system grid data sets over a 20 gbps wide-area network," in *19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 316–319.
- [8] E. K. Lee and R. H. Katz, "An analytic performance model of disk arrays," *SIGMETRICS Perform. Eval. Rev.*, vol. 21, no. 1, pp. 98–109, Jun. 1993.
- [9] [Http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/](http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/).
- [10] T. Hacker, B. Athey, and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, 2002, pp. 10–19.
- [11] 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," *Commun. ACM*, vol. 55, no. 2, pp. 81–88, Feb. 2012.