

On Causes of GridFTP Transfer Throughput Variance

Zhengyang Liu
University of Virginia
Charlottesville, VA 22904
zl4ef@virginia.edu

Malathi Veeraraghavan
University of Virginia
Charlottesville, VA 22904
mv5g@virginia.edu

Jianhui Zhou
University of Virginia
Charlottesville, VA 22904
jz9p@virginia.edu

Jason Hick
Lawrence Berkeley National
Laboratory
Berkeley, CA 94720
jhick@lbl.gov

Yee-Ting Li
SLAC National Accelerator
Laboratory
Menlo Park, CA 94025
ytl@slac.stanford.edu

ABSTRACT

In prior work, we analyzed the GridFTP usage logs collected by data transfer nodes (DTNs) located at national scientific computing centers, and found significant throughput variance even among transfers between the same two end hosts. The goal of this work is to quantify the impact of various factors on throughput variance. Our methodology consisted of executing experiments on a high-speed research testbed, running large-sized instrumented transfers between operational DTNs, and creating statistical models from collected measurements. A non-linear regression model for memory-to-memory transfer throughput as a function of CPU usage at the two DTNs and packet loss rate was created. The model is useful for determining concomitant resource allocations to use in scheduling requests. For example, if a whole NERSC DTN CPU core can be assigned to the GridFTP process executing a large memory-to-memory transfer to SLAC, then only 32% of a CPU core is required at the SLAC DTN for the corresponding GridFTP process due to a difference in the computing speeds of these two DTNs. With these CPU allocations, data can be moved at 6.3 Gbps, which sets the rate to request from the circuit scheduler.

Keywords

GridFTP, Performance problems in networking applications, Performance evaluation

1. INTRODUCTION

Large datasets are created and moved as part of scientific computing workflows. GridFTP is a popular protocol [1] for moving large datasets, the Globus implementation has been deployed in 3761 GridFTP servers [2], and Globus Online has about 7500 registered users [3]. While cloud computing and wide-area file systems will likely change data-movement

patterns, current Globus reports show that file movement is still significant at levels of 1 petabyte (PB) per day [2]. Therefore, in prior work [4], we analyzed GridFTP usage logs from three national scientific computing centers, National Energy Research Scientific Computing center (NERSC), SLAC National Accelerator Laboratory (SLAC), and National Center for Atmospheric Research (NCAR). The largest individual sessions (transfers of multiple files) in these logs were on the order of tens of terabytes (TB), and some sessions lasted several hours. In other words, there are sessions that require significant amount of network and computing resources to keep transfer durations short.

Our prior analysis of the GridFTP usage log files [4] also found *significant variance in throughput* of transfers executed between the same two data transfer nodes (DTNs), which are servers dedicated for file transfers. We analyzed logs collected for a set of 145 test transfers (32 GB each) between NERSC and Oak Ridge National Laboratory (ORNL), and found the coefficient of variation (CV) to be 32.58%. Also, we analyzed logs for 334 test transfers (32 GB each) separated into four types memory-to-memory (84), memory-to-disk (78), disk-to-memory (87), and disk-to-disk (85) from Argonne National Laboratory (ANL) to NERSC to have coefficients of variation (CV) of 35.69%, 31.63%, 30.80%, and 33.10%. It was surprising that memory-to-memory transfers had a similar CV to transfers that involved the disk.

To address network packet loss contribution to variance, the ESnet OSCARS project [5] enables the use of rate-guaranteed virtual circuits, which can offer lower packet loss rates than IP-routed paths. The Science DMZ project [6] addresses the packet loss rate and low bottleneck link rate problems caused by campus LAN switches and firewall filters, respectively, and proposes an option of staging data into the Science DMZ DTN from shared filesystems before moving data across the WAN. To address storage related causes of variance, the StorNet project [7] developed a version of the Storage Resource Manager, **BestMan-stor-net**, for co-scheduling storage and network resources (using TeraPaths [8] and ES-net's OSCARS).

Today's DTNs typically do not run any CPU job schedulers; instead users login interactively and execute file-transfer applications, or indirectly initiate file-transfer processes via tools such as Globus Online [9] and File Transfer Service

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

NDM'13 November 17, 2013, Denver CO, USA

Copyright 2013 ACM 978-1-4503-2522-6/13/11...\$15.00.
<http://dx.doi.org/10.1145/2534695.2534701>

(FTS) [10]. Without CPU job scheduling, we cannot control the contribution to variance by competing tasks on the DTNs. Schedulers such as the Portable Batch Scheduler (PBS) [11] usually assign a whole node to a computational task, but DTNs are not operated in this mode.

Problem statement: (i) Understand systematically the causes of variance and quantify the impact of each factor on throughput, and (ii) Create validated statistical models for transfer throughput as a function of various factors towards developing algorithms that determine the amount of concomitant resource allocations required at various managed resources on end-to-end paths.

Motivation: Solving the first problem allowed us to determine whether or not the above-listed architectural and software solutions [5–9] were sufficient. This study led us to formulate the second problem. Solving this second problem will support the advance-reservation mechanisms of OSCARS and StorNet by providing a mechanism for users/applications to determine what values to use in their resource scheduling requests.

Consider the broader question of why is it important to reduce throughput variance. Tasks that require a large fraction of a resource (e.g., 50% of link capacity to decrease transfer time of a large data set) are better supported by reservation systems than queueing systems. To support such tasks with acceptable user-perceived performance while operating the system at relatively high utilization, reservation systems are required. Reservation schedulers need task durations to be specified in requests, and hence solutions are required to minimize variance in durations [12].

Methods used: (i) Execute controlled experiments on research testbeds and collect measurements; (ii) Run instrumented large-sized transfers between operational data transfer nodes (DTNs) and collect measurements; and (iii) Create regression models for transfer throughput to enable algorithms that can determine concomitant resource allocations to use in scheduling requests.

Novelty and contributions: (i) A method for determining concomitant resource allocations at various components of an end-to-end path for use in scheduling requests is proposed. Since DTNs at different sites will naturally have different capabilities, and communication applications require resources at both ends of a unicast connection, such methods are required for any network/CPU/storage coscheduling techniques. (ii) A systematic identification of all causes of throughput variance and quantification of the specific impacts of various factors. For example, a non-linear regression model derived a function $f(p_i)$, where p_i is packet loss rate of the i^{th} transfer, as a contributing additive component to transfer throughput. Even though packet loss rate on a path between two operational DTNs was low (observed maximum over hourly transfers in one week was $1.1e-3$), the mean contribution to throughput, i.e., mean $f(p_i)$, was -683.3 Mbps when the average transfer throughput was 4.226 Gbps, making packet loss rate a significant factor. The OSCARS virtual circuits solution can thus be useful. (iii) For high-speed

data transfers, throughput is linearly related to the amount of CPU time available for the file-transfer application. (iv) CPU load of operational data transfer nodes can be high and highly variable, causing significant variance in GridFTP transfer throughput. Running CPU schedulers such as PBS and placing resource limits on concurrent jobs can help reduce throughput variance.

The rest of the paper is organized as follows. Section 2 reviews related work. Potential causes of variance are identified in Section 3. Sections 4 and 5 describe the experiments conducted and statistical models derived. The implications of this work are described in Section 6, and the paper is concluded in Section 7.

2. RELATED WORK

As stated in Section 1, ESnet’s OSCARS project, Science DMZ project and StorNet offer solutions for reducing transfer throughput variance. Globus Online [9] allows for automated invocations of GridFTP transfers, reducing the impact of user-specified values for configurable parameters (e.g., the use of `-fast` option). Others have used OSCARS circuits based overlay networks to demonstrate a significant improvement in throughput [13]. An effort to manage the number of GridFTP transfer sessions running simultaneously on a server was to modify the GridFTP server itself [14]. But this solution does not address the issue of other concurrent file-transfer processes. The General-purpose Architecture for Reservation and Allocation (GARA) [15] describes a framework for which resource managers were implemented for DiffServ networks, a Distributed Soft Real-Time (DSRT) CPU scheduler [16] and for the Distributed Parallel Storage System (DPSS) [17]. Also, I/O scheduling [18], and staging solutions [19], can reduce throughput variance. Software for co-allocation of network and CPU resources has been developed in the CONDOR [20] and KOALA projects [21].

Regression and neural network models have been proposed [22–24] to predict GridFTP transfer throughput as a function of disk I/O and network parameters. Disk I/O was found to account for 30% of transfer time [24]. In these studies, independent tools were used to obtain disk I/O measurements, and available network capacity, and the timestamps from GridFTP usage logs were used to determine the disk I/O rates and network capacity available at the time of each transfer. In contrast, in our studies, we measure CPU times, disk I/O times, packet losses (which are indicative of network congestion across all domains on the end-to-end path) corresponding to each transfer, and relate these variables to the GridFTP usage log reported throughput for the transfer. For example, if the previous approach was used, available network capacity would be obtained between two perfSONAR servers [25], but this may not be the available network capacity between the two DTNs. Therefore, our approach of using `tcpdump` during test transfers to obtain corresponding packet loss rates (representative of network conditions) between the DTNs will lead to higher-accuracy models.

3. CAUSES OF THROUGHPUT VARIANCE

Factors that contribute to transfer throughput variance are roughly classified into three groups: (i) intrinsic but less controllable factors, (ii) intrinsic and controllable factors,

and (iii) extrinsic factors. Factors that are *intrinsic* to the transfer itself but less controllable include file size, and bottleneck link rate and round-trip time (RTT) of the transfer path. Throughput of small files will be lower than that for large files on high bandwidth-delay product (BDP) paths due to TCP’s Slow Start procedure for increasing the congestion window (`cwnd`) at the sender. Therefore throughput of files of different sizes are not compared unless all files are much larger than BDP. Similarly, throughput of transfers on different end-to-end paths are not compared due to the dependence on bottleneck link rate and RTT.

Intrinsic controllable factors include choice of application and its corresponding parameters, and choice of transport protocol and its corresponding parameters. There are several applications for file transfers such as GridFTP, FDT, `bbcp`, `scp` [26], and even with a single application, there are different configurable parameters. For example, GridFTP offers a `-fast` option, which specifies that the TCP connection should be reused for transferring multiple files. The throughput without this option was measured to be 20 times slower than throughput with this option for transfers across a high BDP path. Similarly, there are different transport protocols, such as H-TCP, Cubic TCP, and UDT, each of which has its own corresponding parameters, such as TCP buffer sizes. Tutorial Websites [26] make recommendations for the best applications and transport protocols, and best settings for their corresponding parameters. Therefore, any variance caused by choice of these factors can be eliminated with user training.

Extrinsic factors, which depend on competing tasks and communications, include (i) CPU and memory bandwidth at the DTNs, (ii) available capacity and packet loss rate on the end-to-end path, and (iii) disk I/O access rates at the DTNs. Our experiments study the impact of these factors on transfer throughput. Competition for CPU resources is studied in controlled experiments on a testbed (Section 4.1) and with instrumented transfers between operational DTNs (Section 5.1). The impact of packet losses is studied in controlled testbed experiments (Section 4.2), and the impact of contention for disk I/O is studied using instrumented transfers (Section 5.2).

Contention for *memory bandwidth* is difficult to characterize. In multicore systems, even if one core was dedicated to handle applications for scheduled high-throughput large transfers, competition for memory bandwidth by best-effort (not using a resource scheduler) tasks executing on other cores (e.g., application processes to transfer small files or datasets) could add to throughput variance. Therefore in a managed solution where variance is controlled, CPU usage allocation on all cores of a DTN should be handled with a scheduler such as PBS.

4. CONTROLLED EXPERIMENTS

Sections 4.1 and 4.2 describe experiments on the impact of (i) available CPU time, and (ii) packet loss rates, respectively.

4.1 Impact of competition for CPU resources

Experiments were run on a 100 Gb/s wide-area network testbed. Varying number of competing processes were initi-

ated at the sender, at the receiver, and at both hosts, while a memory-to-memory transfer was being executed. A linear relationship was observed between transfer throughput and the amount of CPU resources available to the file transfer tasks at both hosts.

Experimental setup: The *ESnet 100G testbed* [27] was used in this experiment. It consists of two sites: NERSC, a supercomputing center in Northern California, and ANL, a DOE laboratory in the Chicago area. Each site has multiple high-performance hosts, each equipped with multiple 10 Gbps Ethernet Network Interface Cards (NICs) that are connected to an Alcatel-Lucent router. The NERSC and ANL routers are interconnected by a dedicated 100 Gbps Ethernet link. A researcher is allowed to schedule complete access to specific hosts, and to request a dedicated virtual circuit on the wide-area link.

For this experiment, one high-performance host was selected at each site: host `an1-memtp-1` has two AMD 6140 eight-core processors (for a total of 16 cores), 48 GB DDR3 RAM, and runs CentOS 5.7 (kernel: 2.6.32 x86_64), and host `nersc-disktp-1` has two Intel Xeon E5650 six-core processors (for a total of 12 cores), 64 GB DDR3 RAM, and runs CentOS 6.1 (kernel: 2.6.32 x86_64). Only one 10 Gbps NIC on each host was used in the experiment, and a 10 Gbps guaranteed-rate virtual circuit was requested on the wide-area link for this experiment. Thus, the end-to-end bottleneck link rate was 10 Gbps. The round trip time between the two hosts was 48.8 ms. Given the dedicated nature of the testbed-resource allocation, no other users were executing tasks on the selected hosts during our experiment.

Experimental software configuration: The GridFTP server `globus-gridftp-server` was executed at the two hosts, `nersc-disktp-1` and `an1-memtp-1`. A third host was used to initiate a transfer between the two `globus-gridftp-server` processes using the GridFTP client, `globus-url-copy`. This client maintains control-plane connections to the two GridFTP servers, but the data transfer proceeds directly between the server processes on `nersc-disktp-1` and `an1-memtp-1`.

Both `an1-memtp-1` and `nersc-disktp-1` were configured to run Hamilton TCP (H-TCP) [28]. As the bandwidth-delay product across the path is 61 MB, following best current practices [26], the maximum TCP buffer size was set to 128 MB, the Linux TCP autotuning send and receive buffer limits were configured to be (4096, 1048576, 128 MB), where the three numbers stand for minimum (in bytes), default (in bytes) and maximum sizes, respectively, TCP window scaling was enabled, `netdev_max_backlog` was set to 250000 and `txqueuelen` were set to 10000, respectively.

To create competing processes, the `double` utility from the `byte-unixbench` benchmark suite [29] was used. This utility works by performing double-precision floating-point computations that are highly demanding of CPU resources.

Experimental execution: Each test run consisted of a single 30-second memory-to-memory GridFTP transfer from `an1-memtp-1` to `nersc-disktp-1`. For each run, a monitoring script invoked the Linux `top` command every second, and

Table 1: ANL-to-NERSC memory-to-memory transfer throughput statistics (Mbps)

Conf.	Min.	1st Qu.	Med.	Mean	3rd Qu.	Max.	CV
baseline	8880	9093	9173	9155	9240	9286	0.012
snd-12	8666	9095	9163	9130	9240	9267	0.016
snd-16	964.9	983	992.4	994.4	1005	1022	0.018
snd-20	695.7	774	786.2	799.8	825	900.9	0.057
snd-24	691.9	714	729.9	731.3	747	787.6	0.034
rx-12	2647	2796	2805	2794	2819	2842	0.018
rx-16	1998	2102	2140	2184	2271	2818	0.072
rx-20	1877	2028	2103	2116	2196	2370	0.058
rx-24	1303	1556	1585	1766	2007	2289	0.164
both-12	2501	2795	2806	2789	2816	2911	0.027
both-16	913.5	960	975.3	976.2	988	1021	0.026
both-20	706.6	737	774.7	779.5	792	911.4	0.065
both-24	659.9	720	737.3	738.5	756	819	0.044

recorded the CPU time used by the GridFTP process. In the baseline configuration, the GridFTP process was the only user executable running on each host.

An additional set of runs were executed, each with a different configuration of processes. Specifically, 12 configurations were created by running four different numbers of `double` instances (i) on the sender alone (e.g., `snd-16` for 16 double instances), (ii) on the receiver alone (e.g., `rx-16`) and (iii) on both hosts (e.g., `both-16`). The minimum number of `double` instances used was 12, because with a smaller number of instances, there is no contention for cores on either host.

For each configuration, 35 test runs were executed, and for each run, transfer throughput, as reported in the GridFTP usage log, and CPU times for the GridFTP process (recall it is the only GridFTP process as no other users shared the hosts during our experiments) as reported by `top`, were collected.

Results: Table 1 summarizes observed-throughput statistics across the 35 test runs for each configuration. Fig. 1 shows average throughput (after removing outliers) for each configuration.

Average throughput for the baseline configuration tests was 9155 Mbps. When 12 `double` instances were executed on the sender (`anl-mempt-1`), the average throughput decreased only slightly (9130 Mbps), since the sender has 16 cores. However, when 12 `double` instances were run on the receiver side, the throughput dropped to 2794 Mbps as there are

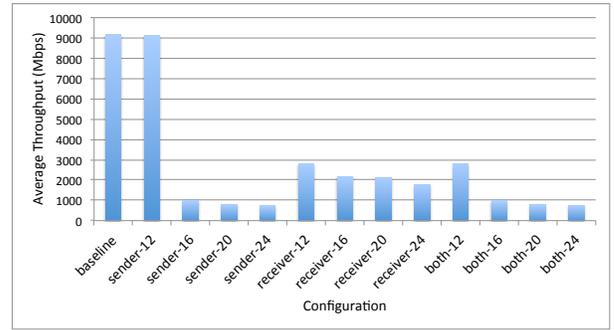


Figure 1: Average throughput in the ANL-to-NERSC experiments

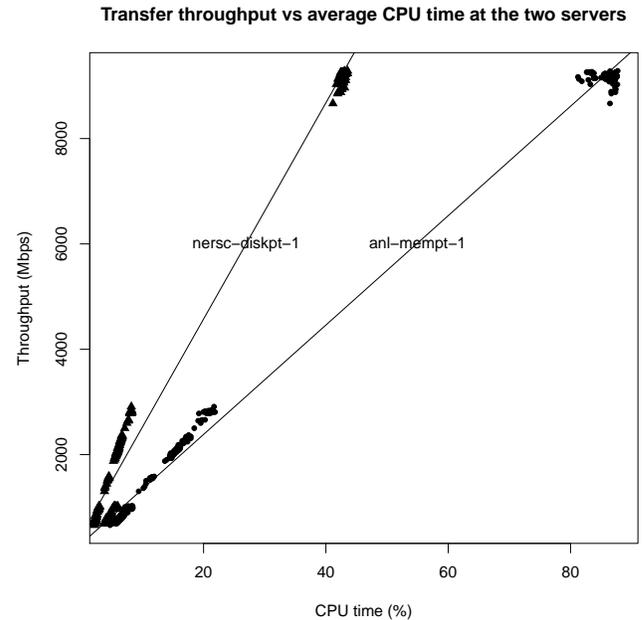


Figure 2: ANL-to-NERSC memory-to-memory transfers

only 12 cores in the receiver (`nersc-diskpt-1`), causing the GridFTP process to compete with the `double` processes for CPU time.

The scatter plots in Fig. 2 show a linear relationship between transfer throughput and average CPU time on each host (where the averaging is across all the CPU time values reported by `top` during the transfer), as the correlation coefficients between throughput and CPU time on `anl-mempt-1` and `nersc-diskpt-1` were 0.997 and 0.988, respectively. The two clusters seen in each of the plots in Fig. 2 are explained by heights of the bars in Fig. 1.

Another interesting observation can be made in Fig. 2: for a given throughput value, the CPU usage on the ANL side was higher than that on the NERSC side. In order to determine whether this phenomenon was caused by the direction of data flow, several test runs were executed from NERSC to ANL, reversing the roles of sender and receiver. The recorded CPU usage from these runs on the ANL side was still significantly higher than that on the NERSC side. This is because the Opteron CPUs in the ANL hosts are slower than the Xeon CPUs in the NERSC hosts.

Table 2: Experiment 2 throughput statistics

-p	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	CV
1	53.35	59.17	94.51	92.96	122.1	137.8	0.31
2	96.61	115.2	169.7	161.4	199.6	216.9	0.24
4	192	205.2	240.1	233.1	254.8	268.9	0.10
8	265.8	274.9	288	286.9	296.1	315.0	0.04
16	310.6	321.5	327.9	329.6	337.5	359.0	0.03

4.2 Impact of packet loss rate

A set of experiments was conducted on a local-area testbed called Emulab, which provides support for packet loss injection. The GridFTP application was used to study the impact of packet loss rate on transfer throughput under varying numbers of parallel TCP streams. Transfers involving higher numbers of parallel TCP streams recovered faster from packet losses.

Experimental setup and software configuration:

The University of Utah’s Emulab [30], a local-area testbed, was used because of its support for emulating a lossy environment. Each host has a dual-core Xeon CPU, 2GB RAM, and runs CentOS 5.5 (kernel: 2.6.18 x86_64). The GridFTP server was executed on two hosts, and Emulab software was used to inject random packet losses at specified rates. The TCP version used was Reno, and the TCP parameters were configured for maximum throughput.

Experimental execution and results: Artificial packet loss was introduced using the Emulab software by dropping a fixed percentage of packets. The packet loss rates tested were 0.1%, 0.2%, and 0.5%. Each test run consisted of a 30-second memory-to-memory GridFTP transfer. For each configuration (packet loss rate and number of parallel streams setting), 30 number of runs were executed. Error bars were less than 8.6 %. The *rtt* was 0.2 ms, bottleneck link rate was 1 Gbps, and TCP *mss* was 1460 bytes.

Fig. 3 shows that in all configurations there was a significant drop in throughput when packet loss was introduced. However, transfers with a larger number of parallel TCP streams had significantly higher throughput than those with fewer parallel streams. For example, when emulated loss rate is 0.5%, the average throughput of 16-stream transfers was 321.14 Mbps, while the average throughput of 1-stream transfers was only 57.27 Mbps (a factor of 5.6).

Another interesting trend that can be observed in Fig. 3 is the smaller slopes of plots corresponding to larger numbers of parallel streams. Table 2 shows that in the presence of losses, the coefficient of variation (CV) was smaller with larger numbers of TCP connections. This is because the aggregate TCP congestion window recovers faster after losses when there are multiple TCP connections. Thus, in the presence of packet loss, not only did the transfers using multiple parallel streams enjoy higher throughput, the coefficient of variation was smaller.

5. INSTRUMENTED TRANSFERS BETWEEN OPERATIONAL SERVERS

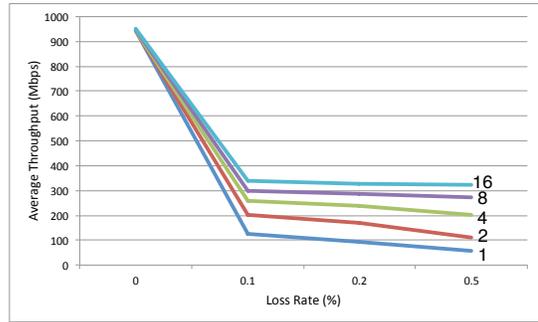


Figure 3: Experiment 2: Mean throughput in the presence of packet losses for a 30-sec transfer (corresponds to ~ 3.4 GB in 0-loss-rate settings)

While the controlled experiments described in Section 4 show high correlation between transfer throughput and the CPU time used by the GridFTP process, the artificial CPU load used in the experiments may differ significantly from the CPU load in operational data transfer servers. Also, operational data transfer nodes have only 4 or 8 cores, unlike the 12- or 16-core machines in the ESnet 100G testbed. Therefore, an experiment was conducted between the operational data transfer nodes at NERSC and SLAC.

The data transfer node at NERSC has 2 dual-core AMD Opteron processors (4 cores in total), 8 GB of RAM, and runs CentOS 5.8 (kernel: 2.6.18 x86_64); the node at SLAC side has 2 quad-core Intel Xeon processors (8 cores in total; 16 logical cores with Hyper-Threading), 48 GB of RAM, and runs RHEL 5.9 (kernel: 2.6.18 x86_64). Both nodes have 10 Gbps Ethernet NICs. The path between these sites traverses the two campus networks and ESnet (a core research-and-network that provides IP services to US DOE national laboratories [31])¹. The bottleneck link rate on this path is 10 Gbps and the round-trip time is 2.47 ms. Unlike in the controlled experiments, the traffic generated by our experimental GridFTP processes competed for CPU time with other processes at the DTNs, and for bandwidth with actual traffic on the NERSC-SLAC path. The *traceroute* program showed that the path traverses 10 IP routers.

To isolate the impact of disk I/O access, first memory-to-memory transfers were executed, and a model developed for transfer throughput as a function of CPU time available to the file transfer process at each end (DTN), and packet loss rate across the path, which captures network congestion at LAN and WAN switches/routers. Section 5.1 describes this model. The key finding is that it is possible to create a model that can be used to determine concomitant resource allocations needed at the two DTNs to sustain a certain transfer rate. This rate can be used to request a virtual circuit from site and ESnet circuit schedulers.

Next, a set of preliminary experiments was conducted to study disk I/O access rates. It shows that disk I/O time was on average about 45% of total transfer time. These results are presented in Section 5.2.

5.1 Memory-to-memory experiments

¹This core network carries actual user traffic unlike the “ES-net 100G testbed,” which is used strictly by researchers for experiments.

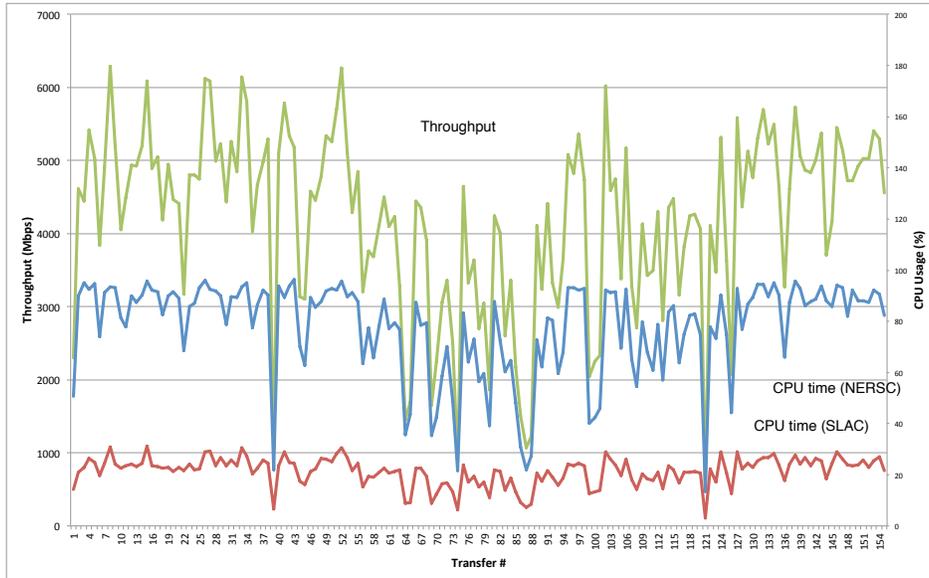


Figure 4: Throughput (left y-axis in Mbps) and average CPU time (%) at the NERSC and DTN servers (right y-axis) for each of the 155 transfers (x-axis)

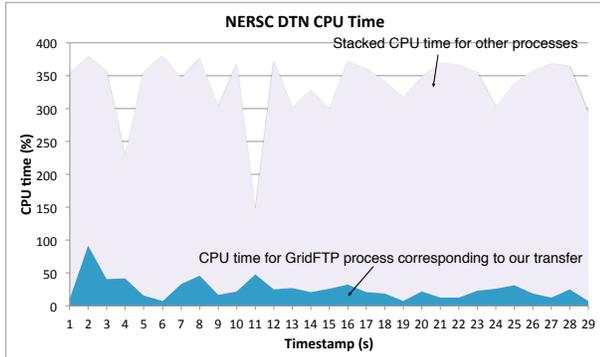


Figure 5: NERSC DTN CPU time for GridFTP and other processes during a transfer

The `globus-gridftp-server` processes are executed for usage by regular users of these data transfer nodes. For our experiment, we used the Linux `cron` command to start the `globus-url-copy` client on a scheduled basis, i.e., once every hour. Each invocation of the `globus-url-copy` client initiated a single 30-second memory-to-memory transfer from SLAC to NERSC. A script initiated (i) the `top` program to collect CPU time used by the GridFTP process corresponding to our transfer, and (ii) `tcpdump` to collect network packets from our GridFTP transfer. Transfer throughput was recorded from the GridFTP usage logs. Data sets, consisting of throughput, CPU time, and TCP retransmissions for each transfer, were collected in Oct. and Dec. 2012. Both data sets consist of information for 155 transfers. The impact of two factors, CPU time and packet loss rate, are considered below.

Impact of CPU time:

The coefficient of variation of transfer throughput in the Oct. and Dec. data sets were 0.156 and 0.292, respectively, which are both larger than the controlled-experiment values in Table 1. Table 4 shows that there is considerable variance

Table 3: Correlation coefficients between throughput and CPU usage

	Oct	Dec
NERSC DTN	0.820	0.954
SLAC DTN	0.904	0.976

in transfer throughput.

The correlation coefficients between transfer throughput and process CPU time were high as in the controlled experiments. The values are shown in Table 3. Fig. 4 plots the throughput and CPU time for each transfer in the Dec. 2012 data set. As visible in these plots, when a transfer receives a smaller percentage of CPU time, its throughput drops.

To verify that the cause of lower CPU time allocation for our GridFTP process was due to competing processes, for selected runs, the CPU time was recorded for all processes. Fig. 5 shows the CPU time allocation during one particular run in which the total CPU time allocated for other processes overwhelms that of our GridFTP process at the NERSC DTN. The average CPU time used by our GridFTP process for this transfer was 25%, and the throughput was 635 Mbps (which is at the lower end as seen in Table 4). In Fig. 5, the stacked height at each time stamp was around 400%, indicating that the 4-core server was operated under heavy load.

Fig. 4 also shows that the NERSC CPU usage and SLAC CPU usage plots have similar patterns, suggesting a linear relationship. Indeed the correlation coefficient is 0.94. A linear model was fitted as follows:

$$SLACcpu_i = \beta_0 + \beta_1 NERSCcpu_i + \epsilon_i \quad (1)$$

where $SLACcpu_i$ and $NERSCcpu_i$ represent the SLAC CPU usage and NERSC CPU usage (both expressed as percentages) corresponding to the i^{th} transfer, respectively, and ϵ_i is the residual term. Since the NERSC CPU was the bottleneck (the Opterons at NERSC are slower than the Xeon

CPUs at SLAC), SLAC CPU usage was selected as the dependent variable and NERSC CPU usage was the independent variable. The residual terms ϵ_i obtained from this model were saved for the next regression, which includes packet loss rates.

Impact of packet loss rate: For the transfers executed in Dec. 2012, the `tcpdump` tool was used to capture all TCP packet headers, and the `tcptrace` tool [32] was used for analysis. Of all the 155 SLAC-to-NERSC transfers in the Dec. data set, 14% (22) transfers had a zero packet retransmission rate. The median retransmission rate was $4e-5$, while the maximum retransmission rate was $1.1e-3$, as shown in Table 4. The relationship between transfer throughput and retransmission rate is not linear with a correlation coefficient of -0.12 . This observation is to be expected given the well-known result that TCP throughput is proportional to $1/\sqrt{p}$ [33]. As the newer TCP variants used in these experiments may not fit this model, we used a non-linear regression as presented next.

Non-linear regression model combining the factors:

The transfer throughput (y_i) was expressed as a function of (i) CPU time at NERSC ($NERSCcpu_i$), (ii) residuals obtained from linear regression in (1) (ϵ_i), and (iii) packet loss rates p_i , in a regression model:

$$y_i = \beta'_1 NERSCcpu_i + \beta'_2 \epsilon_i + f(p_i) + e_i, \quad (2)$$

where $f(p_i)$ is the non-linear term, and e_i is the random error term. Using B-splines, the function $f(p_i)$ can be approximated as:

$$f(p_i) \approx \sum_{j=1}^{k+m} \alpha_j B_j(p_i) \quad (3)$$

where $B_j(p_i)$ are the B-spline basis functions.

The `bsplineS` function from the R `fda` package [34] was used to generate basis functions $B_j(p_i), 1 \leq j \leq (k+m)$ where k is the number of inner quantile knots in the range of p_i , and m is the spline order and $m-1$ is the degree of the piecewise polynomials $B_j(p_i)$. A leave-one-out cross-validation (LOOCV) procedure was executed to find the optimal number of inner knots for two values of m . The best fit was found for $m=3$ (i.e., the basis functions are piecewise quadratic polynomials) and $k=2$ (33% and 66% quantile knots). Before executing the `bsplineS` function, 4 outliers were removed, leaving 151 transfers for use in the model.

The B-spline approximation allows for a linear model to be estimated to capture the nonlinear effect of p_i . Thus, after finding the basis functions using `bsplineS`, a linear regression model was solved using the R `lm` function. A high adjusted R-squared value of 0.997 was obtained from this fit, which is supported by Fig. 6a. Fig. 6b shows the estimated non-linear function $f(p_i)$ smoothed by B-splines, which models the effect of packet loss rate on transfer throughput. The range of packet loss rates for this model was $(5e-5, 3.5e-4)$. Of the 151 transfers, 60 transfers have packet loss rates within this range. The upper bound of this range is the 98-percentile value in the set of packet loss rates. The reason the estimated function $\hat{f}(p_i)$ is not shown for packet loss

rates less than $5e-5$ is because of the well-known boundary effects of smoothing methods in Statistics. For these very small values of packet loss rate (i.e. less than $5e-5$), the dominant factors in determining memory-to-memory throughput are the CPU times available to the transfer process at the two hosts.

The 95% confidence intervals for the regression coefficients β'_1 and β'_2 in (2) are (60.3,65.1) and (133.2,173.1), respectively. Using the statistics presented in Table 4, we can compute the mean throughput (4.226 Gbps) using the mean NERSC CPU usage (78.28%), mean value of linear-regression residuals (0), and mean value of $f(p)$ (-683.3 Mbps), and the mean regression coefficient values for the NERSC CPU usage and linear-model residuals of 62.708 and 153.194, respectively.

Using this model, concomitant resource allocations can be determined. For example, consider the maximum values shown in Table 4. If 97% or effectively a whole NERSC CPU can be assigned to the GridFTP process executing a large memory-to-memory transfer to SLAC, then only 32% of a CPU core is required at the SLAC DTN for the corresponding GridFTP process due to a difference in the computing speeds of these two DTNs. With these CPU allocations, data can be moved at 6.3 Gbps, which sets the rate to request from the circuit scheduler. The Portable Batch Scheduler [11] supports a feature to place resource limits on jobs, and controls how jobs are assigned to resources (e.g., to assign 32% of a CPU as needed in the example). As mentioned earlier, today's DTNs do not run PBS or any such scheduler; but for large transfers since large resource allocations are required to reduce transfer times, reservation schedulers will be beneficial.

5.2 Disk I/O access times

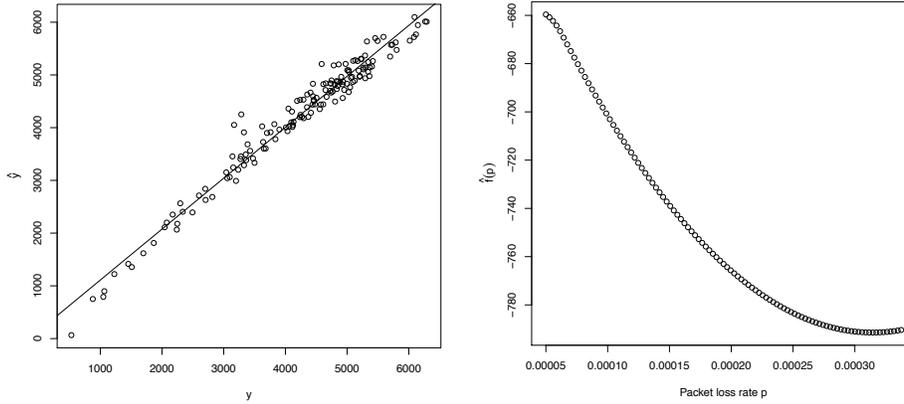
File systems are commonly shared between computational systems and data transfer nodes. For example, the NERSC `global scratch` file system is shared between the DTN servers and `Hopper`, `Carver`, and other NERSC computing clusters [35]. This system was used to: (i) characterize the variance in disk I/O read and write times with a `dd` benchmark, and (ii) measure the disk I/O times and file transfer times for NERSC-to-SLAC disk-to-memory GridFTP transfers (the `strace` utility was used to calculate the time spent on disk I/O read operations). These are a preliminary set of experiments.

Both experiments were conducted on `dtn01.nersc.gov`, a server with 8GB RAM. At regular intervals, a script consisting of the following steps was executed:

1. Invoke `dd` to write an 8 GB file (`first file`) to the `global scratch` file system with the `sync` system call to force the completion of pending disk writes, and record the time taken for the write operation.
2. Invoke `dd` to write another 8 GB file (`second file`) to the `global scratch` file system to ensure that the first file is no longer in the filesystem cache, which is required for the next step.
3. Invoke `dd` to read back the first file (which is now on disk, not cache) and record the time taken for the read

Table 4: Instrumented transfers from SLAC-to-NERSC DTNs, Dec 2012; $f(p_i)$ is given in (2)

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Retransmission rate	0	1.1e-5	4.1e-5	7.3e-5	8.4e-5	1.1e-3
$f(p_i)$ in Mbps	-791.8	-679.0	-670.5	-683.3	-664.0	-658.9
Throughput (Gbps)	0.5316	3.452	4.48	4.226	5.071	6.29
NERSC CPU usage (%)	13.45	70.26	86.81	78.28	91.78	96.42
SLAC CPU usage (%)	2.968	18.83	22.47	21.45	24.54	31.14
ϵ residuals from (1)	-5.815	-1.241	-0.2146	0.0000	0.9625	5.296



(a) Throughput vs. estimated throughput

(b) Estimated $\hat{f}(p)$ function

Figure 6: Results of the regression model ((2))

operation. (Conveniently the second file will be forced out of the cache).

- Using `globus-url-copy`, transfer the second file to `/dev/null` on the SLAC DTN, and use `strace` to record system calls for further analysis. The only disk I/O operation required is the reading of the second file, which we know is not in the cache.

While the first `dd` call does write out the 8 GB file to disk due to the `sync` call, since the cache still retains a copy, omitting the second step and directly executing the third step could result in a shorter reported time. Unlike in the CPU time experiments reported in Section 4, where third-party transfers were initiated between GridFTP servers running on the NERSC and SLAC DTNs, here the file transfer was executed by the `globus-url-copy` client itself (which was in turn executed by `strace`) because `strace` can only trace on its child processes on the test DTN server due to security policies, and with third-party transfers, `globus-gridftp-server` processes are spawned by a super-server daemon `xinetd`, and hence cannot be traced on the test server.

A sample of 116 data points was collected from the periodic runs executed on an hourly basis. Statistics for the disk I/O read and write times are shown in Table 5. Recall that while our test `dd` tasks were being executed, the `global scratch` file system was simultaneously being accessed by processes running on Hopper, Carver, and other computing systems. Therefore there was significant variance in both read and write times.

For the GridFTP disk-to-memory transfers executed with

Table 5: NERSC Global Scratch file system operations

	Read (MB/s)	Write (MB/s)
Min	109.4	173.1
1st Qu.	247.6	530.2
Median	306.9	585.6
Mean	372.8	559.3
3rd Qu.	513.0	654.8
Max	746.8	747.4
CV	47.5%	23.5%

`strace` monitoring, the output contains information on every system call made, the parameters to the system calls, their return values, and most importantly, the time spent in each system call. The impact of `strace` was small, increasing total transfer time in one instance from 24 s without `strace` to 25 s with `strace`. A manual examination of the `strace` log was performed to identify the file descriptor corresponding to the data file, which was then used in an `awk` script to identify all the `read` system calls to the same file descriptor. The total time spent in these `read` system calls was computed, and is referred to as the disk I/O time. Table 6 shows statistics for 116 transfers. On average, 45% of the total transfer time was used in disk I/O operations.

Table 6 also shows statistics for disk-to-memory transfer throughput. The transfer with minimum throughput at 0.768 Gbps spent 81.24% of total transfer time on disk I/O, the second highest percentage among all 116 transfers. However, the maximum transfer throughput at 3.228 Gbps was achieved with a disk IO/total time ratio of 35.08%, while the transfer with minimum disk IO/total time ratio at 26.47% only achieved a throughput of 1.814 Gbps, slightly lower than the mean and median throughput. These results in-

Table 6: NERSC-to-SLAC disk-to-mem transfers

	Ratio of Disk IO time to total transfer time	Throughput (Gb/s)
Min	0.2647	0.768
1st Qu.	0.3415	1.687
Median	0.4010	1.994
Mean	0.4524	1.989
3rd Qu.	0.5523	2.32
Max	0.8191	3.228
CV	30.37%	26.42%

dicating that while disk I/O is an important factor affecting throughput, it is not the only one.

6. DISCUSSIONS

Synthesizing information learned from the various experiments, we make the following networking solution recommendation to achieve guaranteed-rate (low-variance), high-throughput transfers. It consists of four steps: (i) leverage Science DMZ and stage a local copy from the filesystems that are shared with compute clusters into the Science DMZ servers to eliminate competition for disk I/O and the resultant variance caused to wide-area file transfer throughput, (ii) use a two-phase cycle in the Science DMZ DTNs to alternate between unmanaged periods during which local copies are made from/to shared filesystems, and managed periods for wide-area transfers in which PBS is used on all cores to control the number of concurrently scheduled wide-area file-transfer processes (PBS is required to manage tasks on all cores to avoid variance resulting from memory contention), (iii) using the concomitant resource allocation models determined from prior runs, simultaneously request CPU time allocations at the DTNs and an end-to-end (virtual) circuit with a guaranteed rate (policing and scheduling mechanisms should be enabled [36]), and (iv) execute a file-transfer application in which the sending rate is set to match the virtual circuit rate using a transport protocol such as UDT [37] with which the sending rate can be fixed.

The amount of CPU time required at the two ends may be different because invariably the servers at the two ends will not have the same computing capacity as seen in our experiments. Since the scheduling process requires a simultaneous allocation of resources from multiple entities, i.e., the DTNs and the network, resource request and commit phase may need multiple communications to accommodate varying levels of resource availability. Overall, we conclude that guaranteed-rate (low-variance), high-throughput transfers can be achieved across circuits provided servers (all cores) are dedicated for PBS-controlled transfers.

7. CONCLUSIONS

Scientists who move large data sets want high throughput with low variance for their transfers. Towards determining the extrinsic causes of transfer throughput variance, i.e., factors that depend on competing tasks and communications, the impacts of server CPU time, packet loss rate, and disk I/O access time, were studied in depth. Experiments were conducted on testbeds (high-speed wide-area as well as local-area) and across operational data transfer nodes at two national scientific research centers, and statistical models were created for transfer throughput. Our conclusions

are as follows: (i) Models can be created for determining concomitant resource allocations at various managed components of an end-to-end path for use in scheduling requests; (ii) Packet losses contribute a significant amount to throughput variance, and therefore reserved virtual circuits will be useful for large data set transfers; and (iii) CPU usage levels at data transfer nodes should be controlled with schedulers such as the Portable Batch Scheduler (PBS) and resource limits should be placed on concurrent jobs to reduce throughput variance for high-speed transfers.

8. ACKNOWLEDGMENTS

The authors thank Brent Draney and Jason Lee, NERSC, and Wei Yang, SLAC, for access to the data transfer nodes, and Brian Tierney, Eric Pouyoul, and several others on the ESnet testbed support team. This research used resources of the ESnet Testbed, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231. This work was supported by NSF grants OCI-1038058, OCI-1127340, and CNS-1116081, and U.S. DOE grants DE-SC0002350 and DE-SC0007341.

9. REFERENCES

- [1] GridFTP v2 Protocol Description. [Online]. Available: <http://www.ggf.org/documents/GFD.47.pdf>
- [2] S. Tuecke, "Enhancing and supporting GridFTP: An essential component of DOE high-speed networking," 2013, talk presented at DOE Next-generation Networks for Science Program PI Meeting. [Online]. Available: <https://indico.bnl.gov/contributionDisplay.py?contribId=36&confId=566>
- [3] I. Foster, "Transforming research using software as a service," 2013, keynote speech at TIP 2013. [Online]. Available: <http://www.internet2.edu/presentations/tip2013/20130115-Foster-keynote.pdf>
- [4] Z. Liu, M. Veeraraghavan, Z. Yan, C. Tracy, J. Tie, I. Foster, J. Dennis, J. Hick, Y. Li, and W. Yang, "On using virtual circuits for GridFTP transfers," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 81:1–81:11.
- [5] On-Demand Secure Circuits and Advance Reservation System (OSCARS). [Online]. Available: <http://www.es.net/services/virtual-circuits-oscars/>
- [6] Science DMZ: A Scalable Network Design Model for Optimizing Science Data Transfers. [Online]. Available: <http://fasterdata.es.net/science-dmz/>
- [7] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee, "Stornet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, 2011, pp. 121–126.
- [8] TeraPaths: Configuring End-to-End Virtual Network Paths with QoS Guarantees. [Online]. Available: <https://www.racf.bnl.gov/terapaths/>
- [9] 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, no. 2, Feb. 2012.
- [10] Grid File Transfer Service (FTS). [Online]. Available: <http://www.scc.kit.edu/dienste/5726.php>
 - [11] Portable Batch System (PBS). [Online]. Available: <http://www.pbsworks.com/>
 - [12] Q. Wu, M. Zhu, Y. Gu, P. Brown, X. Lu, W. Lin, and Y. Liu, "A distributed workflow management system with case study of real-life scientific applications on Grids," *Journal of Grid Computing*, vol. 10, no. 3, pp. 367–393, 2012. [Online]. Available: <http://dx.doi.org/10.1007/s10723-012-9222-7>
 - [13] L. Ramakrishnan, C. Guok, K. Jackson, E. Kissel, D. Swamy, and D. Agarwal, "On-demand overlay networks for large scientific data transfers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 359–367.
 - [14] J. Bresnahan, M. Link, R. Kettimuthu, and I. Foster, "Managed GridFTP," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, May 2011, pp. 907–913.
 - [15] I. Foster, M. Fidler, A. Roy, V. Sander, and L. Winkler, "End-to-end quality of service for high-end applications," *Computer Communications*, vol. 27, no. 14, pp. 1375–1388, 2004.
 - [16] H.-H. Chu and K. Nahrstedt, "CPU service classes for multimedia applications," in *Multimedia Computing and Systems, 1999. IEEE International Conference on*, vol. 1, 1999, pp. 296–301 vol.1.
 - [17] B. Tierney, J. Lee, L. T. Chen, H. Herzog, G. Hoo, G. Jin, and W. E. Johnston, "Distributed parallel data storage systems: a scalable approach to high speed image servers," in *Proceedings of the second ACM international conference on Multimedia*, ser. MULTIMEDIA '94. New York, NY, USA: ACM, 1994, pp. 399–405. [Online]. Available: <http://doi.acm.org/10.1145/192593.192709>
 - [18] G. Shipman, S. Atchley, Y. Kim, G. Vallee, G. Bosilca, T. Herault, and S. Moreaud, "An adaptive end-to-end approach for terabit data movement optimization," Next-Generation networks for science program PI Meeting, Emeryville, CA, March 18-20. [Online]. Available: <https://indico.bnl.gov/getFile.py/access?contribId=26&resId=0&materialId=slides&confId=566>
 - [19] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky, "Just in time: adding value to the IO pipelines of high performance applications with JITStaging," in *Proceedings of the 20th international symposium on High performance distributed computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 27–36. [Online]. Available: <http://doi.acm.org/10.1145/1996130.1996137>
 - [20] J. Basney and M. Livny, "Managing network resources in Condor," in *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on*, 2000, pp. 298–299.
 - [21] H. H. Mohamed and D. H. J. Epema, "The design and implementation of the KOALA co-allocating grid scheduler," in *Proceedings of the 2005 European conference on Advances in Grid Computing*, ser. EGC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 640–650. [Online]. Available: http://dx.doi.org/10.1007/11508380_65
 - [22] S. Vazhkudai and J. M. Schopf, "Using regression techniques to predict large data transfers," *International Journal of High Performance Computing Applications*, vol. 17, no. 3, pp. 249–268, 2003.
 - [23] R. Rahman, K. Barker, and R. Alhajj, "Predicting the performance of GridFTP transfers," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, April 2004, p. 238.
 - [24] S. Vazhkudai and J. Schopf, "Using disk throughput data in predictions of end-to-end grid data transfers," in *Grid Computing & GRID 2002*, ser. Lecture Notes in Computer Science, M. Parashar, Ed. Springer Berlin Heidelberg, 2002, vol. 2536, pp. 291–304.
 - [25] Esnet perfSONAR dashboard. [Online]. Available: <http://ps-dashboard.es.net>
 - [26] Linux tuning. [Online]. Available: <http://fasterdata.es.net/host-tuning/linux/>
 - [27] ESnet 100G Testbed. [Online]. Available: <http://www.es.net/RandD/100g-testbed/>
 - [28] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," Proc. of PFLDnet, Feb. 16-17, 2004.
 - [29] byte-unixbench - A Unix benchmark suite. [Online]. Available: <http://code.google.com/p/byte-unixbench/>
 - [30] "Emulab - network emulation testbed." [Online]. Available: <https://www.emulab.net/>
 - [31] ESnet. [Online]. Available: <http://www.es.net/>
 - [32] tcptrace. [Online]. Available: <http://www.tcptrace.org/>
 - [33] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, Jul. 1997.
 - [34] "fda: Functional Data Analysis R statistical package." [Online]. Available: <http://cran.r-project.org/web/packages/fda/index.html>
 - [35] NERSC file systems. [Online]. Available: <http://www.nersc.gov/users/data-and-networking/file-systems/>
 - [36] Z. Yan, M. Veeraraghavan, C. Tracy, and C. Guok, "On how to provision Quality of Service (QoS) for large dataset transfers," in *Proceedings of the Sixth International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ)*, 2013.
 - [37] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Comput. Netw.*, vol. 51, no. 7, pp. 1777–1799, May 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.comnet.2006.11.009>