

# Optimizing Large Data Transfers over 100Gbps Wide Area Networks

Anupam Rajendran<sup>1</sup>, Parag Mhashilkar<sup>2</sup>, Hyunwoo Kim<sup>2</sup>, Dave Dykstra<sup>2</sup>, Gabriele Garzoglio<sup>2</sup>, Ioan Raicu<sup>1,3</sup>  
arajend5@hawk.iit.edu, parag@fnal.gov, hyunwoo@fnal.gov, dwd@fnal.gov, garzoglio@fnal.gov,  
iraicu@cs.iit.edu

<sup>1</sup>Department of Computer Science, Illinois Institute of Technology, Chicago IL, USA

<sup>2</sup>Scientific Computing Division, Fermi National Accelerator Laboratory, Batavia IL, USA

<sup>3</sup>Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL, USA

**Abstract**— The Advanced Networking Initiative (ANI) project from the Energy Services Network provides a 100 Gbps testbed, which offers the opportunity for evaluating applications and middleware used by scientific experiments. This testbed is a prototype of a 100 Gbps wide-area network backbone, which links several Department of Energy (DOE) national laboratories, universities and other research institutions. These scientific experiments involve movement of large datasets for collaborations among researchers at different sites and thus require advanced infrastructure for supporting large and fast data transfers. A 100 Gbps network testbed is a key component of the ANI project and is used for DOE's science research programs. This work presents results towards obtaining maximum throughput in large data transfers by optimizing and fine-tuning scientific applications and middleware to use this advanced infrastructure efficiently. A detailed performance evaluation is discussed measuring both applications, from High Energy Physics (HEP) and from data transfer middleware (GridFTP, Globus Online, Storage Resource Management, XrootD and Squid) at 100 Gbps speeds and 53 ms of latency. Results show that up to 97% efficiency of such high bandwidth high latency network is possible, achieving 80-90 Gbps in most test cases with a peak transfer rate of 100 Gbps.

**Keywords**- ANI Testbed; Grid Computing; Big Data; Caching; Wide Area Network

## I. INTRODUCTION

In 2011 Energy Services Network (ESnet) [1] deployed a 100 Gbps wide-area network backbone linking several Department of Energy (DOE) [2] National Laboratories, universities and research institutions. ESnet also provided a 100 Gbps cross-country prototypical testbed as a part of Advanced Networking Initiative (ANI) [3]. This testbed helps network researchers validate the infrastructure by testing the scalability of different applications and middleware in such a high-speed environment.

Fermilab hosts the US Tier-1 Center for the Large Hadron Collider (LHC) Compact Muon Solenoid (CMS) experiment. This experiment running at CERN, the European Organization for Nuclear Research, in Geneva, Switzerland, generates large data that must be distributed for collaboration purposes. Fermilab acts as the tier 1 site for CMS in US and is responsible for distribution of the data over the WAN to different institutions for further processing. The current scale of data produced from these experiments is 40 PB with a WAN traffic of 30 Gbps and 140 Gbps of LAN traffic from archive to local processing farms. Apart from the HEP communities, Fermilab is also involved with other Grid

initiatives like Extreme Science and Engineering Discovery Environment (XSEDE) [4] and the Open Science Grid (OSG) [5].

In preparation for Fermilab connecting to the 100 Gbps backbone, we have been running the High Throughput Data Program (HTDP) since 2011. The HTDP project is driven by three major thrusts; (1) Evaluate Grid middleware on 100 Gbps to prepare lab and its stakeholders for the transition (2) Foster the process to make network a manageable resource by actively participating in OSG Networking activity (3) Integrate network intelligence with data and job management middleware.

This paper describes the work done as part of the HTDP project in context of evaluating Grid applications and middleware on the 100 Gbps ANI testbed. *The contribution of this paper is the evaluation of different layers and services (GridFTP, Globus Online, Storage Resource Management, XrootD and Squid) involved in end-to-end analysis systems at 100 Gbps speeds in a wide-area network testbed, showing that it is feasible with careful tuning to utilize high bandwidth high latency networks.* This work aims to make effective use of the advanced infrastructure and ensure maximum throughput possible across each layer and service. This work also includes technological investigations to identify gaps in the middleware components integrated with the analysis systems and the development of system prototypes to adapt to the high-speed network.

The rest of the paper is organized as follows: Section II describes the related work about the tests conducted for evaluating the Grid middleware. In Section III, we present the essential overview of different middleware namely GridFTP [6][7][8], Globus Online[12], Storage Resource Management (SRM) [13][14], XrootD [15] and Squid [16]. Section IV describes the 100 Gbps prototype used for middleware evaluation. It also shows the evaluation and experimental results of performance. Conclusions are drawn and future work is envisioned in Section V.

## II. RELATED WORK

One of the goals of HTDP is to analyze the performance of the end-to-end analysis systems of stakeholders by performing large data transfers over the 100 Gbps ANI testbed. The experimental results would reveal the gaps in different middleware that needs to be corrected for making them adaptable in the high-speed environment.

In [6], the authors reported the design and performance of the Globus Striped GridFTP Framework on a 30 Gbps network. The tests showed that this framework achieved

90% efficiency and is hence faster and scalable than other FTP servers.

The authors of [7] demonstrated transfer of Grid datasets on a 20 Gbps network during Super Computing 2009 Bandwidth Challenge. They achieved a sustained rate of 15 Gbps when moving 10 TB of data.

In [8], network performance of WAN data transfer was characterized using GridFTP and FDT protocols. The tests utilized two 10 Gbps links and achieved a data transfer rate of 15 Gbps. These results were used to analyze WAN data transfer issues related to a Hadoop Distributed File System (HDFS) storage.

In 2010, a data management framework suitable for distributed biomedical research environments was proposed in [9]. It is a distributed data-sharing system and is used by medical researchers at Bioinformatics Research Network (BIRN). The framework includes security tools, catalogs for managing datasets, and a secure data transfer service.

Most of the studies to date have been done on 30 Gbps networks or slower. Besides our own work, there was just one other study that showed 100 Gbps evaluations. The authors of [10] discuss the application issues and host tuning strategies for enabling applications to scale to 100 Gbps rates. They also demonstrated a climate data movement over 100 Gbps network at Super Computing 2011[17] and observed an average performance of 85 Gbps. The work presented in this paper shows that with the right middleware stack, as well as with careful tuning, a peak transfer rate of 99 Gbps can be achieved on a 100 Gbps network.

### III. PROPOSED WORK

As one of the three driving thrusts, the High Throughput Data Program (HTDP) at Fermilab aims to test the performance of end-to-end analysis systems of the stakeholders by identifying gaps in various tools used by researchers and ensure that they get corrected and function effectively at 100 Gbps scale.

In Grid Computing, Storage Element refers to a physical site hosting a storage infrastructure for storing and accessing data using standard interfaces with the required authorization policies. It also enables transfer of large amounts of data between grid jobs. The main services provided by the storage element are shown in **Figure 1**.

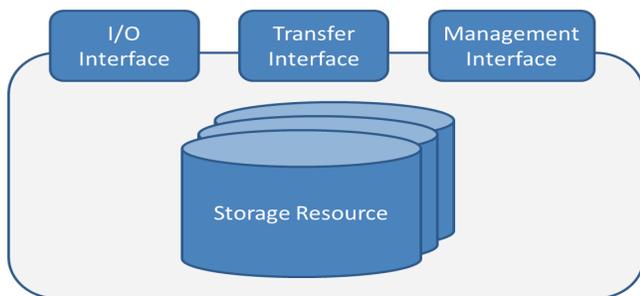


Figure 1: Storage Element

This paper discusses the various tests conducted to analyze the performance of different Grid middleware used by storage elements to identify the gaps that affect the throughput of data transfer on the 100 Gbps testbed. Here we

present an essential overview of the different middleware (GridFTP, Globus Online, Storage Resource Management, XrootD and Squid) that were tested for their scalability.

#### A. GridFTP

GridFTP [6][7][8] builds on top of FTP and is a high performance, secure, reliable data transfer protocol optimized for high bandwidth, wide-area networks. It provides a uniform way of accessing data. GridFTP adds an abstraction layer that encompasses different modes of access and exposes one single API for accessing data. FTP was chosen for its widespread use and for its well defined architecture for extensions to the protocol. Other features include

- *GSI Security for file transfers authentication/encryption*
- *Data channel reuse*
- *Third-party transfers: C can initiate transfers from A to B*
- *Parallel, Concurrent, Striped transfers*
- *Partial file transfer, Restart failed transfers*
- *Tunable network and I/O parameters*

In 2011, prior to the tests conducted on the 100 Gbps testbed, HTDP used the Long Island Metropolitan Area Network (LIMAN) testbed, a 30 Gbps ANI prototype, for testing GridFTP and Globus Online. The LIMAN testbed was one of the two fast network test environments connecting Brookhaven National Laboratory (BNL) and New York.

The goal of the evaluation was to maximize the utilization of the 30 Gbps network bandwidth using GridFTP and Globus Online. The throughput of the system is limited by the latency of the network prototype. The commands flowing on the control channel also add to the overhead for each file transferred over the network. This overhead becomes predominant when the size of file being transferred is small. Therefore the tests were conducted for three different datasets namely large, medium, and small files. The network utilization was low for small files when compared to large and medium files. With similar tests performed for Globus Online, the control channel latency was found to be higher than for the GridFTP tests. Also, the ANI testbed was accessible only through a Virtual Private Network (VPN) whereby a machine at Fermilab was designated as VPN gateway and implemented port forwarding mechanism to forward the accepted control port connections to the server machines. To have a fair comparison, the GridFTP client was also tested in the same condition by having it outside the network and handled using port forwarding.

Throughput for small files suffered when compared to large and medium files due to higher control channel overhead on per file basis combined with high latency. Globus Online auto-tuning appears to be more effective on medium files than on large ones.

The second fast-network environment was a full 100 Gbps network showcased at Super Computing 2011[17]. With an allocated bandwidth of 60 Gbps, the Grid & Cloud Computing Department of Fermilab, in collaboration with UCSD, demonstrated the transfer of 30 TB of CMS data in one-hour window with GridFTP, achieving a sustained rate of 66 Gbps.

### B. Storage Resource Management (SRM)

SRM [13][14] is a web service protocol operating over http and is the most common protocol for interfacing storage on the Grid. Its main purpose includes

1. Metadata operations
2. Data movement between storage elements
3. Generic management of backend storage

SRM is not designed for high throughput transfers, so it simply redirects the client to a transfer protocol such as GridFTP. This can also effectively load balance transfers over multiple nodes and thus shows good scalability. **Figure 2** shows the interaction between GridFTP/SRM client and the underlying storage system.

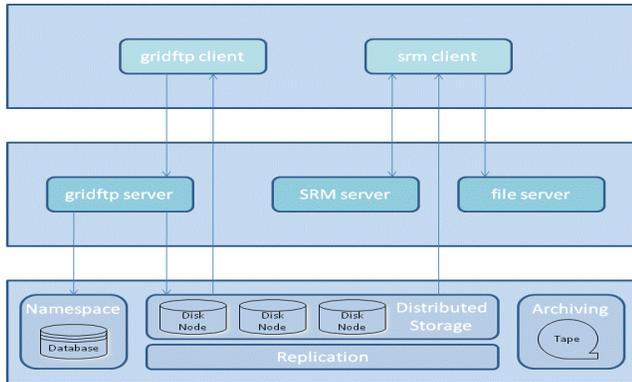


Figure 2: Storage Access by GridFTP and SRM client [14]

### C. XrootD

XrootD [15] is a popular data location and transfer tool used by the HEP community. XrootD provides the capability to manage a cluster of hosts as storage nodes as well as a basic data transfer protocol. The XrootD package consists of the storage server xrootd and the client tool xrscp.

### D. Squid

Squid [16] is a web proxy and cache that is used by the HEP community to reduce wide-area network traffic. Caching can improve response time through data reuse. It can also be used for load balancing web servers. It implements a Least Recently Used (LRU) algorithm to replace data items in cache.

## IV. PERFORMANCE EVALUATION

This section gives a brief description on the ANI 100 Gbps testbed. It then discusses the various experiments performed on this testbed using the Grid middleware explained in the previous section. All the tests were conducted to identify and correct the bottlenecks in the different applications and middleware used by the HEP community, so that they can function effectively at the 100 Gbps scale. All tests involved large data transfers and the throughput of these transfers was measured to evaluate the performance of the middleware.

### A. Testbed Description

The ANI 100 Gbps testbed has three sets of hosts: physical performance I/O servers, memory performance

servers, and virtual machines. Virtual machines were not involved in testing. **Figure 3** shows the network configuration of the testbed.

The machines at NERSC are disk performance servers and machines at ANL are memory performance servers. We had access to three machines at NERSC (nersc-diskpt-1, nersc-diskpt-2 and nersc-diskpt-3) and three machines at ANL (anl-memtp-1, anl-memtp-2 and anl-memtp-3). Each machine had four 10 Gbps interfaces through which it connected to the 100 Gbps cross country link. **Figure 4** illustrates the connection between the two sites and the subnet configuration of the network.

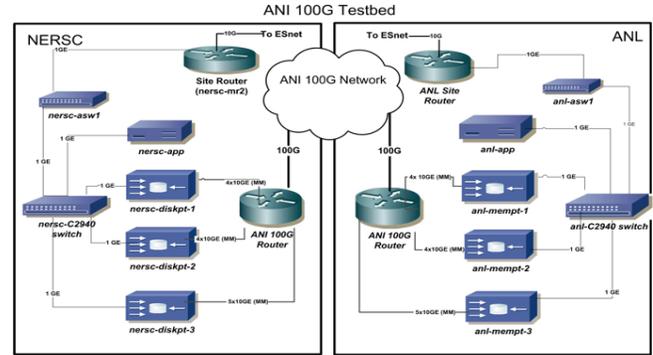


Figure 3: ANI 100 Gbps Test Bed [3]



Figure 4: Illustration of connections between NERSC and ANL nodes

ANL machines had two AMD 6140 Opteron Processor with 16 cores (8 cores each) at 2.6 GHz and 64GB memory.

NERSC machines 1 and 2 had Intel Xeon X5650 with 12 cores at 2.67 GHz and 48 GB of memory while NERSC 3 had Intel Xeon E5530 with 8 cores at 2.4 GHz and 24 GB of memory.

The round trip time between NERSC and ANL machines was measured to be 53 ms. **Figure 5** shows the round trip time (RTT) between the hosts involved.

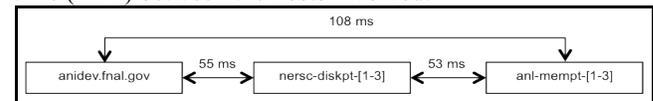


Figure 5: Testbed connections with RTT

10 Gbps interfaces on the performance hosts at NERSC and ANL are part of a 100 Gbps private network not

accessible from the public network. Each of these hosts also has one 1 Gbps interface connected to the respective site routers through a switch. Traffic to the 1 Gbps interfaces is fully routed within the ANI network and is used for port forwarding. Inbound access to the hosts is only available through a VPN. A machine at Fermilab, `anidev.fnal.gov`, provides access to the 100 Gbps testbed through the VPN. The VPN gateway ran the VPN software, accepted GridFTP control port connections from the Internet, and forwarded them to the network control interfaces (1 Gbps) of the three server machines, using `xinetd` port forwarding. In turn, again using `xinetd`, the three server machines forwarded those connections to their own GridFTP control ports, bound to the 10 Gbps interfaces.

### B. GridFTP and Globus Online

There were three different datasets used in the tests. In each dataset file size increased in powers of 2.

1. *Small* - 8KB to 4MB (Total: almost 8MB)
2. *Medium* - 8MB to 1GB (Total: almost 2GB)
3. *Large* - 2GB to 8GB (Total: 14GB)

All tests involved repeatedly transferring the datasets from NERSC hosts to ANL hosts. The operating system's RAM buffer cache was big enough to hold the entire dataset, so disk access was not a bottleneck. The GridFTP tests were done in three ways:

1. **Local Client-Server:** Each NERSC host invoked GridFTP client command to transfer files to every 10 Gbps interfaces of all the ANL hosts.
2. **Local Server-Server:** Third-party transfer of files from every 10 Gbps interface of each NERSC hosts to every 10 Gbps interface of all the ANL hosts.
3. **Remote Server-Server:** Same as local server-server but the client initiating third-party transfers was outside the VPN and used port forwarding.

Globus Online tests were similar to Remote server-server tests but the Globus Online machine initiated the third-party transfers.

TABLE 1: GRIDFTP & GLOBUS ONLINE PERFORMANCE MEASUREMENTS

Dataset	Local: Client-Server (Gbps)	Local: Server-Server (Gbps)	Remote: Server-Server (Gbps)	Globus Online (Gbps)
Large	87.92	92.74	91.19	62.90
Medium	76.90	90.94	81.79	28.49
Small	2.99	2.57	2.11	2.30

Table 1 and **Figure 6** summarize the throughput results of all the tests.

**Large and Medium Files:** With proper tuning of concurrency (`-cc 4`) and parallelism (`-p 4`) options, GridFTP performed well. Globus Online showed relatively lower performance. This is possibly due to higher control channel latency ( $> 150$ ms) between Globus Online servers and the testbed. There was also not much control over the tuning options compared to GridFTP.

**Small Files:** GridFTP performance was affected by the Lots Of Small Files (LOSF) problem. Although GridFTP Pipelining [11] is aimed at solving this problem, it did not

work as explained when transferring individual files. It works only when directories are transferred. **Figure 7** shows the GridFTP profile (throughput variation with file size from 1 MB to 1 GB).

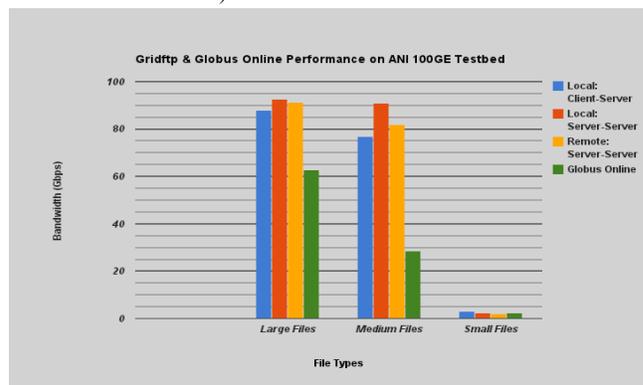


Figure 6: GridFTP & Globus Online performance comparison

At 100 Gbps speeds, medium is the new small. From **Figure 7**, it is quite clear that at higher network speeds, even the medium file sizes suffer from the LOSF problem, because of the problem with pipelining (see *3b* above).

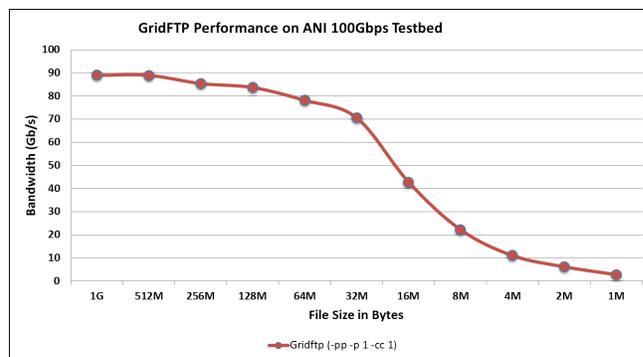


Figure 7: GridFTP Performance on ANI Testbed

To better understand the potential shortcomings for small file sizes, we analyzed the message flow of the GridFTP protocol over the data and control channels.

For a local server-server third-party transfer, there were two control channels and one data channel formed as shown in **Figure 8**.

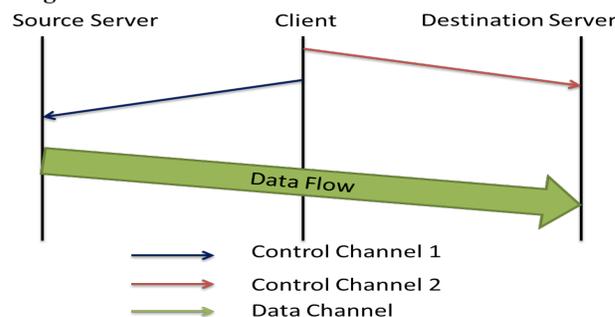


Figure 8: Control and Data Channel in Third-Party Transfer

Through tcpdump analysis, the series of commands that flow on control channel can be summarized as

1. Initially GSSAPI authentication takes place on both control channels.
2. Client sends a STOR command on control channel 2 to destination.
3. Client sends a RETR command to source server on control channel 1. Source server replies with Beginning transfer.
4. Destination server also replies Beginning transfer to the client and then sends the first Performance Marker.
5. Then data starts flowing on the data channel.
6. If the file is big enough, the destination server keeps sending Performance Marker and Range Marker every 5 seconds on the control channel 2.
7. When the source server has finished sending data, it sends Transfer Complete to client on control channel 1.
8. After receiving all data on data channel, the destination server sends the final Performance Marker, Range Marker and Transfer Complete status on the control channel 2.
9. If there are more files to send, then the process repeats from step 2.

Performance Marker is the instantaneous state of transfer indicating the number of bytes that have been transferred on a stripe at a given timestamp. Extended block mode (MODE E) uses Performance Marker to monitor the performance of data transfer.

Range Marker is the concatenation the number of bytes received on all stripes in a transfer. This can be used as Restart Marker if a client needs to restart the transfer of a particular range of data using the REST command. It is present to ensure backward compatibility with BLOCK mode.

The above analysis was done with the following options:

1. Pipelining (-pp)
2. One parallel stream (-p 1)
3. One connection (-cc 1)
4. No data channel authentication (-nodcau)

If the pipelining option (-pp) is removed, then the client sends SIZE command to source server and ALLO command to the destination server before sending the STOR command on the control channel 2. The source server replies with the size of the file. The destination server after receiving ALLO command sends back an "ALLO command successful" to the client. Then the above process continues from step 2. The complete flow diagram of above steps is shown in **Figure 9**.

A more abstract and high-level way of looking at the above steps is shown in **Figure 10** and **Figure 11**. **Figure 10** shows how an ideal third-party transfer should take place and **Figure 11** shows how it actually happened.

The throughput is limited by the STORE command that needs to be sent for every file only after receiving the Transfer Complete response for previous file sent on the same data channel. This causes a delay of one RTT between every file transfers. For small files, this wait time is larger than the actual time required for data transfer. When there are many small files, this overhead affects the throughput substantially. Thus we see a low throughput of 2.5 Gbps for

small files.

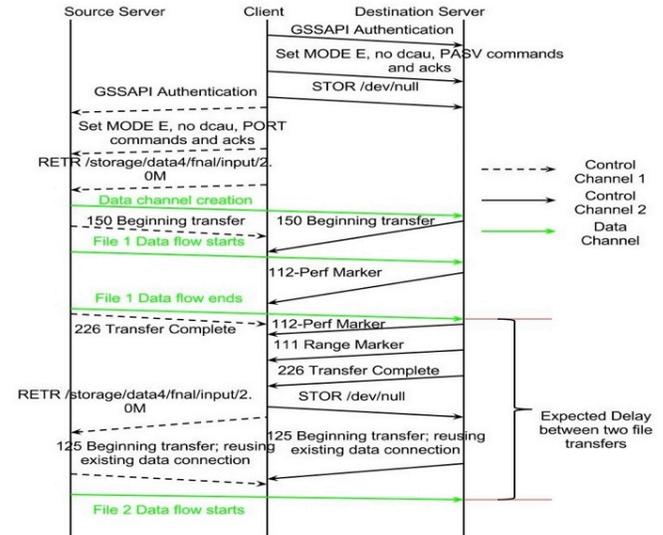


Figure 9: Command Flow Sequence in Third-Party Transfer

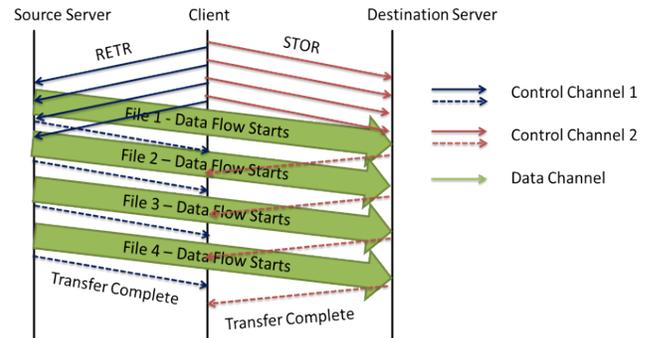


Figure 10: Ideal Third-Party Transfer

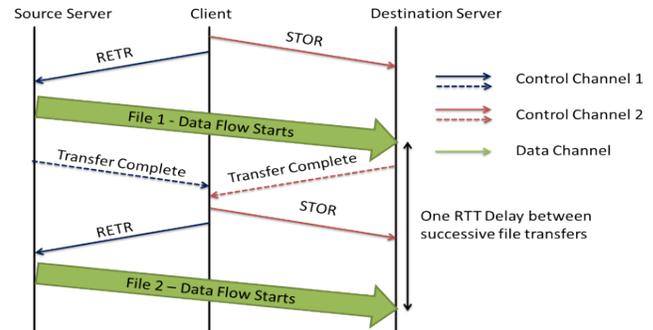


Figure 11: Actual Third-Party Transfer

### C. Storage Resource Management

The dataset used is same as that in GridFTP and Globus Online tests. Throughput measurements were made only for Local Server-Server Third-Party tests. The tests were conducted similarly to the GridFTP tests. There were no options available for controlling concurrency, but the option for multiple streams (parallelism) was enabled. **Table 2** and **Figure 12** show the throughput comparison of GridFTP, Globus Online and SRM Performance on the ANI Testbed.

TABLE 2: GRIDFTP, GLOBUS ONLINE AND SRM PERFORMANCE ON ANI TESTBED

Dataset	GridFTP (Gbps)	Globus Online (Gbps)	SRM (Gbps)
Large	92.74	62.9	87.36
Medium	90.94	28.49	77.15
Small	2.51	2.3	0.5

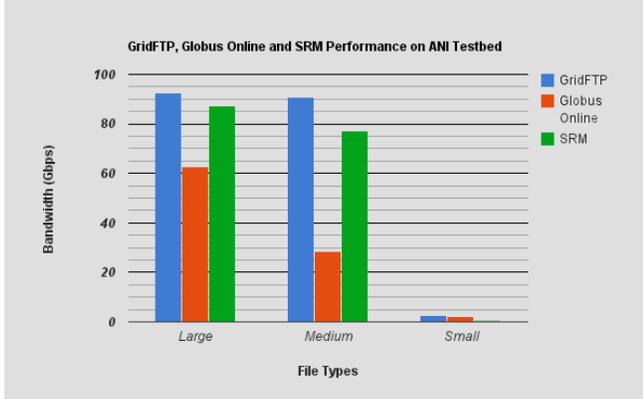


Figure 12 GridFTP, Globus Online and SRM Performance on ANI Testbed

SRM adds one more additional layer on top of GridFTP for transferring data. SRM has its own overhead, for example it needs to handle URL conversion (`srn://` to `gsiftp://`). Hence the throughput for data transfer using SRM is bounded by that of GridFTP. Further, it has no data channel caching, i.e. it closes data channel after every file transfer and opens a new data channel for next file transfer. This affects the throughput. This is clearly seen in the case of lots of small files as a reduction in performance.

#### D. XrootD

XrootD server supports concurrent and multi-stream transfers. Multiple clients were used in order to implement concurrency. Multi-stream transfers did not improve performance in our testing. Unlike GridFTP, xrootd does not allow writing data to a null device (`/dev/null`) and hence a RAM disk was used at the destination in order to avoid disk overhead. The size of RAM disk, therefore, becomes the bottleneck for the amount of data that can be transferred. Hence for file size greater than 2 GB aggregate throughput was estimated by scaling the results for one NIC. For other file sizes, a direct measurement was made.

From **Table 3**, XrootD server performed better with increased number of clients with almost 80% bandwidth utilization with 4 clients. As shown in **Table 4**, the scale factor was estimated using throughput measurements of small files (510 MB and 1 GB) for 1 NIC and 12 NICs (aggregate). This scale factor was then applied to files of sizes 4 GB and 8 GB.

As with GridFTP, XrootD shows poor performance for small files. **Figure 13** shows actual (for small and medium files) and scaled (for large files) throughput measurements for XrootD server.

TABLE 3: XROOTD PERFORMANCE (DIRECT MEASUREMENTS) ON ANI TESTBED

Dataset	1 Client (Gbps)	2 Clients (Gbps)	4 Clients (Gbps)	8 Clients (Gbps)
Large 1-NIC (8 GB)	3	5	7.9	N / A
Large, 1-NIC (2 / 4 GB)	2.3 – 2.7	3.5 – 4.4	5.6 – 6.9	7.7 – 8.7
Medium (64 MB / 256 MB)	2.9 – 8.8	5.7 – 14.7	11.2 – 23.9	22 – 39
Small (256 KB / 4 MB)	0.03 – 0.19	0.07 – 0.38	0.11 – 0.76	0.1 – 1.4

TABLE 4: AGGREGATE THROUGHPUT ESTIMATION USING A SCALE FACTOR FROM MEDIUM SIZE FILES

Dataset (GB)	1 NIC measurements (Gb/s)	Aggregate Measurements (12 NIC) (Gb/s)	Scale Factor per NIC	Aggregate estimate (12 NIC) (Gb/s)
0.512	4.5	46.9	0.87	–
1	6.2	62.4	0.83	–
4	8.7 (8 clients)	–	0.83	86.7
8	7.9 (4 clients)	–	0.83	78.7

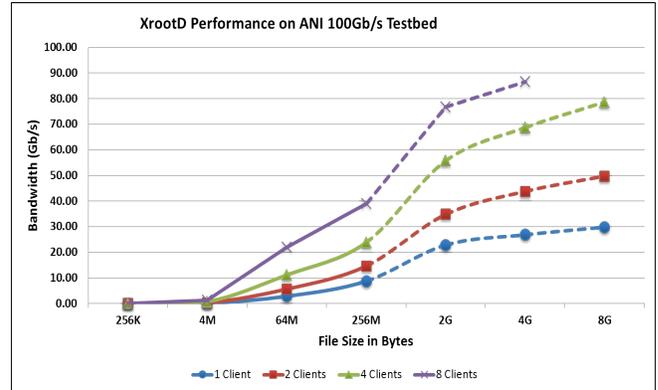


Figure 13: XrootD Performance on ANI Testbed

#### E. Squid

The testing of Squid is done by repeatedly fetching a 8MB file stored in a database at CERN using the `wget` command and using the Squid server as proxy. Only for the first request, the proxy fetches the file from the CERN database. All future requests for the same file are completed using the cached copy. Since the 8MB file is always in the file system buffers, disk access is not necessary. The parameter space in the tests involved was:

1. Transferring data in both directions (NERSC  $\leftrightarrow$  ANL)
2. Client to Server connection
  - a. One to One transfer - nersc-diskpt-1 client requesting 8MB file to anl-memtp-1 server, nersc-diskpt-2 to anl-memtp-2 and nersc-diskpt-3 to anl-memtp-3
  - b. All to All transfer - each diskpt client requesting 8MB file to all memtp servers
3. Number of Squid processes running in each host. We used squid2 that is single threaded, but ran multiple squid server processes listening on the same port for connections coming in to all 4 NICs.

4. With and without Core Affinity for Squid instances. Core affinity was enabled using the *taskset* command. Each instance of Squid was pinned to one core. Once a Squid instance is pinned to a core, it will always use the same caching layer. Core affinity can help minimize the number of L2 cache misses and in turn prevents the data being copied from one CPU to another.

In all-to-all transfers, for each NIC on the server side, there are 250 parallel processes running on the client side. There are 12 NICs in total on the server side. So there are 250 x 12 = 3000 clients per machine and three machines on the client side; therefore, there are 9000 clients in total. Each client repeatedly does wgets to fetch the 8MB file using the Squid server as http proxy.

In one-to-one transfers, each machine on the client end connects to only one machine on the server end. The number of NICs contacted is reduced from 12 to 4. So we increase the number of parallel processes from 250 to 750 per NIC to maintain the same total number of clients. The tests were also repeated for half the number of clients for some cases.

**Table 5, Table 6, Figure 14, and Figure 15** summarize the throughput results of the Squid tests. As we increase the number of Squid servers per host, the throughput increases in both directions. When there is lesser capacity at the server end, more clients overload the server thus bringing down the throughput; therefore, fewer clients maximize the utilization of that capacity. In addition, enabling core affinity increased the throughput.

**ANL to NERSC:** One-to-one gave lower performance than all-to-all. One possible reason could be that nersc-diskpt-3 is a slower machine and had one 10 Gbps NIC(eth2) with frequent problems. This decreased the utilization with more client load. Core affinity improved performance slightly, especially when there were higher numbers of Squid servers per host.

**NERSC to ANL:** One-to-one performance was almost the same as all-to-all. When the servers were running on the NERSC side, even the one bad NIC did not cause any issue

because the other three NICs were compensating by sending more data.

For both direction benchmarks, it is worthwhile noting that the core-affinity optimization was able to improve the overall performance by up to 21%. Furthermore, increasing the number of Squid servers per machine also improved aggregate performance, with some speeds approaching 100 Gbps.

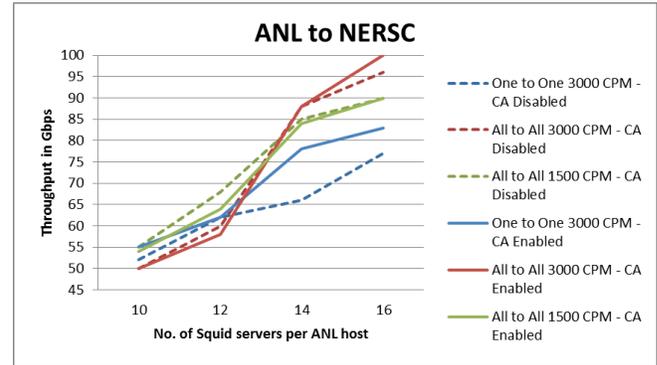


Figure 14: Squid Performance on ANI Testbed (ANL to NERSC) CPM – Clients per machine, CA – Core Affinity

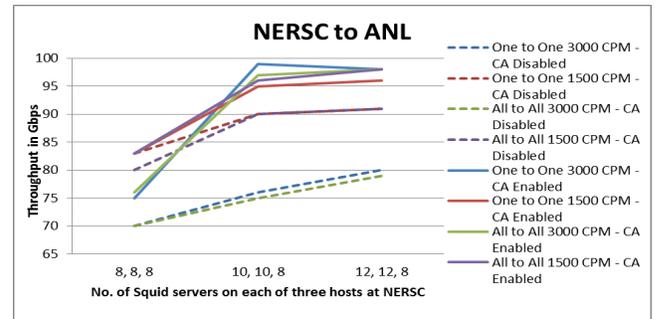


Figure 15: Squid Performance on ANI Testbed (NERSC to ANL) CPM – Clients per machine, CA – Core Affinity

TABLE 5: ANL TO NERSC

No. of Squid servers per ANL host	Core Affinity Disabled			Core Affinity Enabled		
	One to One 3000 clients per machine (Gbps)	All to All (Gbps)		One to One 3000 clients per machine (Gbps)	All to All (Gbps)	
		3000 clients per machine	1500 clients per machine		3000 clients per machine	1500 clients per machine
10	52	50	55	55	50	54
12	62	60	68	62	58	64
14	66	88	85	78	88	84
16	77	96	90	83	100	90

TABLE 6: NERSC TO ANL

No. of Squid servers on each of 3 hosts at NERSC	Core Affinity Disabled				Core Affinity Enabled			
	One to One (Gbps)		All to All (Gbps)		One to One (Gbps)		All to All (Gbps)	
	3000 clients per machine	1500 clients per machine	3000 clients per machine	1500 clients per machine	3000 clients per machine	1500 clients per machine	3000 clients per machine	1500 clients per machine
8, 8, 8	70	83	70	80	75	83	76	83
10, 10, 8	76	90	75	90	99	95	97	96
12, 12, 8	80	91	79	91	98	96	98	98

## V. CONCLUSION AND FUTURE WORK

The purpose of the High Throughput Data Program at Fermilab is to prepare Fermilab and its stakeholders to transition to the 100 Gbps network backbone by evaluating different layers and services involved in end-to-end analysis systems. The middleware components tested include GridFTP, Globus Online, SRM, XrootD and Squid.

The analysis of results indicates that the different middleware technologies have the potential and could scale up to 100 Gbps in certain cases. It also reveals the cases where the middleware performs poorly and the features that need to be improved for better performance.

In GridFTP, to improve the performance for small and medium files, implementation of “pipelining” needs to support moving “individual” files between two GridFTP servers, rather than directories only.

In Globus Online, the control channel latency is very high, since the tests involved forwarding the commands in the control channel to the GridFTP servers through VPN. Another factor that might be responsible for lower performance is that there are fewer available tuning options as compared to GridFTP.

In SRM, since it uses the GridFTP protocol internally for transferring data, the performance is limited by the GridFTP implementation. Besides, its own implementation adds some overhead that causes lower performance than GridFTP. Finally the absence of Data Channel Caching shows its impact in the case of small files, where the throughput attained is much less than that of GridFTP.

XrootD performance for small files was poor, similarly to GridFTP, but has the potential to scaling up to over 85 Gbps with sufficient number of clients.

Squid utilized the full throughput of the testbed reaching nearly full bandwidth of 100 Gbps, when 3000 clients on ANL performed the All-to-All requests to NERSC servers. Most of the other cases also showed good throughput. The tuning strategy consists in having the right number of clients in different settings so as not to overload the server. When the number of servers is less than the available CPU capacity, this number forms the bottleneck. When there were sufficient servers, the number of parallel clients decides the throughput.

As a part of future work, we plan to test other technologies used in Grid Computing, such as CVMFS [18], iRODS [19], dCache [20], NFS v4.1 and Lustre [21]. The list is mainly driven by the stakeholders needs.

## VI. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant No. 1007115, ExTENCI: Extending Science Through Enhanced National Cyber Infrastructure. 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 is supported by the U.S. Department of Energy under contract No. DE- AC02-07CH11359. We thank the Globus Team for their availability and support.

## REFERENCES

- [1] Energy Sciences Network. Accessed on Jun 1, 2012. <http://www.es.net>
- [2] The U.S. Department of Energy. Accessed on Jun 1, 2012. <http://energy.gov>
- [3] The Advanced Networking Initiative. Accessed on Jun 1, 2012. <http://www.es.net/RandD/advanced-networking-initiative/>.
- [4] F. Bachmann et al, “XSEDE Architecture – Level 1 and 2 Decomposition” Feb 21, 2012. White paper. <https://www.xsede.org/>
- [5] R.Pordes et al. “The Open Science Grid”. In Proceedings of the CHEP’04 Conference, Interlaken, Switzerland, September 27th - October 1st, 2004 2004. Published on InDiCo.
- [6] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, The Globus Striped GridFTP Framework and Server, in Proc. of the 2005 ACM/IEEE conference on Supercomputing, pp.54-64, Seattle, Washington USA, November 2005.
- [7] Lessons learned from moving Earth System Grid data sets over a 20 Gbps wide-area network, Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC 2010), June 2010.
- [8] Haifeng Pi 2010 High Throughput WAN Data Transfer with Hadoop-based Storage (PS23-4-071) CHEP 2010, October 18-22, 2010 Taipei, Taiwan
- [9] A Data Management Framework for Distributed Biomedical Research Environments, IEEE eScience Workshop on High-Performance Computing in the Life Sciences, Australia, Dec 2010
- [10] Experiences with 100Gbps Network Applications, Proceedings of the fifth international workshop on Data-Intensive Distributed Computing, 2012
- [11] GridFTP Pipelining, John Bresnahan, Michael Link, Rajkumar Kettimuthu, Dan Fraser and Ian Foster, Proceedings of the 2007 TeraGrid Conference, June 2007
- [12] Foster, I. Globus Online: Accelerating and democratizing science through cloud-based services. IEEE Internet Computing(May/June):70-73, 2011.
- [13] Storage Resource Managers: Essential Components for the Grid, A. Shoshani, A. Sim, J. Gu, 2003
- [14] Storage Infrastructure Software. Accessed on Jun 1, 2012. <https://www.opensciencegrid.org/bin/view/Documentation/StorageInfrastructureSoftware>
- [15] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky. XROOTD-A Highly scalable architecture for data access. WSEAS Transactions on Computers, 1(4.3), 2005.
- [16] D. Wessels, Squid: The Definitive Guide, O’Reilly & Associates, Inc. Sebastopol, CA, USA ©2004, ISBN:0596001622
- [17] SuperComputing 2011. Accessed on Jun 1, 2012. <http://sc11.supercomputing.org/>
- [18] P Buncic et al, CernVM – a virtual software appliance for LHC applications, 2010 J. Phys.: Conf. Ser. 219 042003
- [19] Rajasekar, A., Moore, R., Hou, C.-Y., Lee, C.A., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S.-Y., Gilbert, L., Tooby, P. and Zhu, B. iRODS Primer: Integrated Rule-Oriented Data System. Morgan and Claypool Publishers, 2010.
- [20] P. Fuhrmann. dCache: the commodity cache. In Twelfth NASA Goddard and Twenty First IEEE Conference on Mass Storage Systems and Technologies, Washington DC, Spring 2004.
- [21] Philip Schawn, Cluster File Systems, Inc., Lustre: Building a File System for 1,000-node Clusters, 2003.