DMOVER: Parallel Data Migration for Mainstream Users

Nathan T.B. Stone, Bryon Gill, John Kochmar, Rob Light,
Paul Nowoczynski, J. Ray Scott, Jason Sommerfield, Chad Vizino
{stone, bgill, kochmar, light, pauln, scott, jasons, vizino}@psc.edu
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh, PA 15213

Abstract

DMOVER is a combination of three highly portable scripts and a network optimization specific to the Pittsburgh Supercomputing Center (PSC). DMOVER provides users with a queue-oriented means for initiating large, parallel, inter-site data transfers that is expected to be familiar to all mainstream HPC users. Its flexibility enables users to effectively manage a parallel, load-balanced transfer of hundreds of files between dozens of file servers. It provided these highly-desirable features at a time when they were not present in other standardized grid services and which remain elusive on some platforms. Performance measurements consistently indicate that file transfers remain disk-rate-limited and that the DMOVER strategy of maintaining parallel streams is a highly efficient means for aggregating available resources.

1. Introduction

The Extensible Teragrid Facility[1] (ETF) is a massive collection of high-performance computing, networking and storage resources woven together by a software infrastructure designed to extend its functionality and ease of use to the current generation of scientific researchers. This substantial task has turned out an interconnected web of authentication & authorization, data transfer, job management and accounting software that can still confuse even informed users. Many developers within the ETF community have rededicated themselves to eliminating this confusion, redesigning some interfaces or providing additional tools to ease the learning curve for users in the hopes that this will deliver the promises of grid computing to mainstream users. "DMOVER"[2] is one such effort at the Pittsburgh Supercomputing Center (PSC) directed at facilitating large, inter-site, parallel data transfers.

1.1. Problem Summary

Data migration is a recurring challenge for most ETF users. Despite the presence of a high-performance network and file transfer middleware, moving thousands or even hundreds of files across the ETF network can still be a slow or tedious process for mainstream users – those with a lower tolerance for instability or infrastructural complexity. Such users are perceived to be dependent upon resources and utilities delivering exceptional performance (in this case, aggregate bandwidth) exactly as advertised every time. This is an obvious challenge to the resource administrators and middleware and system software developers.

Furthermore, a recent survey of Hierarchical Storage Management (HSM) utilization at PSC revealed that over 93% of data stored in our HSM are transferred in "sessions" involving 10 or more files. In such sessions the average number of files was 378, with an average file size of 93 MB. So if we are to adequately address the problem of transferring real datasets it must be in a context of facilitating transfers with large file count and moderately large files.

1.2. Solution Summary

DMOVER[2] is fundamentally an integration effort unifying three readily available resources on "LeMieux"[3], PSC's 3000+ processor ETF computing resource. First, DMOVER uses the batch scheduling system to allocate both dedicated file servers and Application GateWay (AGW) nodes (see Section 3.4). Second, it uses the standardized grid file transfer tools (*e.g.* `globus-url-copy`) to transfer files resident within LeMieux's global parallel scratch file systems. And third, it uses a transparent communication library called Qsockets to redirect TCP-based traffic from the default Ethernet network to the Quadrics QSNet-based computational interconnect and to the ETF network via the AGW nodes. DMOVER executes these transfers in parallel, managing a user-definable number of transfer streams until all designated files are transferred.

2. Problem Details

At PSC we were motivated by the expressed concerns of a particular user who had a clearly specified need: to transfer terabytes of data from PSC to the San Diego Supercomputing Center (SDSC) in a timely manner. (Transfer rates at the time, he complained, were typically of order 1-3 MB/s—more characteristic of an `scp` client than of any high-performance tool). If we failed to establish an effective channel then this user would simply not compute at our site. So we were motivated to write our own solution both to raise the performance and to "lower the bar", making it likely that this user would use our solution in this case and perhaps others.

We first surveyed the grid middleware tools at our disposal, listed following.

- The `globus-url-copy`[4,5] client even in recursive mode was clearly inadequate, due to the observed deficiency of single-stream bandwidth and the high file count involved. Other grid data transfer clients (*e.g.* `gsincftp`), while providing a slightly different interface, deliver no better performance.
- "Striped GridFTP"[6,7], a protocol for sending parts of a single file in parallel via multiple 3rd-party transfers, was not yet available at that time. At present it is still not available for Tru64 UNIX due to a reliance upon the Globus Toolkit V4.0[8], which does not exist yet for that OS. Nevertheless, the unsuitability of this protocol to the transfer of large file counts is confirmed by the command line switch required to activate it ("`-big`", pertaining to file size).
- The `uberftp`[9] client supports a parallel mode which is expected to be effective for 3rd-party transfers but of limited usefulness for direct transfers (client-based upload/download). The parallel streams in this case are implemented in such a way that each file is divided in <N> parts (where <N> is specified on the command line) but all transfers terminate at the client's host. This inherently limits the total aggregate bandwidth to the network connection of the client host, although it may benefit cases in which single-stream bandwidth has lower hard

limits.

- Reliable File Transfer[10,11] (RFT) has also been only recently deployed. It is generally well-suited to the parallel transfer of large numbers of files, and the administrator-configured number of parallel streams by default is of order 12 to 15. However, it relies on the presence of 3rd-party GridFTP servers for accessing all of the storage in question. As noted above, this alternative is unavailable within LeMieux, since it is running Tru64 UNIX.

If users are forced to write their own "helper scripts" or other tools in order to use our middleware they will simply not adopt it.

3. Solution Details

We resolved to build a light-weight parallel aggregation layer over `globus-url-copy`, to at least utilize its authentication scheme and tunable options. Our aggregator, now known as DMOVER, incorporated the following components at the PSC site:

1. Parallel file servers: LeMieux contains a pool of 64 file servers that export a global parallel file system. At present 16 of these are dedicated to a reserved "DMOVER" queue for allocation by the scheduling system.
2. QSNet[12]: LeMieux is interconnected by a high-performance Quadrics network called "QSNet", which routinely delivers DMA transfer rates of order 250 MB/s per NIC. The Application GateWay (AGW) nodes, HSM cache nodes, and Rachel (rachel.psc.edu) are also on the PSC QSNet.
3. Application GateWay (AGW) nodes[13]: PSC maintains a scalable number (currently 16) AGW nodes dedicated to relaying network traffic between the ETF network and PSC's internal QSNet. Each node has interfaces on the ETF network, the PSC LAN, and the PSC QSNet. Because there are a finite number of AGW nodes, we have configured these nodes are a scheduled resource, allocated by our batch scheduling system.
4. Qsockets[14]: Developers in PSC's networking group have created the "Qsockets" system, which unites a Qsockets server that serves as a data router and a client process via the QSNet. Qsockets provides a client library that intercepts normal TCP socket function invocations in legacy binaries running on LeMieux compute nodes and relays them over the QSNet to the Qsockets server resident on an AGW node. The server then passes the traffic on to the ETF network or whatever external network is most appropriate. In this way processes within LeMieux gain virtual interfaces to the ETF network.

Our solution requires a user to do three things:

1. Acquire the proper grid credentials (no different than for normal grid operations)
2. Specify source and destination by changing a few (4-6) lines in a batch script
3. Queue a job to a dedicated DMOVER queue (via the same queuing system with which they are already familiar)

The scheduling system itself allocates the requisite number of AGW nodes for parallel, multi-stream transfers. The DMOVER suite expands the source specification into a list of target files, builds the command lines, and launches multiple transfer streams in parallel, keeping a fixed number of streams running through the AGWs at all times until all transfers are completed. Below are details describing each of the components.

DMOVER consists of a trio of scripts: a batch script, a process manager and a transfer agent (command line wrapper). This solution required no modifications or additions to any other existing elements at any site.

3.1. The Batch Script

The first script in the trio is the DMOVER batch script (*e.g.* `dmover.pbs`). A user submits this script to the scheduler (OpenPBS) to initiate a transfer. The advantage of this approach is that batch scripts are highly familiar to all HPC users. This eliminates the usage barrier of interface style. Figure 1 below shows a sample job script that a user could queue to initiate DMOVER transfers to SDSC via 4 parallel streams.

```
#PBS -l rmsnodes=4:4
#PBS -l agw_nodes=4

# root of the file(s)/directory(s) to transfer
export SrcDirRoot=$SCRATCH/mydata/

# path to the target sources, relative to SrcDirRoot
export SrcRelPath="*.dat"

# destination host name
export DestHost=tg-c001.sdsc.teragrid.org,
tg-c002.sdsc.teragrid.org,tg-c003.sdsc.teragrid.org,
tg-c004.sdsc.teragrid.org

# root of the file(s)/directory(s) at the other side
export DestDirRoot=/gpfs/ux123456/mydata/

# run the process manager
/scratcha1/dmover/dmover_process_manager.pl
"$SrcDirRoot" "$SrcRelPath" "$DestHost" "$DestDirRoot"
"$RMS_NODES"
```

Figure 1: Sample DMOVER job script.

Number of Streams

Note first that the user specifies the number of streams for the transfer by setting both the number of nodes to use from the dedicated pool of file servers (`rmsnodes=4:4`) and the number of AGW nodes (`agw_nodes=4`). These directives are for the scheduling system, thus are prefixed with the "`#PBS -l`" resource specification. Per-stream performance is optimal when using one process per node (*e.g.* "`rmsnodes=4:4`") as compared to sending multiple streams from the same file server (*e.g.* "`rmsnodes=1:4`", representing a request for 1 node on which 4 processes will run). Furthermore, performance is best when the number of AGW nodes reserved matches the number of transfer streams. (See Section 5 below for a more thorough discussion of number of streams and performance.)

Source Path
The "`SrcDirRoot`" specifier identifies the absolute root path to the files to be transferred. This is a convenience that may or may not help the user in specifying "`SrcRelPath`". The "`SrcRelPath`" variable specifies the files and/or directories to be transferred relative to "`SrcDirRoot`" by name or by wildcard. In the case where there is a single directory tree to be transferred this is a straightforward process. But the value of `SrcRelPath` is expanded in the DMOVER process manager script by UNIX glob, which may in general match a large number of files and/or directory trees. The net result is that the process manager converts this information into a list of file targets to be transferred.

Destination Host
The destination is specified as a hostname string. In general that string can be a comma-separated list of valid hostnames. This was preferable over using some DNS-based host aliases because earlier versions of `globus-url-copy` would abort transfers in the case where the `gethostbyname()` invocation returned a server that was down as the first entry in the IP list. By allowing users to specify only servers that are up and available we enabled them to avoid known potential problems if they so choose.

This understated feature is of extreme relevance to bridging the gap between "early-adopters" and "mainstream users". There is often a disconnect between the tools used by these respective groups – one arcane version for early adopters and one "simple" version for mainstream users. The latter is often "simpler" because it is completely different, and many of the control interfaces are removed or obscured.

In this case we provide one single field for specifying a destination host. It can either be something "simple", like a single DNS-based host alias (*e.g.* `tg-gridftp.psc.teragrid.org`) or it could be a list of selected server names (*e.g.* `tg-c001.sdsc.teragrid.org, tg-c002.sdsc.teragrid.org,`…). But in either case the vehicle is the same, and the user can specify as little or as much detail as desired. This strategy effectively bridges the complexity gap by allowing either case, growing with the users as they grow their experience.

The next script level, the DMOVER process manager, will use a round-robin scheme to utilize all of the available hosts in turn for each successive transfer, thus achieving a basic level of load-balanced parallelism.

3.2. DMOVER Process Manager

As noted above, the process manager (`dmover_process_manager.pl`) parses and expands the values of the source and destination variables the user specifies in the batch script, building a list of specific transfers from these values. It then spawns an independent transfer for each case, launching a fixed number of transfers in parallel.

As earlier transfers finish the process manager selects another transfer from the list of remaining targets and starts it. Although the process manager itself runs on one of the OpenPBS “mom” nodes (currently configured as the LeMieux login nodes) the transfers are initiated remotely on the dedicated file servers allocated to the DMOVER queue.

This mode of launching many parallel transfers simultaneously is precisely what many users have been missing. This is contrasted with striped transfer, which is typically only advantageous for small numbers of very large files. To clarify, if all other transfer characteristics are equal, parallel streams will generally out-perform striped streams for cases where the number of files is much larger than the number of servers. And, as noted in the introduction, the average number of files transferred in a typical “session” is an order of magnitude greater than the number of file servers available at any site.

3.3. DMOVER Transfer Agent

Each of the parallel transfers is a single execution of the DMOVER transfer agent (`dmover_transfer.sh`). The transfer agent merely executes the Globus transfer client of choice (currently `globus-url-copy`). It uses the source and destination arguments supplied by the process manager and supplies these and other proper defaults to the file transfer client. But before it executes the Globus transfer it modifies the LD preload path (`_RLD_LIST`) to allow the otherwise “normal” TCP socket operations of the transfer client to be redirected over Qsockets, described in detail below. Using this technique we have benchmarked single-stream memory-only transfers across the ETF using `globus-url-copy` at roughly 1 Gb/s, or the theoretical maximum per-stream bandwidth for the hosts available.

The encapsulation of the final execution command line in the transfer agent script also provides a place for advanced users to optimize their transfer. For example users could select an alternative grid file transfer client (*e.g.* `gsincftp`) or customize their command line parameters to suit their needs by either modifying the provided script or substituting their own in the process manager.

3.4. AGWs and Qsockets on LeMieux

“LeMieux” is the largest compute resource on the ETF today. Its architecture leverages the high speed QSNet interconnect between nodes, but does not inherently provide for large bandwidths to remote hosts. Instead of connecting each compute node to the ETF network directly our design team introduced the concept of Application GateWay (AGW) nodes.

AGW nodes have network interfaces both on the ETF network (2 x 1 GigE NICs per node) and on the PSC QSNet network (1 QSNet NIC per node). The two-to-one matching between GigE and QSNet NICs led to a natural division of each AGW node into two “virtual” nodes, one serving each GigE interface. Each virtual node is equipped with a Qsockets “Qserver” that routes traffic from the PSC QSNet network (*e.g.* from LeMieux) to the ETF in a seamless manner. This configuration is so successful that an AGW node has no difficulty filling each of its GigE pipes from the QSNet, as described below. On the LeMieux side, Qsockets provides a client library that intercepts TCPrelated system functions. In the case of setup/tear-down or ioctl functions, Qsockets merely passes these to the AGWs as requests over a QSNet-based RPC protocol. In the case of send operations the data is transferred over the QSNet to the AGW and from there routed to the ETF (and the converse for receive operations). In this manner processes running on LeMieux compute nodes can behave programmatically just as though they were directly connected to the ETF. This is all achieved without modifying any source or compiled code in the target application.

The efficiency and effectiveness of this strategy were proven during the “Bandwidth Challenge” during the SC’04 conference in November of 2004. In preparation for SC’04, PSC established a team whose purpose it was to design an end-to-end science demonstration suitable for the Bandwidth Challenge. The strategy of this team was to use a real scientific application that was instrumented to write its checkpoint data to a remote file server over a TCP connection. That much of the application had been created years prior as part of a simple checkpoint-recovery demonstration. For the Bandwidth Challenge, this application was redirected to transmit data not merely over Ethernet but over a combination of the QSNet and a custom 40 Gb/s link to the SC’04 show floor without changing any networking code in the application, but merely by setting the LD preload path as described above. Using this configuration that team sustained a write bandwidth of over 31.1 Gb/s end-to-end: from 32 LeMieux compute nodes, over 16 QSNet / dual GigE-connected AGW nodes, to 7 10GigE-connected file servers on the SC’04 show floor. This clearly demonstrates the high performance (high aggregate bandwidth) and low overhead (31.1 Gb/s over 32 GigE links on only 16 nodes) of the Qserver routing protocol on the AGW nodes.

4. Portability

The queue and process elements described above are completely portable. Every super-computing site has a scheduling system that can facilitate remote execution of multiple transmission streams on specialized nodes. Furthermore, the DMOVER scripts are written in Perl and Bourne shell, which is certainly accessible at all high-performance computing sites. The only non-portable aspect of this system is the AGW/Qsockets layer, which is a custom interconnect feature. Other sites have addressed their networking needs differently, for example by connecting compute nodes directly to the WAN. So the AGW/Qsockets layer does not require portability. If the DMOVER scripts were run at other sites mainstream users could simply comment out the AGW/Qsockets-related lines and run as at PSC.

Whether users would want to import DMOVER to a non-PSC site remains a valid question. Without DMOVER the only parallelism available to mainstream users is the number of GridFTP servers that have been configured by the site administrators. With DMOVER users have only to queue a DMOVER transfer job to a valid execution queue and they can utilize as many parallel streams as are available on that compute resource. This constitutes a perfectly reasonable utilization of HPC resources and a mainstream user’s compute allocation.

5. Performance

Below (Figures 2a and 2b) are performance measurements recorded for DMOVER transfers involving 32 files of size 2 GB (blue diamonds), 100 files of size 200 MB (red squares) and 300 files of size 100 MB (green triangles) from PSC (`/scratchb2`) to SDSC (`/gpfs`). These represent two reasonable test cases and a typical user storage scenario (as noted in Section 1.1), respectively. The transfers were performed varying the number of streams, and the bandwidths are reported as a function of number of streams. Figure 2a is a plot of the average per-stream bandwidth and Figure 2b is a plot of the bandwidth aggregated over all streams. At the time of testing the compute resource LeMieux was fully loaded. Thus the values represent a typical case for mainstream user activity. Indeed these results are consistent with lower values observed in other more optimal circumstances.

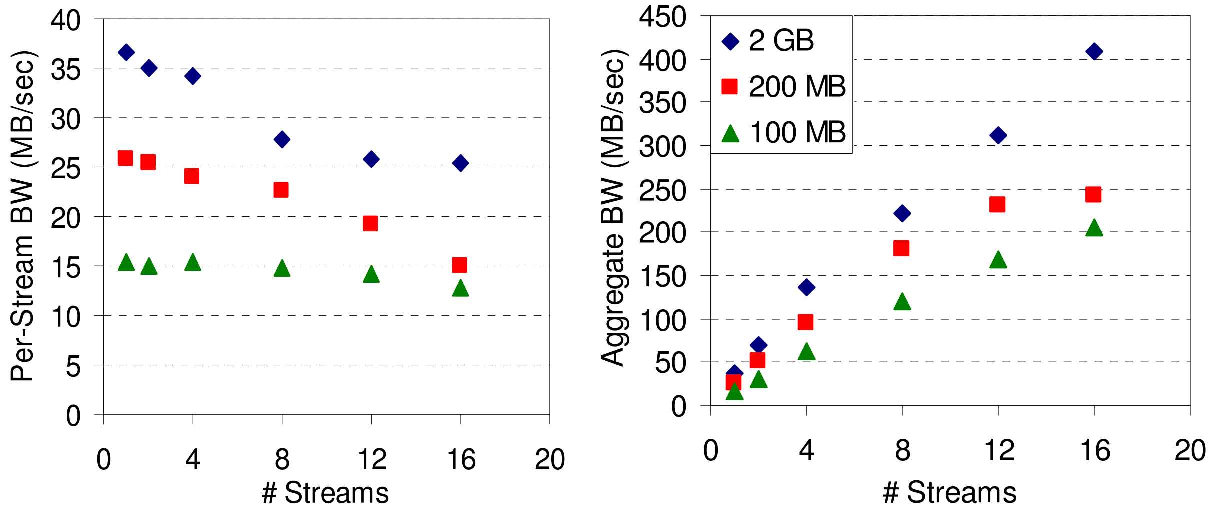

Figure 2: a) Average, per-stream bandwidth and b) aggregate bandwidth of DMOVER transfers for three file sizes, as labeled.

First we observe the trends in the data. In all cases we observe that the per-stream bandwidth for larger files is greater than that for smaller files. In fact, our 2GB test case achieves roughly double the bandwidth of the 100MB “average user” case. This exposes the extent to which the protocol overhead requires longer transfers to amortize the per-stream setup cost. We further observe that the per-stream bandwidth decreases with increasing number of streams. Thus the upward pressure to minimize aggregate transfer times is clearly at odds with the downward pressure of the scalability of the parallel transfers.

The aggregate bandwidth measurements are perhaps more encouraging. Even for the highest number of streams shown we do not appear to have reached a plateau or a point of diminishing returns for increasing stream count, despite the loss in efficiency. So we clearly demonstrate that users can easily achieve several hundred MB/sec aggregate with DMOVER by selecting the appropriate number of streams for their file size.

We continue by analyzing the host and network configuration underlying these results. At the sending side the number of hosts and associated GigE interfaces was matched to the number of streams, up to a maximum of 16. At the receiving side, the maximum was 12. Since the highest per-stream bandwidths observed up to 12 streams represent only 20-30% of the available network bandwidth to each host, the limitation is clearly not in the network. Furthermore, since (at least for cases with 12 or fewer streams) the transfer clients at each end are running on separate hosts we know that the processes cannot be adversely impacting one another. And yet the observed per-stream bandwidth clearly decreases with increasing stream count, even below 12. We conclude that the predominant bottleneck is in the parallel file systems at one or each end. To isolate this further (*e.g.* identify the limitations at each end) we would have to use the same transfer tool (`globus-url-copy`) to run device-only copies (*e.g.* from `/dev/zero` to `/dev/null`). This feature is not yet supported by `globus-url-copy` for `file://` type transfers. Device-only transfers between servers, however, have been benchmarked in excess of 110 MB/sec—wire speed for GigE-connected hosts.

Various groups have created ongoing ETF network and GridFTP diagnostics that continually monitor the performance of the network. One such effort is the PSC "SpeedPage".[15] This tool has indeed measured point-to-point single-stream bandwidths in excess of 90 MB/sec for some sites and file systems. This further supports the conclusion that investment in high-performance parallel file systems will make the greatest difference in inter-site GridFTP transfers.

6. User Experience

While the SC'04 Bandwidth Challenge well-established the efficiency and effectiveness of Qsockets for routing high throughput traffic from legacy applications, this is not sufficient to demonstrate its usefulness in the context of grid computing. The ultimate measure of the success of grid computing is the number of users and applications that use it to succeed in tasks that they could not have otherwise accomplished. The original incentive to create the DMOVER framework was the expressed need of an experienced user who wanted to migrate large amounts of data (of order Terabytes) in large numbers of files (of order thousands) from PSC to SDSC. Furthermore, he wanted to do this with regularity – not merely once or as a proof of concept. This motivated us to create an infrastructure to achieve substantial throughput from end-to-end between file servers at opposite ends of the country using Globus tools.

The resulting product is DMOVER, as described above. After using it himself our target user observed that this tool got him "past the Globus roadblock". He transferred Terabytes of data from PSC to SDSC at roughly 200 MB/s aggregate. So the parallelization model of DMOVER was precisely what enabled this user to achieve this level of aggregate performance.

7. Conclusions

DMOVER is a tool for initiating load-balanced parallel file transfers between sites on the ETF network. It achieves scalable aggregation of parallel streams thereby achieving a high effective bandwidth for large file-count transfers. Furthermore, this file-wise mode of parallelism is expected to be most applicable to typical user scenarios within the ETF community. DMOVER is an invaluable bridge between the raw grid services and the needs of mainstream users. Although the Globus ToolKit (GTK) incorporates similar functionality in RFT[10] even that utility is limited by administrative configurations (having sufficient GridFTP servers running wherever your data is staged) and code portability issues (GTK is not available on all platforms). If there are few or no servers running in such a location that they can effectively serve the file system of interest then mainstream users would be forced to resort to some other file transport system like DMOVER. Fortunately, with the portability and simplicity of DMOVER this should remain a viable option for mainstream users for the foreseeable future.

References

[1] The ETF (a.k.a. "TeraGrid") Project: http://www.teragrid.org/

[2] Usage documentation online at http://teragrid.psc.edu/lemieux/jobs.html#dmover

[3] Machine documentation online at http://www.psc.edu/machines/tcs/lemieux.html

[4] "Data Management and Transfer in High Performance Computational Grid Environments" B. Allcock, et al. *Parallel Computing Journa*l, Vol. 28 (5), May2002, pp. 749-771.

[5] Online documentation at http://www-unix.globus.org/grid_software/data/globus-url-copy.php

[6] "The Globus Striped GridFTP Framework and Server" Bill Allcock, John Bresnahan, Raj Kettimuthu, Mike Link, Catalin Dumitrescu, Ioan Raicu, Ian Foster. *Submitted to the 2005 High Performance Distributed Computing Conference (HPDC 14)*

[7] Online documentation at http://www-unix.globus.org/grid_software/data/stripedftp.php

[8] Private communication: Raj Kettimuthu of Argonne National Laboratory

[9] Software and online documentation at http://dims.ncsa.uiuc.edu/set/uberftp/

[10] "Reliable Data Transport: A Critical Service for the Grid" W.E. Allcock, I. Foster, R. Madduri. *Building Service Based Grids Workshop, Global Grid Forum 11*, June 2004

[11] Online documentation at http://www-unix.globus.org/grid_software/data/rft.php

[12] Product description at http://www.quadrics.com/Quadrics/QuadricsHome.nsf /DisplayPages/3A912204F260613680256DD9005122C7

[13] User documentation online at http://teragrid.psc.edu/lemieux/jobs.html#agw

[14] "Qsockets" M. Heffner, CMU Technical Report CMU-PSC-TR-2004-01, 2004 (online at http://www.psc.edu/publications/tech_reports/qsockets/qsockets.html)

[15] GridFTP performance monitoring at http://gridinfo.psc.edu/gridftp/speedpage.php