A Classification and Evaluation of Data Movement Technologies for the Delivery of Highly Voluminous Scientific Data Products

Chris A. Mattmann, Sean Kelly, Daniel J. Crichton, J. Steven Hughes,
Sean Hardman, Paul Ramirez and Ron Joyner
Jet Propulsion Laboratory
California Institute of Technology
{mattmann, kelly, crichton, jshughes, shardman, pramirez, rjoyner}@jpl.nasa.gov

Abstract

In this paper, we present a preliminary study of several different electronic data movement technologies. We detail our approach to classifying the technologies included in our study and present the preliminary results of some initial performance benchmarking. Our studies suggest that highly parallel TCP/IP streaming technologies, such as GridFTP and bbFTP, outperform commercial and open-source UDP-bursting technologies in several of the key data movement dimensions that we studied.

1. Introduction

Scientific data systems within the National Aeronautics and Space Administration (NASA) are collecting, producing, sharing, and disseminating large amounts of data, which are growing by orders of magnitude in volume in remarkably short time periods. The issue of *data delivery*[1] is becoming of prime importance as the increasing volumes of data are outgrowing the existing data distribution and delivery mechanisms in place in many large scale data systems and archives. For instance, a motivating example occurs within the context of our recent work in NASA's Planetary Data System [1] (PDS). The PDS is NASA's archive for planetary science data. All NASA planetary science missions are required to ensure that the data generated by their scientific instruments is formatted in such a way that the PDS can receive it, catalog it using standard metadata, and make it available for distribution to the larger planetary science and educational community. A recently launched NASA planetary science mission, the Mars Reconnaissance Orbiter (MRO), however, will increase the size of the current PDS archive from 10 terabytes (TB), to over 100 terabytes. This is an almost *ten-fold* increase in the total data volume of the planetary archive to date (nearly 30 years), just from a single mission. Nonetheless, the requirements of PDS mandate that the large volumes of data generated by MRO must still be distributed from the PDS to the planetary science community in the same fashion as is done today.

To address the enormous projected volume increase from MRO, we have begun an exploratory study at the Jet Propulsion Laboratory (JPL). We are evaluating a set of data movement technologies for their key properties in enabling large volume data distribution. These include the basic file transfer protocol (FTP) [2], GridFTP [3], a commercial [4] and open source [5] UDP data transfer technology, and several "hard media" (e.g., storage brick [6], DVD) technologies. The major problem that we have identified during this task is the lack of knowledge with respect to classifying and comparing how each of the available off-the-shelf data movement technologies are able to deal with different use-case scenarios for data distribution.

In this paper, we present a preliminary approach to classifying data movement technologies for large scale data delivery. Our approch involves the classification of data movement technologies along seven key dimensions of data delivery which were selected via a literature study [7-9], and from our experience working on the PDS project. The rest of this paper is organized as follows. Section 2 presents the data movement technologies and the dimensions used to classify them. Experimental results from benchmarking these technologies are presented in Section 3. Section 4 follows with a discussion of the experimental results and Section 5 rounds out the paper.

2. Approach and Technologies Studied

We identified seven key dimensions of data movement that allowed us to compare data movement

[1] Also referred to as "data movement" and "data distribution" throughout the paper

technologies. Although our study also considered so called "hard media" technologies, such as Storage Bricks, and HD-DVD media, in this paper we will focus on the electronic data movement mechanisms. Due to page limitations, we cannot include the full classification matrix that we generated; however, its full treatment is provided in [10].

2.1 Dimensions

Scalability – The amount of data (the *volume*) that must be delivered to a particular *amount of nodes* (such as from point-to-point).

Reliability – The measure of the amount of faults per data transfer.

Ease of Use – The ease with which to *install* and *configure,* and *use* the data movement technology.

Transfer Rate – The amount of data (bits/second) that can be transferred point-to-point over a fixed period of time.

Cost to Operate – The estimated cost to operate a data movement technology once it has been deployed (estimated using a numerical scale, with 0 indicating negligible costs, and 5 indicating extremely high operational costs)

Cost to Implement – The cost of procuring the data movement technology, along with its deployment cost (estimated using the same 1-5 scale above).

Industry Adoption – The pervasiveness of the data movement technology (e.g., pervasive, bleeding edge).

2.2 Data Movement Technologies and their Classification

FTP. The classic File Transfer Protocol (FTP) defines a capability for clients to send data to servers using the underlying TCP/IP protocol of the public Internet. FTP is an open standard and widely deployed in almost every operating system, is well documented and is broadly accepted as the de facto data movement mechanism available in modern software.

Our studies showed that FTP exhibited low scalability in large volume transfers performed over a Wide Area Network (WAN). It was highly reliable; we did not record a single fault or operational failure in 20 hours of testing, and over 100 gigabytes (GBs) of data movement. FTP's operational and implementation costs are both negligible (e.g., both 0 on our scale).

SCP. The Secure Copy Protocol (SCP) [11] allows for the secure transfer of files between two machines using the SSH protocol for authentication and encryption over a network. Akin to FTP, SCP is built on top of the underlying TCP/IP protocol and is a public, open standard.

SCP, like FTP, exhibited poor scalability over the WAN in our testing results. SCP was highly reliable, with no faults incurred in over 20 hours of testing, using the same FTP datasets. The operational and implementation costs of SCP are both negligible, comparable to that of FTP.

GridFTP. GridFTP [12] extends the FTP protocol with new features required for large volume, fast data transfer, such as striping, partial file access and highly parallel stream-based usage of TCP/IP. GridFTP is the basic data movement mechanism provided by the Globus Toolkit [12], the widely adopted software packages for implementing grid-based software applications.

GridFTP's scalability was extremely high, it was able to transfer at a rate that was directly proportional to that of the dataset volume. GridFTP also exhibited high reliability, with no faults experienced. We found GridFTP to be difficult to deploy, requiring the setup of certificate management for hosts, users, and services. Because of this, we estimate that the implementation costs to be medium (around 3). On the other hand, the operational costs would be negligible (0), as the system is highly reliable and configurable as soon as it is deployed.

bbFTP. bbFTP [13] is an open-source parallel TCP/IP data movement technology. bbFTP's main capabilities are the ability to use SSH and certificate based authentication, on-the-fly data compression and customizable time-outs.

bbFTP is bleeding edge with little industry adoption, and documentation. We found bbFTP to be highly scalable, and also highly reliable (no faults during testing) and configurable. Cost to implement is negligible (0), as bbFTP was easy to find, download, configure and install. Its operational costs are somewhat higher (around 3), due to the small amount of user documentation provided.

UFTP. UFTP [5] or *U*DP-based *f*ile *t*ransfer *p*rotocol with multicast is an open-source data movement mechanism designed for efficient and reliable transfer of large amounts of data to multiple receivers simultaneously. UFTP is particular effective over a satellite link (with two way communication), or over high-delay Wide Area Networks (WANs) where the reliability mechanisms in TCP/IP severely under-utilize the available network throughput capabilities.

We found UFTP to be highly scalable in the only environment that we were able to test it in (the LAN), outperforming both Aspera and FTP and SCP on similar datasets and volumes. UFTP exhibited extremely poor reliability with the fault rate a function of the total dataset volume. UFTP is a bleeding edge technology, with a small customer base and little

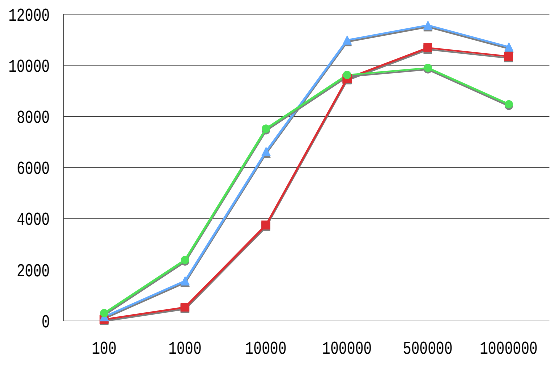

Figure 1. Transfer rate versus file size parallel TCP/IP LAN

documentation. Though it was not difficult to install, it was extremely difficult to configure its firewall rules. Because of this, we estimate high (4) implementation costs, and higher (5) costs to operate it.

Aspera. Aspera is a commercially available product built on top of a proprietary protocol which fully utilizes the capabilities of UDP to send bursts of data from one place to another. Aspera builds on top of the SCP protocol to provide secure, reliable, and most importantly fast transfer of voluminous data sets independent of network latency and packet loss.

We found that Aspera was not scalable, experiencing low transfer rates (nearly linear) based on dataset volume size. The fault rate in Aspera was negligible. Aspera was easy to deploy, and comes as an installer package for most operating systems. Aspera's documentation was fairly poor. Like UFTP, we could not discern the appropriate firewall rules to transfer data using Aspera in the WAN environment. We estimate the cost to implement Aspera to be low (2), however, the cost to operate it very high (4).

3. Experiments and Results

This section describes the results of the transfer speed experiments that we performed using the technologies studied. The experiments are broken down into two categories: LAN-based and WAN-based experiments. LAN experiments were conducted by transferring data between hosts on the local JPL LAN, an Ethernet (10/100 Mbps) LAN with stable connectivity and low latency. WAN-based experiments involved transfers between a host at JPL and a host at the USGS Imaging Node of the PDS. The experiments are further divided up into two categories based on the technologies studied (UDP and parallel TCP/IP).

3.1 TCP/IP parallel streaming JPL LAN

Transfer rate for varying file sizes (from 1B to 1GB) was measured across the JPL LAN. The tests were performed between two machines at JPL: the sender ran Fedora Core 1 on a dual 733MHz Pentium III processor machine with a 100Mbps Ethernet connection; the receiver ran Fedora Core 3 on a quad Intel 2.8GHz Xeon machine, also on a 100Mbps Ethernet connection. We fixed the amount of parallel TCP/IP streams to 8 on both GridFTP and bbFTP.

Figure 1 demonstrates the transfer rate (Y-axis, KB per second) versus file size (X-axis, KB), for three TCP/IP protocols on the LAN.

GridFTP (blue triangles) achieved the highest speed of 12MBps, followed closely by bbFTP (red squares) at 11MBps. Conventional FTP (green circles) could only achieve 10MBps, although for smaller file sizes (around 10,000KB) its single stream performs better than the parallel streamers.

3.2 UDP bursting JPL LAN

As with the TCP tests, we used a similar experimental setup for UDP bursting technologies. The sender for UFTP was an Apple running Mac OS X 10.3 on a dual PowerPC 2.0GHz machine on a 100Mbps Ethernet connection; the receiver was the sender from the TCP tests above. For Aspera, the roles were swapped. We tested file sizes between 47MB and 1GB.

Figure 2 depicts the transfer rate (Y-axis, KB per second) versus the file size (X-axis, KB).

The venerable FTP technology (green circles) based on TCP outperforms both of the UDP counterparts presented to it. Aspera (red squares) showed virtually no improvement regardless of file size, lethargically transferring data at a frustrating pace. UFTP (blue triangles) at first showed promise with smaller file sizes (under 78000KB) but quickly paled compared to FTP.

3.3 TCP/IP parallel streaming WAN

Latency is far greater on a WAN than on a LAN, and would also be the far more common use case for distribution of PDS data. Therefore, we measured transfer rates between two distant nodes separated on a WAN, in this case the public Internet. The sender ran Fedora Core 3 on a quad Intel 2.8GHz Xeon machine at JPL (Pasadena, CA). The receiver ran RedHat Linux 9 on an 800MHz Pentium III machine at the USGS (Flagstaff, AZ). While both machines had 100Mbps Ethernet connections, the WAN traffic in between them traversed over various media. Both GridFTP and bbFTP ran with 8 parallel streams.

Figure 3 shows the transfer rate (Y-axis, KB per second) versus file size (X-axis, KB) for three TCP/IP protocols on the WAN:

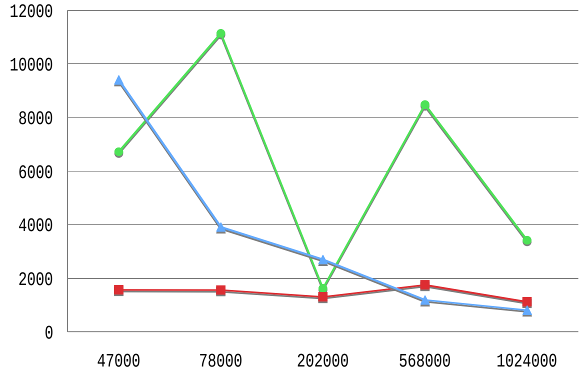

Figure 2. Transfer rate versus file size UDP LAN

The WAN shows an order of magnitude decrease in speed compared to the LAN. But it also clearly demonstrates that multiple streams enormously outperform a single stream. This time, bbFTP (red squares) won the contest with 1.2MBps transfer rate. GridFTP (blue triangles) came in a close second at 1.1MBps. Conventional FTP (green circles) made a dismal showing of only 0.19MBps.

3.4 UDP bursting WAN

While we enjoyed a successful series of tests with TCP and UDP technologies on the LAN, and with TCP technologies on the WAN, we were unfortunately unable to complete UDP tests on the WAN.

Both Aspera and UFTP make use of both TCP communication (for authentication) and UDP communication (for data movement). This requires that multiple firewall ports of multiple types be opened on both sides of the data transfer. While most network administrators are familiar with opening TCP ports, opening UDP ports seems to be more of a challenge. Worse, UFTP uses multicast UDP ports, which requires participants to be connected to the multicast backbone (MBONE) portions of the Internet. Neither JPL nor USGS were connected to the MBONE.

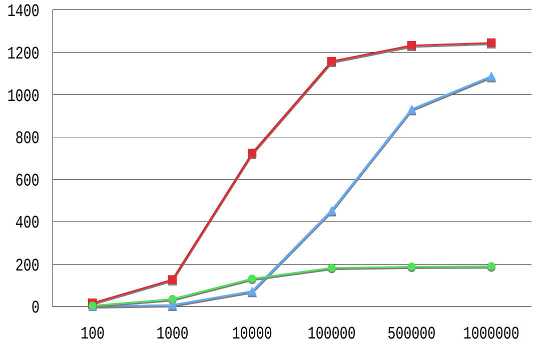

Figure 3. Transfer rate versus file size parallel TCP/IP WAN

As our requests for firewall exceptions escalated through multiple administrative strata, and our network analysis tools showed openings that should “just work,” we resigned to give up our efforts after several months. This was, however, valuable data: deploying UDP-based technologies throughout all users of the PDS would involve similar headaches at multiple nodes, and therefore we conclude, would be wholly inappropriate.

4. Discussion

The technologies we studied essentially rely on two basic strategies to improve data movement: parallel TCP streams and UDP bursting. These technologies provide only small improvements (or no improvement at all) in a LAN environment where the number of hops that a packet must traverse is low and data transfer rates are high. However, on a WAN, we see much more efficient usage of the network and higher throughput.

Latency on a WAN is much higher than on a LAN, and indeed high latency is the one characteristic that tends to cripple data transfer. Because TCP is a reliable data protocol, every packet must be acknowledged by the receiver. However, TCP does not wait for an acknowledgement before sending a single packet. Instead, both the TCP sender and receiver negotiate how many packets to try to keep “flying” during a transfer. This is called the “window size,” and the term “sliding window” refers to the window of packets that are currently pending acknowledgement. Larger window sizes generally mean better transfer rates, but require more resources in both sender and receiver. Further, because packets may arrive out of order, TCP must buffer additional packets for reassembly in the correct order before the receiving process can accept the data.

UDP, on the other hand, is not a reliable protocol. A sender that transmits a UDP packet has no guarantee that the packet is received, or that a series of packets are received in any particular order. Issues such as retransmitting packets and reassembly of data must be implemented manually within the application and not automatically by the operating system. This does afford applications the opportunity to make intelligent decisions about such issues that can be optimized for particular needs, such as bulk data movement. However, solutions to such optimization problems usually end up involving some combination of sliding windows and reassembly buffers, in effect re-implementing a TCP-like protocol over UDP. Moreover, TCP itself has had decades of testing and improvements, making it a well-understood and highly reliable technology.

Furthermore, the UDP technologies that we evaluated required such difficult changes to a site's firewall rules and network policies, that we were unable to successfully test their capabilities across the WAN. In the end, we wasted nearly *three* man months between personnel distributed across three institutions, ranging from computer scientists, to system administrators to network firewall experts, and we *still* could not get the UDP technologies to work across the WAN. In this light, we can see why the parallel TCP stream approach works better. Both bbFTP and GridFTP were testable across the WAN, while Aspera and UFTP were not.

Furthermore, channel utilization of bbFTP and GridFTP approached their theoretical limits. During our testing, we experienced such high levels of saturation with bbFTP and GridFTP that interactive terminal sessions where noticeably affected.

The majority of sites that connect to the public Internet institute as a matter of standard policy strict firewalls to protect internal machines from the ravages of worms, viruses, hackers, script kiddies, and so forth. Many sites deny UDP traffic outright (save domain name lookups) since there is often no need to pass such traffic outside the institution. This means that adoption of UDP flooding technology would require reconfiguration of firewalls across the PDS.

Adoption of parallel TCP stream technology would also require reconfiguration of firewalls. However, reconfiguration of TCP ports is an activity that most security administrators are intimately familiar with since ports must be frequently changed to support new applications, almost all of which use TCP. Opening additional ports for either bbFTP or GridFTP is simple. On the other hand, as evidenced by our failure to get Aspera and UFTP working on the WAN, opening UDP ports is not.

GridFTP, as part of the Globus Toolkit, is built on top of the Grid Security Infrastructure, GSI [14]. GSI is a highly secure system based on keypairs and certificate authorities. GSI assigns certificates to hosts, to services on those hosts, and to users. A data system could take advantage of this security infrastructure to control access to pre-calibration data, for example, and later to enable open access once datasets are published. Although this does make it harder to deploy GridFTP, the benefits may well be worth it. bbFTP can optionally be configured to use GSI but by default uses the more limited Unix-style /etc/passwd security. Either can be configured for anonymous access.

5. Conclusion

We presented a preliminary study and set of benchmarking experiments against two classes of data movement technologies: parallel TCP/IP technologies such as GridFTP and bbFTP, and UDP bursting technologies such as Aspera, and UFTP.

The results of our study indicate that parallel TCP/IP data movement mechanisms have a clear advantage over the UDP technologies in several comparison dimensions, including security support, transfer rate performance, and access policy and firewall configuration.

6. Acknowledgements

This effort was supported by the Jet Propulsion Laboratory, managed by the California Institute of Technology.

7. References

[1] J. S. Hughes and S. K. McMahon, "The Planetary Data System. A Case Study in the Development and Management of Meta-Data for a Scientific Digital Library.," in *Proc. of ECDL*, pp. 1998.

[2] J. Postel and J. Reynolds, "File Transfer Protocol (FTP) RFC Document, http://www.ietf.org/rfc/rfc959.txt."

[3] W. Allcock, J. Bester, et al., "GridFTP: Protocol Extensions to FTP for the Grid, http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf," RFC Draft Document 2001.

[4] "Aspera Software http://www.asperasoft.com," 2005.

[5] D. Bush, "UFTP - UDP based FTP with multicast. http://www.tcnj.edu/~bush/uftp.html," 2005.

[6] "Designing TeraByte Storage Bricks, http://elib.cs.berkeley.edu/storage/brick/system.html," 2005.

[7] A. Nayate, M. Dahlin, et al., "Transparent Information Dissemination," in *Proc. of Middleware*, pp. 2004.

[8] M. Franklin and S. Zdonik, "A Framework for Scalable Dissemination-based systems," in *Proc. of OOPSLA*, Atlanta, Georgia, pp. 1997.

[9] U. Centintemel and M. Franklin, "Self-adaptive user profiles for large-scale data delivery," in *Proc. of ICDE*, pp. 622-633, 2000.

[10] "http://www-scf.usc.edu/~mattmann/DM-Matrix-090105.doc," 2005.

[11] "Secure copy - http://en.wikipedia.org/wiki/Secure_copy," 2005.

[12] C. Kesselman, I. Foster, et al., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Intl' Journal of Supercomputing Applications*, pp. 1-25, 2001.

[13] "bbFTP - Large files transfer protocol, http://doc.in2p3.fr/bbftp/," 2005.

[14] V. Welch, F. Siebenlist, et al., "Security for Grid Services," in *Proc. of Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*, pp. 2003.