Experimental Studies of Data Transport and Data Access of Earth Science Data over Networks with High Bandwidth Delay Products

Robert L. Grossman, Yunhong Gu
David Hanley, Xinwei Hong and Parthasarathy Krishnaswamy
Laboratory for Advanced Computing
University of Illinois at Chicago

October 10, 2003

This is a draft of the paper: Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong and Parthasarathy Krishnaswamy, Experimental Studies of Data Transport and Data Access of Earth Science Data over Networks with High Bandwidth Delay Products, Computer Networks, 2004.

Abstract

Although the amount of earth science data is growing rapidly, as is the availability of high performance networks, our ability to access large remote earth science data sets is still very limited. This is particularly true of networks with high bandwidth delay products (BDP), such as those between the US and Europe. Recently, several network protocols have emerged that improve the situation and hold the promise of being much more effective than striped TCP. (In striped TCP, data is striped across multiple TCP streams). In this paper, we report on experimental studies using one of these new protocols called UDT, and compare UDT to other approaches. In addition, we consider the effectiveness of these new protocols when reading and writing data from disk over high BDP networks. We also consider the problem of accessing remote data by attribute over these same networks. We show that with the appropriate protocol, accessing data across the Atlantic can be improved significantly. We note that the UDT protocol used here can be deployed as an application library for earth science applications and neither requires upgrades to existing network infrastructure, such as routers, nor to the Linux kernels on the servers involved.

Keywords– TCP/IP, SABUL, UDT, high performance data transport, high-performance networking, high-performance computing.

1 Introduction

In this paper, we report on experimental studies using new network protocols for transporting and accessing earth science data over networks with high bandwidth delay products.

The bandwidth delay product (BDP) is defined to be the product of the network bandwidth and the round trip time (RTT), which is the time required for a TCP packet to travel from the source to the target and for the acknowledgement to return. For example, Table 1 shows the maximum effective bandwidth using various protocols for a high BDP link. The specific scenario represented in the table is the transfer of data between Chicago and Amsterdam over a 1 Gb/s link, where the round trip time (RTT) is about 110 ms, yielding a BDP of 12.5 MB.

As is well known and as can be seen from the table, the de facto transport protocol, TCP, is unsuitable for the high performance transport of data over routes with high BDPs. For example, when moving data between Chicago and Amsterdam, where the round trip time (RTT) is about 110ms, with a normal configuration (64KB buffer size), TCP achieves only 5Mb/s. Carefully tuning the window size can improve performance by 3-6 times. However, the performance is still far from satisfying. It will still take more than 4 days to move 1 TB data, which is unacceptable for most data intensive applications, such as scientific computing and distributed data mining.

The relation between RTT and maximal bandwidth for single TCP flows is, in practice, well estimated by the Mathis Equation [20]:

$$BW < \frac{C * MSS}{RTT * \sqrt{p}}.$$

Here BW is the number of bytes transmitted per second, MSS denotes the maximum segment size, p is the packet loss ratio, which is the number of packets that are retransmissions divided by the total number of packets transmitted, and C is a constant.

This paper addresses three important aspects of the problem of accessing remote earth science data over routes with high BDPs.

1. Designing alternative network protocols to move bits over high BDP links more efficiently than possible with currently deployed TCP implementations.

2. Improving the integration of network protocols with local disk input/output so that bits can be moved from local disk to remote disk over high BDP links more efficiently.

3. Integrating network protocols with local data management systems so that attribute-based queries of data records can be moved from local disk to remote disk over high BDP links more efficiently.

To say it another way, the first problem is concerned with blasting bits over networks, the second with reading and writing bits from disks, and the third with reading and writing data attributes from disks.

Most prior research has focused on the first problem. In this paper, we provide some preliminary experimental studies concerning the second and third problems.

In particular, earth science data is commonly viewed as consisting of one more attributes attached to each point in a two (latitude-longitude), three (latitude-longitude-height or latitude-longitude-time) or four (latitude-longitude-height-time) dimensional grid. For example, one or more hydrology measurements may be attached to each point in the grid. Traditionally, the gridded data is packaged into a file and the file as a whole is moved over a network. In this paper, we are also interested in the case in which direct support is provided to support remote and distributed access to attribute-based data in contrast to file-based data. By this we mean the case in which access to the gridded data is through specifying ranges for grid values and attributes of interest, or, more generally, by specifying an attribute-based query which retrieves some subset of the data.

The paper is organized as follows. Section 2 contains background and related work. Section 3 describes the various protocols used in the tests. Section 4 describes the testbed. Section 5 describes our experimental results. Section 6 is the summary and conclusion.

2 Background and Related Work

Broadly speaking, efforts to improve data transport over high BDP routes can be divided into three approaches:

The first approach is to improve TCP. Sally Floyd has proposed a variant of TCP called HighSpeed TCP (HSTCP) [6]. HSTCP modifies TCP's slow start algorithm to avoid sudden increases in the congestion window, which is usually an important source of packet loss in TCP. HSTCP also modifies TCP's additive increase multiplicative decrease (AIMD) algorithm. Unlike

Protocol	**Bandwidth**	**Comment**
TCP	5 Mb/s	standard TCP using 64 KB window
TCP	30 Mb/s	using tuned 12 MB = 1 Gb/s * 110 ms window
SABUL	950 Mb/s	does not require tuning or striping

Table 1: The table contains the bandwidth obtained using different protocols over a 1 Gb/s between Chicago and Amsterdam where the RTT is 110 ms. Here the BDP is 12.5 MB.

standard TCP, which uses constant increase and decrease rates, HSTCP varies the increase and decrease rates based on the current congestion windows. HSTCP has been implemented and incorporated into the Linux kernel by several parties. Experiments and simulations have shown promising results.

FAST TCP [18] is another project that modifies TCP. FAST is built on the idea of TCP Vegas. Using both queuing delay and packet loss as signals of congestion, it can reach the equilibrium state for the connection quickly and easily. Experiments show that FAST can sustain 1.3Gbps on a transatlantic OC-48 link. This project is still in its early stages. No API is available, and the detailed mechanism is still unpublished at this time.

Although the TCP-related solutions maintain compatibility with TCP, they are still far from being available to the ordinary user since they require new Linux system kernels with specialized TCP drivers. For this reason, many users prefer UDP-based methods which are described next and can be deployed without any changes to the generally available infrastructure.

The second approach is build reliable data transport algorithms on top of UDP. The basic idea is to use a UDP-based data channel and a TCP-based control channel to provide reliability, congestion control, flow control, etc. Since these types of protocols are built on top of TCP and UDP, no changes are required on the kernel level. There are several competing implementations of this approach include [10], TSUNAMI[21], and RBUDP[16]. Although the basic mechanisms behind these solutions are very similar, they have different rate control and congestion control algorithms, resulting in different properties and performances. Both Tsunami and RBUDP use block based data transfer without congestion control. Instead, an initial sending rate is negotiated and used throughout the session. At the end of each block, the receiver feeds back a list of lost packets to be retransmitted.

The third approach is to develop new network infrastructures. XCP [19], which is an open loop protocol operating at the network level, is an example of this approach. As another example, some researchers are advocating replacing the ISO Level 3 routed infrastructure with ISO Level 2 switches that directly switch application specific light waves or *lambdas*. Sometimes the resulting infrastructure is called a lambda grid. The idea is that an application on a lambda grid could set up, monitor, and tear down their lambdas, assuring them of the bandwidth they require [4].

For any approach it is possible to improve performance through the use of network striping [11]. For example, using multiple TCP connections improves application bandwidth as mentioned in the first section. Perhaps the most widely deployed striped TCP implementation is GridFTP [2]. It is relatively easy to develop an application that uses multiple TCP streams. No kernel level work is required. But as the number of the streams used grows, this method becomes complicated and the system overhead begins to impact the overall performance. Parallel TCP streams are also not fair in regard to network bandwidth consumption. GridFTP is the most widely used tool in this category. It will be described in more detail in the next section. It is also possible to stripe multiple UDP connections, as is discussed in [13].

A survey for high-speed data transfer protocols and tools can be found at [15].

Less work has been done on data protocols, that is internet protocols directly designed to access remote and distributed data. Data protocols include efficient and fair transport protocols and data access control protocols. There are two main deficiencies in current data protocols: 1) the data access protocols rely on TCP or higher level protocols like HTTP, which are not efficient for data transport and 2) there are few truly efficient and fair transport protocols available. A good example of this approach is the Distributed Oceanographic Data System or DODS [5]. DODS relies on the http protocol to access remote earth science data and has been integrated with a wide variety of different applications. However, DODS is not designed to address the problem of BDP networks. Another example is the DataSpace Transfer Protocol or DSTP [12], which was originally designed as a stand alone protocol to access and query remote and distributed data, but more recently has been redesigned as a web service.

3 Network and Data Protocols Studied

In this section, we will describe the network and data protocols used in our experimental studies.

3.1 GridFTP

GridFTP [2] is a high-performance data transfer protocol that is optimized for high-bandwidth, wide-area networks. It provides a superset of the features of the standard FTP protocol. It has become a standard tool in grid environments.

GridFTP provides support for striped data transfer by using multiple TCP streams between the source and its destination. It also provides capabilities for the portioning of data across multiple servers in order to improve the aggregate bandwidth. GridFTP allows an authenticated third-party to initiate, monitor and control the transfer of data between storage servers. It supports Grid Security Infrastructure (GSI) and Kerberos authentication. GridFTP supports the transfer of subsets or of the regions of a file. Check-Pointing provides fault tolerance. Failed transfers are restarted from the previous checkpoint. GridFTP also allows either manual or automatic control of the TCP buffer size. In our experiments, we controlled the TCP buffer size manually to obtain predictable performances.

Generally, GridFTP is an application layer tool which provides the secure, robust, reliable and flexible transfer of data in a high performance computing environment. A detailed description of GridFTP can be found in [1]. Background on the Grid Environment can be found in [8].

3.2 SABUL

SABUL stands for Simple Available Bandwidth Utilization Library. It is designed for applications that require very high bandwidths over networks with high bandwidth delay products.

SABUL can merge features of UDP and TCP to produce a light-weight protocol with flow control, rate control and reliable transmission mechanisms [10]. It uses a UDP-based data channel and a TCP-based control channel. On the UDP channel, separate from the regular UDP header, packets consist of a 32 bit SEQ field (packet sequence number) and a data field. By using UDP, SABUL can take advantage of UDP's fast transmission and its checksum mechanisms. Meanwhile, SABUL uses the TCP channel for rate control, congestion control and packet correction. Connection parameters are also exchanged through the TCP channel. By using both UDP and TCP, SABUL is able to achieve all of its functions on the application layer and thus provide simple and easy-to-use APIs [3]. It can be used by applications without making any changes to the operating system's kernel.

SABUL was developed on Linux and has been ported to other operating systems such as BSD, MacOS, etc. In our previous experiments, SABUL can easily get 900Mb/s throughput across the Atlantic from Chicago to Amsterdam over a 1 Gb/s link [13]. In contrast, TCP can usually get about 5Mb/s on the same link. During the IGrid 2002, we achieved 2.8Gb/s throughput across the Atlantic from Chicago to Amsterdam with 3 parallel SABUL connections. Detailed descriptions of SABUL, including experimental results can be found in [13].

3.3 UDT

UDT (UDP-based Data Transfer) extends SABUL by removing the TCP [14]. The design of SABUL does not require a reliable control channel, such as provided by TCP, per se. If a control packet, which contains packet loss information and/or acknowledgement, is dropped then the protocol ensures the resending of corresponding information in a later control packet.

UDT is built on top of UDP purely to utilize its checksum mechanism. Although only UDP is employed, UDT still maintains two channels: a control channel and a data channel, as in SABUL. There are various reasons not to use TCP. First, a protocol based on one underlying protocol is less complicated and more desirable than one based on two or more underlying protocols. UDT can provide failsafe transport even without the use of TCP.

Second, as already mentioned, UDT can independently guarantee the retransmission of the control information. No additional reliable channel is required. Third, TCP's retransmission mechanism actually defers the transfer of the control information in case of the loss of a control packet. The reason for this is that TCP has to depend on time-out to retransmit a packet, and this takes a long time in high BDP networks. Finally, UDT simplifies achieving TCP-friendliness. (We say a protocol is "TCP friendly" if TCP performance degradation in the presence of said protocol is minimal). In SABUL, achieving TCP-friendliness is more complicated because of the embedded TCP connection. To summarize, UDT avoids undesirable TCP retransmission and reordering, connection set-up time, and end-host burden, thus increasing performance and flexibility.

UDT improves the congestion control (CC) algorithm as well. It introduces dynamic window control and bandwidth estimation. Although these changes are not fundamental to the (static) window limit rate control of SABUL, they are critical to its performance. Our experiments have shown that UDT achieves better performance, especially in highly congested environments and on hosts with high CPU utilization.

A detailed description, as well as experimental results, can be found at [9]. UDT is open source and available via Source Forge or from the web site [3].

3.4 DataSpace Transfer Protocol (DSTP)

The DataSpace Transfer Protocol or DSTP is an internet protocol for accessing and querying remote and distributed data [12]. The current version of DSTP is Version 3.0 [3] and beginning with this version, DSTP has been redesigned as a web service. Open source DSTP servers and clients are available via Source Forge and from the Project DataSpace Home Page [3].

DSTP is designed to make accessing data via DSTP analogous to accessing documents via HTTP. Once data is put in a directory accessible to a DSTP server, it can be browsed, explored, analyzed, and mined by a DSTP client anywhere on the web. DSTP provides tools that allow end users to use their browsers to explore and query remote data. Publishing a dataset with DSTP can be as simple as copying the files into DSTP and creating a small XML file describing the dataset and its attributes. The data can be in a variety of formats, including text, SQL, netCDF, and HDF.

The DSTP protocol is designed for attribute-based data. By attribute-based data we mean data that is divided into records and in which each record shares a common set of attributes. A client using the DSTP protocol can request metadata from a DSTP server. Based upon the metadata, it can then request specific data sets and subsets of data sets. It can also ask the DSTP server to perform SQL like queries, server side sampling, and attribute selection. In addition, the DSTP protocol supports what are called universal correlation keys, which are keys designed to correlate geographically distributed attributes. For more details, see [12].

Recently, support for the open source R data mining package [23] has been added to the DSTP server.

When results are being returned via pure web services, the serialization and de-serialization of data limits the speed of the data's return as well as limiting the maximum possible return set size. DSTP also supports specialized high performance web services, which allows data to be streamed back via a pure-socket interface or through high performance data transport protocols such as UDT. For short network transport distances, the speed of a simple-socket interface should prove adequate; for long-haul networks UDT should be used. To further enhance speed, data may also be striped across the network. DSTP has been tested to return over 600 megabits per second for long duration operations on commodity hardware.

4 Description of Testbed

In this section we describe the testbed environment, including the network architecture and system configuration.

We built a testbed called the Tera Wide Data Mining (TWDM) Grid Testbed on top of three high speed networks: SURFnet[25], OMNInet[17] and StarLight [24]. The testbed currently consists of clusters located in three locations: at our Laboratory at the University of Illinois at Chicago, at the StarLight facility in downtown Chicago, and at the SARA Computing and Network Services facility in Amsterdam. StarLight functioned as an optical exchange point, allowing all three TWDM Grid clusters to communicate. We used a Trans-Atlantic SURFnet route to connect our cluster in Amsterdam to StarLight. OMNInet is a Chicago metropolitan area optical network, which we used to connect our cluster at the University of Illinois at Chicago to StarLight.

In principle, all of the networks support a bandwidth of up to 10Gb/s. However, currently, the throughput between them is limited. During our experiments, throughput between OMNInet and StarLight was 4 Gb/s, and the throughput between StarLight and SURFnet was set 1Gb/s.

Although experiments were done using all three clusters, the reports on networks with high BDPs, which are described in the next section, used the Amsterdam and StarLight clusters. The RTT over SURFnet between these two clusters was about 110 ms and the maximum bandwidth available was 1 Gb/s.

The Amsterdam cluster consists of 10 Dell Poweredge 2650 with 350GB of disk each. The Starlight cluster consists of 7 servers: five SuperMicro SuperServer 7042M-6 servers and three SuperMicro P4DC6 servers. Each server had between 100-200 GB of disk. All of the servers used Linux Redhat 7.3 (kernel 2.4.18-4smp). Each machine used either copper or fiber GbE ports. In addition, each machine was configured to use either hardware RAID or software RAID. (RAID stands for Redundant Array of Independent/Inexpensive Disks).

5 Experimental Results

5.1 Tuning the Disk Input/Output System

SABUL and UDT can consistently achieve more than 900Mb/s over a high BDP 1 Gb/s link, but the end-to-end disk-to-disk performance typically varied between 300–485 Mb/s for the testbed described above and was the bottleneck for many of the tests. In practice, end-to-end disk performance depends upon several factors, including:

- the speed of the disks;
- the disk drivers;
- whether RAID disk is used, and if so, what type;
- the type of buffering used;
- the block size of reads/writes

In tests with different machines, we were able to achieve up to 600 Mb/s when reading and writing to local disks, but unfortunately these machines were not available for use in the testbed.

In our testbed, each of the servers was attached to 5 disks, which were configured as single RAID disk. The disk data channel was Ultra 160 SCSI. This means that the throughput to and from the disks can theoretically be as high as 8*160Mb/s or 1.28Gb/s.

Our initial tests writing to disk achieved a throughput of about 300 Mb/s. By tuning the disks in various ways, this could be improved to approximately 600 Mb/s. First, changing from RAID 5 to RAID 0, the fault tolerance is sacrificed but write speed is improved significantly since there is no need to do the checksum computation. In this case, the speed was improved by approximately 100

Second, enabling disk write caches further improved the performance. Disk write caches have been considered an undesirable feature for streamed data because the caches quickly become filled. But in most cases for the RAID disks, the disk cache can improve performance since it is not possible for the streamed data (even on large files) to saturate all of the disk write operations. On our systems we can get 20 -30 percent performance improvement if we enable the disk write cache.

Third, setting the block size of the read/write system calls turned out to be an important component when tuning the disk i/o performance. The particular tuning depends upon the operating systems and drivers, especially

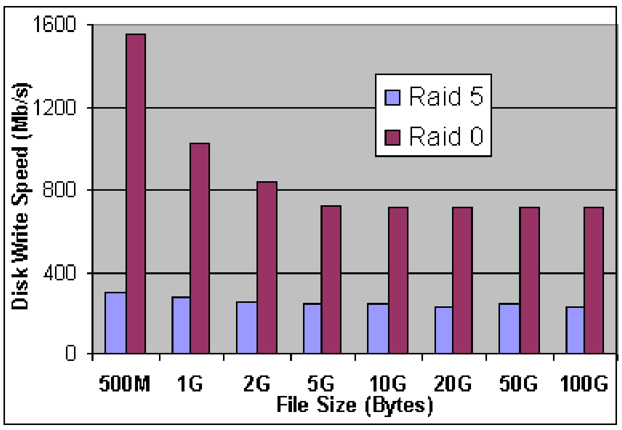

Figure 1: Disk write performance for different file sizes. Larger writes minimize the effect of a write cache.

on how the operating system schedules the lower level disk I/Os. For our disks, setting the block size for disk reads/writes to be the same as the RAID chunk size provided optimal performance.

Some results are presented in Figures 1 and 2. The results are obtained with different file sizes ranging from 500MB to 100GB. As the file size increases, the impact of disk cache decreases. In Figure 1, when the file size is greater than 5GB, the write speed is basically stable. The results for smaller files tended not to be as accurate and we have not included them. The figures also present performance differences between RAID 5 and RAID 0.

To summarize, in our testbed, optimal disk i/o was obtained by using RAID 0 with the write cache enabled and a data block size of 64KB. This was the configuration used in the tests reported in the next two sections.

5.2 End-to-End Disk to Disk Performance over Links with High BDPs

In this section, we describe the results of our experiments using the three different network protocols described in Section 3. The data for these tests was Community Climate Model (CCM-3) data in netCDF format obtained from NCAR [7]. The data was transported from StarLight in Chicago to SARA in Amsterdam over a 1 Gb/s link with 110 ms RTT, as described in Section 4.

The performance of GridFTP is shown in Table 2 and Table 3. Obtaining

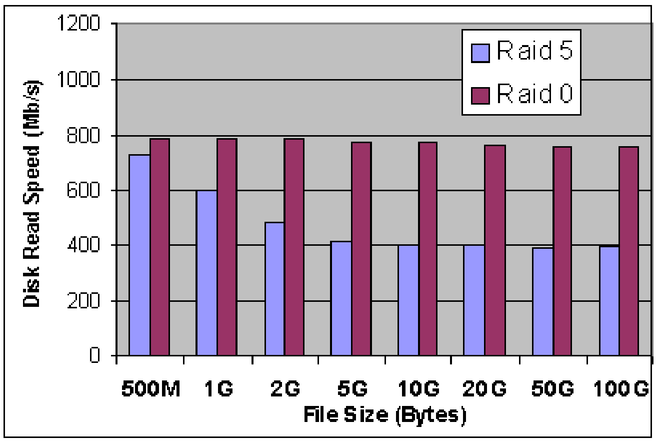

Figure 2: Disk read performance for different file sizes. Typical data mining files may range from kilobytes to petabytes in size.

Connection no.	1	8	16	32	64	128	256	512
Throughput (Mb/s)	49.4	88	103	124	176	240	280	248

Table 2: Performance for GridFTP with different connection number when file size is 2GB.

the performance GridFTP required careful and time consuming tuning in order to determine the correct number of striped connections and the proper window size. The window size and number of connections are decided by the link capacity. The end systems can also affect these values. Dataset size turns out to have a trivial effect on the optimum values.

We next repeated the same tests using SABUL and UDT. We were able to transfer 100 GB using SABUL at about 280 Mb/s, the same speed as for GridFTP, but without any tuning required, a significant advantage.

Finally, we tested UDT. We were able to transport 100 GB of data at about 400 Mb/s, a significant improvement over SABUL and GridFTP. Smaller data sets achieved a throughput as high as 485 Mb/s. This higher throughput for smaller datasets compared to larger datasets is due the cache impact on the disk performance). See Table 3 for details. The superiority of UDT compared to SABUL appears to be because of UDT's traffic control mechanism, which is described in [14].

File Size (bytes)	500M	1G	2G	5G	10G	20G	50G	100G
Time used by GridFTP (s)	22.7	41.8	57.1	141	260	547	1418	2857
Throughput (Mb/s)	176	191	280	283	308	292	282	280
Time used by SABUL (s)	12	24.3	53	138	275	555	1428	2857
Throughput (Mb/s)	331	329	298	290	290	288	280	280
Time used by UDT (s)	8.2	17.4	37	93	194	394	992	1995
Throughput (Mb/s)	485	460	431	430	411	406	403	401

Table 3: Disk-to-Disk Performance for GridFTP, SABUL and UDT where GridFTP uses 256 connections.

Record Count	**SOAP (sec)**	**DSTP (sec)**
10,000	.65	.21
50,000	2.57	.72
150,000	11.13	2.05
375,000	51.18	5.01
1,000,000	5:52.10	13.43

Table 4: Comparing SOAP and DSTP for attribute-based retrieval of data over routes with high BDPs. Approximate record size is 35 bytes.

5.3 Attribute-based Access to Data over Routes with High BDP

Finally, we tested two mechanisms for attribute based access to data from disk. For this test, we again used NCAR data in netCDF format, which consisted of several attributes including, latitude, longitude, temperature, time, and humidity. For this test, we put the data into a comma separated variable (CSV) format and retrieved all records satisfying the SQL-like query:

```
select temperature, time from test data where humidity > 50
```

First, we tested retrieving this data using Simple Object Access Protocol (SOAP) [22] and TCP. Second, we tested the query using the DataSpace Transfer Protocol (DSTP) and UDT. The results are reported Table 4.

As is apparent from the table, as the size of the data set grows SOAP's performance begins to suffer; in contrast DSTP/UDT scales approximately linearly with the number of records retrieved.

In this case, the records were approximately 35 bytes in size. This is a relevant measure for data mining, because in many data mining activities, such as determining medians, building trees, and learning processes, a vast amount of records must be retrieved and examined.

We believe that in many instances, flat files are the appropriate mechanism for storing records. Flat files are often the most straightforward and compact mechanism for representing data, may be most trivially transported, migrated, mirrored and stored, are portable, may be appended in constant time, and introduce a large degree of fault-tolerance. A database may be a superior storage mechanism when small ranges are frequently selected, there is adequate disk space for the required indices, and indexing overhead is not severe.

6 Summary and Conclusion

The amount of archived earth science data has been growing rapidly during the past several years, as has wide area high performance networks. For example, there are now 1 Gb/s research networks linking research sites in North America and Europe. On the other hand, the round trip time (RTT) between these sites can be 100 ms or more, which means the bandwidth delay product (BDP) is relatively high (about 12.5 MB in the testbed we used).

In practice, a scientist's ability to access data remotely over networks with high bandwidth delay products (BDPs) is in many cases limited. Parallel TCP (e.g., GridFTP) and TCP tuning (e.g., optimizing window size) are perhaps the most common methods used today to achieve high performance data access in high BDP networks. However, GridFTP requires that Globus be installed, which although widely available, cannot always be counted on to be installed on systems of interest. Without GridFTP or other specialized protocols installed, bandwidth across the Atlantic is usually about 5 Mb/s, even on a 1 Gb/s network.

Recently, several high performance network protocols have been developed. These new network protocols can effectively blast bits over a 1 Gb/s network with high BDP at a rate exceeding 950 Mb/s [13]. The challenge today is to ensure that accessing bit files from disk and accessing attribute based data from disk can enjoy similar performance. This paper describes preliminary experimental results in this direction.

Meanwhile, we notice that applications usually need to transfer data from disk to disk. A high speed data transfer protocol might be of no help if the disk is too slow. Actually, disk operation and data transfer protocols will affect each other since they will compete for the CPU resources. A slow disk will hurt the performance of data transfer protocols severely. In order to obtain good end-to-end performance, techniques to improve disk read/write speed are required. Given current advances in technology, the disk speed is usually the bottleneck in high speed data transfer. A good data transfer protocol should be able to adjust itself efficiently based on the speed of the disk, and reach the equilibrium state rapidly, so that the interaction between disk access and the protocol is minimized.

In this paper, we experimentally analyzed transporting large earth science data sets from a cluster in Amsterdam to a cluster in Chicago using several different network and data protocols. The data sets were derived from CCM3 data obtained from NCAR and ranged in size from 0.5 GBs to 100 GBs.

The good news is that current high performance data transport protocols such as SABUL/UDT, FAST, etc., can move data significantly faster than can be done with TCP or striped TCP. The bad news is that there is a significant performance drop when accessing data from disk. Such a gap will always be present, but further research can probably reduce the size of the gap. Using the techniques proposed in this paper, the disk speed can be improved by more than 100a standard TCP. In addition, the UDT network protocol that was used does not require any changes to the network infrastructure nor the installation of any specialized Linux kernels.

Work on high performance access to remote and distributed attribute based data is just beginning and much more work needs to be done. On the other hand, by layering attribute based access protocols over high performance data transport protocols significant gains can be achieved. Since the expectation of many is that future systems will be built on web services, we compared accessing remote data over 1 Gb/s network with a high bandwidth delay product using web services and specialized data protocols, such as DSTP layered over UDT [13]. In the studies reported here, even for small data sets containing just a million data records, speed ups of 25x or more could be achieved. Our conclusion is that much further work is required to develop data protocols for accessing remote data by attribute over high performance networks with high BDPs. On the other hand, even from the preliminary work reported here, it is clear that specialized web services for accessing remote data, and which do not rely on today's standard TCP protocol and XML formats, must be developed.

Finally, we mention that 10G networks are now emerging and additional work is required to understand how high performance data and network protocols scale to these networks.

References

[1] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke. Data management and transfer in high performance computational grid environments. *Parallel Computing Journal*, 28:749–771, 2002.

[2] A. Chervenak, I. Foster, C. Kesselman, and S. Tuecke. Protocols and services for distributed data-intensive science. In *ACAT2000 Proceedings*, pages 161–163, 2000.

[3] Project DataSpace. Dataspace home page. http://www.dataspaceweb.org, retrieved on September 4, 2003.

[4] Tom DeFanti, Cees de Laat, Joe Mambretti, Kees Neggers, and Bill St. Arnaud. Translight: A global-scale lambdagrid for e-science. *Communications of the ACM*, 2003.

[5] DODS. Distributed oceangraphic data system. http://www.unidata.ucar.edu/packages/dods/, retrieved on March 30, 2003.

[6] Sally Floyd. Highspeed tcp for large congestion windows. http://www.icir.org/floyd/hstcp.html, 2002.

[7] National Center for Atmospheric Research. Community climate model. http://www.cgd.ucar.edu/cms/ccm3/, April 10 2002.

[8] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. volume 15, 2001.

[9] R. Grossman, Y. Gu, and X. Y. Hong. Using udp for reliable data transfer over high bandwidth-delay product networks. Submitted for publication, 2003.

[10] R. L. Grossman, M. Mazzucco, H. Sivakumar, Y. Pan, and Q. Zhang. Simple available bandwidth utilization library for high-speed wide area networks. *Journal of Supercomputing*, to appear.

[11] R. L. Grossman, H. Sivakumar, and S. Bailey. Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Proceedings of Supercomputing 2000*, 2000.

[12] Robert Grossman and Marco Mazzucco. Dataspace - a web infrastructure for the exploratory analysis and mining of data. *IEEE Computing in Science and Engineering*, pages 44–51, July/August, 2002.

[13] Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong, Dave Lillethun, Jorge Levera, Joe Mambretti, Marco Mazzucco, and Jeremy Weinberger. Experimental studes using photonic data services at igrid 2002. *Future Generation Computer Systems*, 2003.

[14] Y. Gu and R. L. Grossman. Udt: A transport protocol for data intensive applications. http://www.ietf.org, 2003.

[15] E. He and J. Leigh. Survey of protocols and mechanisms for enhanced transport over long fat pipes. http://www.evl.uic.edu/eric/atp, 2003.

[16] E. He, J. Leigh, O. Yu, and T. DeFanti. Reliable blast udp: Predictable high performance bulk data transfer. In *IEEE Cluster Computing*, 2002.

[17] Optical Metro Network Initiative. http://www.icair.org/omninet/, retrieved on September 4, 2003.

[18] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Sing. Fast tcp: From theory to experiments. submitted for publication, 2003.

[19] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *Proceedings of the ACM Sigcomm*, 2002.

[20] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communication Review*, 27(3), 1997.

[21] University of Indiana Advanced Network Management Laboratory. Tsunami. http://www.anml.iu.edu/anmlresearch.html, retrieved on January 10, 2003.

[22] Simple Object Access Protocol. http://www.w3.org/TR/soap/, 2003.

[23] R project. Retrieved from http://www.r-project.org, January 10, 2003.

[24] About StarLight. http://www.startap.net/starlight/ABOUT/, 2003.

[25] Surfnet: High-quality internet for education and research. http://www.surfnet.nl/en/, retreived on September 4, 2003.