Data Mining Middleware for Wide Area High Performance Networks

Robert L. Grossman*, Yunhong Gu, David Hanley, and Michal Sabala
National Center for Data Mining, University of Illinois at Chicago, USA

Joe Mambretti
International Center for Advanced Internet Research, Northwestern University, USA

Alex Szalay and Ani Thakar
Johns Hopkins University, USA

Kazumi Kumazoe and Oie Yuji
Kitakyushu JGNII Research Center, Japan

Minsun Lee, Yoonjoo Kwon, and Woojin Seok
Korea Institute of Science and Technology Information, Korea

ABSTRACT
In this paper, we describe two distributed, data intensive applications that were demonstrated at iGrid 2005 (iGrid Demonstration US109 and iGrid Demonstration US121). One involves transporting astronomical data from the Sloan Digital Sky Survey (SDSS) and the other involves computing histograms from multiple high volume data streams. Both rely on newly developed data transport and data mining middleware. Specifically, we describe a new version of the UDT network protocol called Composible-UDT, a file transfer utility based upon UDT called UDT-Gateway, and an application for building histograms on high volume data flows called BESH for Best Effort Streaming Histogram. For both demonstrations, we include a summary of the experimental studies performed at iGrid 2005.

Keywords
High performance network protocols, high performance networks, high performance data mining, data mining middleware

1. INTRODUCTION
High-speed (1Gb/s and 10Gb/s) wide area networks provide us the opportunity to deploy data intensive applications over large geographic areas. Until recently, distributed data intensive applications were usually designed to minimize inter-process data communications; if large data transfers could not be avoided, large data sets were sometimes loaded onto disks or tapes and physically sent to remote sites. As a consequence, there were usually substantial delays when analyzing large distributed data sets, especially when two or more such data sets had to be integrated.

For example, telescopes in the Sloan Digital Sky Survey (SDSS) [19] collect gigabytes of data per day. This data is currently stored locally, and a data release is made periodically, e.g., every quarter of a year. The data is then sent to astronomers around the world via disks or tapes. Analysis results that produce large data sets are difficult to exchange among the astronomers. Also, overlaying a second data set on top of the SDSS data in order to discover astronomical objects that are too faint to be identified from one data set alone requires a substantial effort.

With high-speed wide area optical links connecting the observation stations, processing centers, and astronomers, these data sets and the analysis results can now be shared in near real time. Thus the processing delay can be significantly reduced and different data sets can be more easily combined.

However, existing applications cannot automatically make use of the emerging high-speed networks. First, the *de facto* Internet transport protocol, TCP as usually deployed, significantly

* Contact Author: grossman@uic.edu

Robert L. Grossman is also with Open Data Partners.

underutilizes the network bandwidth in high-speed long distance environments. Several alternatives and enhancements to TCP have been developed over the past several years [8], including UDT [12]. Second, the current generation of data mining software and middleware was not designed to process data at high speeds across distributed computing sites and data sources.

At iGrid 2005, we demonstrated three middleware applications designed to address these issues. One is a new version of the UDT protocol we have previously described [12] that is composible in the sense that it is designed to easily support different congestion control algorithms [10]. The application is called Composible-UDT. The remaining two middleware applications are built over Composible-UDT. The first of these is a file transfer utility called UDT-Gateway that provides access to UDT-based data services using TCP-based applications for the "last mile." This greatly expands the population of end users that can use UDT-based data services. The second of these is a best effort online histogram application called BESH for Best Effort Streaming Histogram.

At iGrid 2005, we demonstrated Composible-UDT, UDT-Gateway and BESH in two demonstrations. The first was iGrid Demonstration US121, which transported data from the SDSS. The second was iGrid Demonstration US109, which computed streaming histograms using data from web logs for web servers that provided results of the 1998 World Cup.

In Section 2 we describe the experimental set up. In Section 3, we describe the data transport middleware: UDT and UDT-Gateway. In Section 4, we describe the data mining middleware for computing histograms on high volume streaming data. The experimental results are described in Section 5. Section 6 briefly reviews related work. Section 7 contains a summary and conclusion.

2. EXPERIMENTAL SETUP

In this section we describe the hardware and network infrastructure used in our iGrid 2005 demonstrations. We also describe the data sets we used.

2.1 Hardware Infrastructure

We had 4 dual Opteron machines with 10GE NICs at the iGrid 2005 Conference, which were each connected to a 10GE port on one of the iGrid 2005 switches.

Located outside the conference, as part of a testbed we operate called the Teraflow Testbed [22], we had four dual Opteron machines with 10GE NICs. Two of the machines were in Chicago, one in Tokyo, Japan, and one in Daejeon, Korea. Each was connected to one of the four machines at the conference via routed 10 Gb/s link (Figure 1).

All the machines have dual AMD Opterons running at 2.4 Ghz, 4 GB of physical memory, and 1.5 TB RAID 5 or RAID 0 disk space, except for the Korean machine, which uses dual XEON processors. Debian Linux 2.6 SMP is installed on each of these systems.

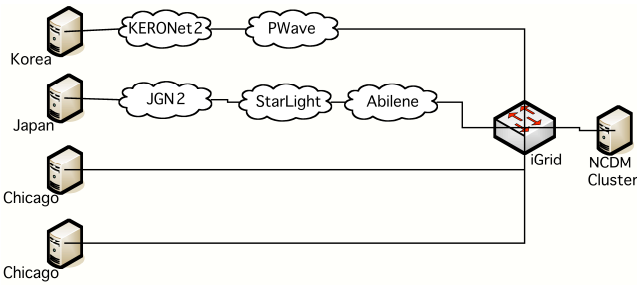

Figure 1. iGrid 2005 NCDM Booth Network Diagram. *This figure illustrates the 10Gb/s network infrastructure between several nodes on the Teraflow Testbed and the Linux cluster at iGrid 2005. All links have 10Gb/s capacity.*

2.2 Network Infrastructure

The two machines in Chicago (Figure 1) were located at StarLight and were each connected to their counterparts at iGrid via 10 Gb/s optical paths. The third machine at iGrid was connected to its counterpart in Daejeon, Korea via the PWave and KERONet2 (GLORIAD) networks. The fourth machine at iGrid was connected to its counterpart in one of the JGN II research centers located in Tokyo, Japan via the Abilene, StarLight, and JGN II networks. Each of the connections was 10 Gb/s. The RTT between the Chicago and iGrid was 66 ms; the RTT between Tokyo, Japan and iGrid was 180 ms; and the RTT between Daejeon, Korea and iGrid was 139 ms.

2.3 SDSS Data Set

In US121 demonstration, we preloaded 797 GB of compressed Sloan Digital Sky Survey (SDSS) [19] data to our four machines at iGrid. This was the BESTDR3 release of the data. The data was divided into 64 files, each of which was about 12.463 GB. During our demonstration, we moved this data to the machine in Korea. When uncompressed, the data was 1.5 TB.

After iGrid, we moved SDSS data from our machines at StarLight to our machines in Japan and applied the streaming histogram algorithm (used in US109) to the SDSS data to analyze the distribution of the brightness of the stars. The results are reported in Section 5.

2.4 World Cup 98 Data Set

The US109 demonstration used web log data from four web servers located in Paris, California, Texas and Virginia that provided information about the 1998 (soccer) World Cup. For the demonstration, we replicated this data onto four of our machines located in Chicago (2), Japan, and Korea.

The original data set contained 8 attributes, including the number of bytes of the data transfer. This attribute ranged in value from 0 to 2^{31}. Additional derived attributes were added for this demonstration bringing the number of attributes to 13. Using the iGrid machines, a separate histogram was built on each of the four streams, and then the four histograms were merged to produce a single summary histogram for all four streams.

3. DATA TRANSPORT MIDDLEWARE

It is well known that TCP substantially underutilizes the network bandwidth in high bandwidth-delay product environments. We have analyzed this problem and developed practical solutions since 2001 [9, 10, 11, 12, 13, 14].

As mentioned above, at iGrid 2005, we demonstrated the current version of UDT, (called Composible-UDT) and a file transfer utility based upon Composible-UDT called UDT-Gateway. In this section we will give a brief review of Composible-UDT and UDT-Gateway.

3.1 UDT

UDT is an application level data transport protocol designed for the emerging applications that will require transfer of large amounts of data distributed over high-speed wide area networks (e.g., 1 Gb/s or above). UDT uses UDP to transfer data but unlike simple UDP it has its own reliability control and congestion control mechanisms. UDT is not only for private or QoS-enabled links, but also for shared networks. Furthermore, the current version of UDT that was used at iGrid is designed using a Composible framework that supports multiple congestion control algorithms. For more information about UDT, see [12].

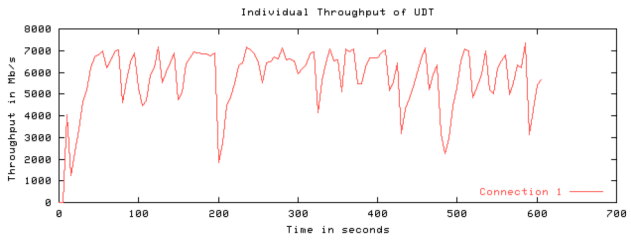

Figure 2. UDT Memory-Memory Data Transfer Performance. *This performance was obtained between two dual AMD Opteron machines between Chicago and Tokyo, via a 10Gb/s link with 180 ms RTT.*

Figure 2 shows the memory-memory data transfer performance using UDT between a pair of machines at Chicago and Tokyo, respectively. Because CPU usage limits the throughput of a single flow of UDT to about 5 Gb/s, we started two parallel UDT flows between the machines to take advantage of the dual processors on each machine. Using two UDT flows, we can reach a peak throughput of more than 7 Gb/s, which is near the hardware limit.

In the experiment shown in Figure 2, UDT reached an average throughput of 6.21 Gb/s, with the maximum throughput of 7.10 Gb/s. The standard deviation of the throughput per one-second unit time is 0.81 Gb/s.

3.2 Composible-UDT

Composible-UDT is not only a transport protocol library, but also a framework that supports many protocol configurations, in particular different congestion control algorithms.

Previously, we have provided an overview of Composible-UDT's congestion control framework, which is called CCC [10]. UDT/CCC allows user defined congestion control algorithms to be easily implemented. The UDT/CCC library enables easy implementations of a large variety of congestion control algorithms. The overhead of the framework, compared to a direct implementation, is minimal. When using the default algorithm, Composible-UDT has essentially the same performance as our direct UDT implementation.

Composible-UDT is still an ongoing project. Future releases will also include more configuration abilities, such as limited data reliability.

3.3 UDT Gateway

For many end users, it is easier to use a file transfer utility employing TCP, or a web application employing HTTP and TCP, rather than to use UDT directly. To support this requirement, we developed the UDT-Gateway utility. To the user, it appears they are accessing data using a TCP-based application on the gateway machine, but, in fact, the data resides on a data server that is connected to the gateway machine using a high performance network and UDT. The data server can serve multiple gateway machines.

Specifically, the UDT gateway behaves exactly as an HTTP file server, and serves clients files via the ordinary HTTP/TCP channels. However, the gateway server does not host the files it serves locally. When a request arrives for a file, the file is streamed from a central repository via UDT, then streamed to the end consumer via TCP. In other words, the gateway machine allows the user to access large data sets using UDT and high performance networks for all except the "last mile," which is handled using more standard networks and TCP.

HTTP/TCP access for the last mile solves many practical issues related to firewalls that are still a problem for many end users. The idea is that gateways can be placed on high-speed backbones, and end-users will simply use gateways close to their actual location. This system also provides the benefit of enabling what we call "lightweight routes." The UDT gateways sit on the high-speed backbone, as does the central data repository. The user fetches the data from the closest gateway, and indirectly retrieves the data from the central repository. This not only makes the best use of high-speed backbones, but also hides the procedure to locate the actual data repository, which increases the security and flexibility of the system.

4. Data Mining Middleware

In this section, we describe the data mining middleware we demonstrated at iGrid 2005.

Simply developing high-speed data transport middleware will not enable wide-area data intensive applications. We also need data mining middleware that scales to high volume data flows. At iGrid, we demonstrated data mining middleware supporting a streaming model for processing data. This model places two requirements on the algorithm: first, the data is examined only once; second, only a fixed amount of storage (independent of the size of the data stream) is available.

There is quite a bit of prior work on streaming algorithms. However, we are not aware of any prior work addressing how to scale streaming algorithms to 1 Gbps or 10 Gbps streams.

A key component of several data mining algorithms is computing histograms. We recently developed a binary partition dynamic histogram algorithm that is designed to scale to high volume data flows. As mentioned above the algorithm is called BESH for Best Effort Streaming Histogram.

At iGrid, we used BESH to solve the following problem. Suppose there is an infinite data stream consisting of integers between 0 and 2^{31}. We want to keep a histogram H to record the total number of each value that has appeared so far.

Because the value space of the attributes of the data stream is too large to maintain an accurate histogram on it (due to the limitation of memory space), we employ an approximate histogram algorithm in which each bucket covers a scope of multiple values, rather than a single value. For example, a bucket [*x*, *y*] tracks the number of records whose specific attribute has a value between *x* and *y*.

Because we must process the data stream in a single pass, we do not know in advance either the number of buckets or the size of each bucket. For this reason, we dynamically split and merge the buckets as we process the stream.

There are three major approximate histograms: equiv-width, equiv-depth, and V-Optimal [4, 5]. The histogram that results from our algorithm is similar to the equiv-depth histogram, but not the same.

The binary partitioning algorithm is illustrated in Figure 3. At the very beginning, there is only one node in the system: [0, 1024] (supposing the values are between 0 and 1024). Once a new value arrives, the root node is split evenly into two nodes. As values continues to arrive, if a node contains more values than a threshold, it will further be split. Similarly, if a node contains a very small number of values, all of its sub-nodes will be merged.

The leaf nodes consist of the histogram. For example, in Figure 3, the histogram on the stream with values varying between 0 and 1024 is: (0 – 256: 1000), (257 – 768: 2500), and (769 – 1024: 1500).

In order to describe the detailed algorithm, we define the following four parameters:

N: Total of values scanned so far.

$V[B_i]$: The number of values in bucket B_i.

Max-thresh: The upper percentage limit of the size of buckets. Buckets exceeding this limit must be split.

Min-thresh: The lower percentage limit of the size of buckets. Back-to-back buckets smaller than this size must be merged.

BESH Initialization: There is one bucket in the histogram, covering values from 0 to 2^{31}.

Computing a histogram using BESH. For each new record:

1. Locate the bucket B that the new value belongs to, update V[B], all its parent nodes, and N;
2. If (V[B] / N > max-thresh), equally split the node into two leaf nodes;
3. If new nodes are generated in Step 2, check every node in the tree. For any node B such that (V[B] / N < min-thresh), remove all its children nodes.

The leaf nodes are not the final result. Once a partial histogram of the stream is needed, a merge process on the leaf nodes is executed:

BESH Merge: For every leaf node B from the left to the right, if (V[B] / N > min-thresh), then it is one of the buckets in the final result. On the other hand, if the last bucket on the current final histogram B_p satisfies ($V[B_p]$ / N < min-thresh), then B will be merged with B_p, and no new buckets will be inserted into the final histogram; otherwise B will be inserted into the final histogram.

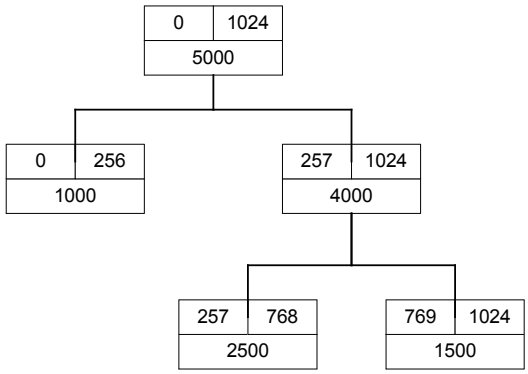

Figure 3. The BESH Dynamic Binary Partition Histogram. *Each node contains the lower and upper limit of values in one bucket, and the number of values in the bucket. The leaf nodes consist of the histogram.*

Merging histograms from different streams. Finally, because there is one histogram on each stream, the final histogram H is produced by merging all the sub-histograms H_i:

1. For every bucket in every sub-histogram H_i, put the boundaries of the bucket into a boundary list.
2. For every interval along the boundary list constructed in Step 1, a new bucket B is allocated. For each bucket B' in every histogram H_i, if B' overlaps with B, then update B with the portion of the contribution from B', assuming that the values in B' are evenly distributed.
3. Apply the BESH Merge described above to the histograms constructed in Step 2.

The dominant cost of computing a BESH histogram is the cost of the updates. Because there is an upper limit and a lower limit for the size of each bucket, the total number of leaf buckets is less than M, where M = 1/min-thresh. For each new record, in the worst case, all the existing buckets may have to be scanned to check if there are any buckets to be merged. For this reason the cost of computing a BESH histogram is O(MN).

5. EXPERIMENTAL RESULTS

For each of our demonstrations, we had two official time slots. We also tested our applications during the nighttime, especially the BESH algorithm.

US109 Experimental Results. The performance of US109 is recorded in Figures 4, 6 and 7, which are the figures taken from the real time display during one of our demos. Figure 4 shows the real time dynamic histogram on the four data streams. Figure 6 shows the aggregate throughput of the streaming data mining application. The average speed is around 8 Gb/s with a peak speed of 14 Gb/s. The per-flow throughput is shown in Figure 7. Each flow realized an average throughput of 2 Gb/s.

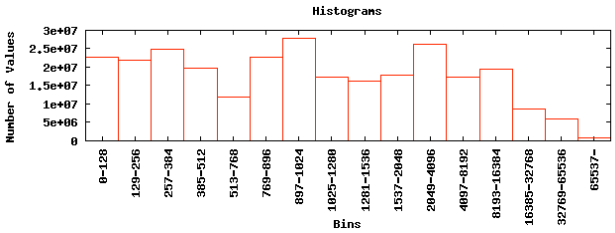

Figure 4. Real Time Histogram on High Speed Data Streams (US109). *This figure is a snapshot of the histogram on the four real time web traffic streams used in US109 demo.*

Figure 5 shows the counterpart histogram obtained by a standard histogram algorithm (i.e., without any of the limitations imposed by the streaming model) using the same buckets as those in Figure 4. The two figures are almost identical, and, at least for this data, there is very little loss in accuracy when using the best effort BESH algorithm.

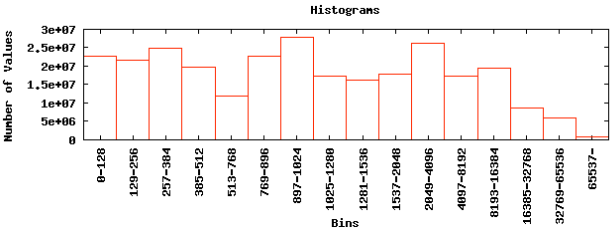

Figure 5. This figure contains a histogram computed from the same data sets that were used for US109 but computed as usual with a standard histogram algorithm. *The same buckets were used for the streaming BESH algorithm in US109.*

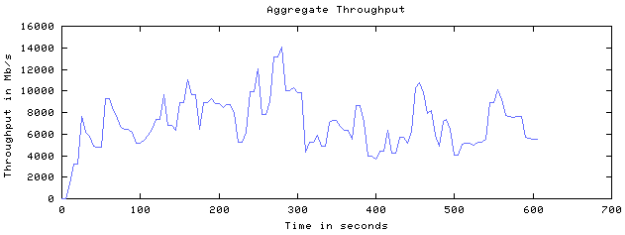

Figure 6. Aggregate Throughput of Streaming Data Mining (US109). *This figure shows the aggregate throughput of computing histograms over four parallel flows during one official US109 demo slot.*

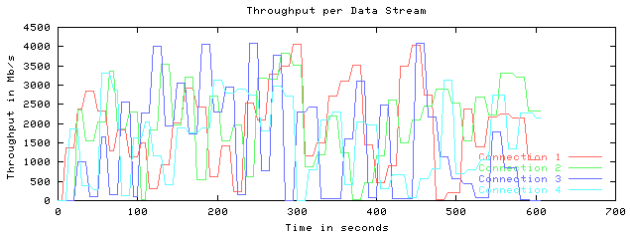

Figure 7. Per Flow Throughput of Streaming Data Mining (US109). *This figure shows the per flow throughput of computing histograms of four parallel flows during one official US109 demo slot.*

As we mentioned previously, after iGrid 2005, we applied the BESH algorithm to the SDSS data between Chicago and Tokyo, which were connected via 10Gb/s link. The data was transferred from Chicago to Tokyo in one stream. The SDSS histogram and the data transfer and processing speed are listed in Figure 8 and Figure 9, respectively. Figure 9 shows that an average of 3 Gb/s throughput had been reached.

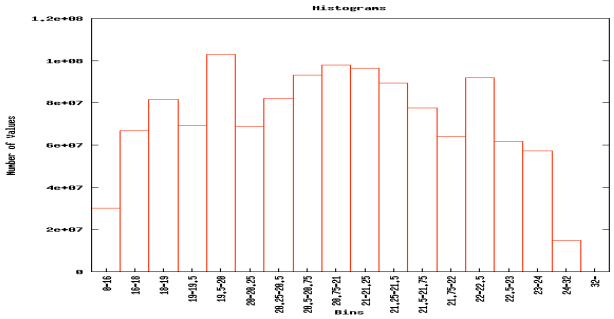

Figure 8. Analysis of SDSS data using BESH. *We transferred SDSS data from Chicago to Tokyo via 10Gb/s link and used BESH to analyze the distribution of the brightness of the stars. This figure shows the brightness histogram.*

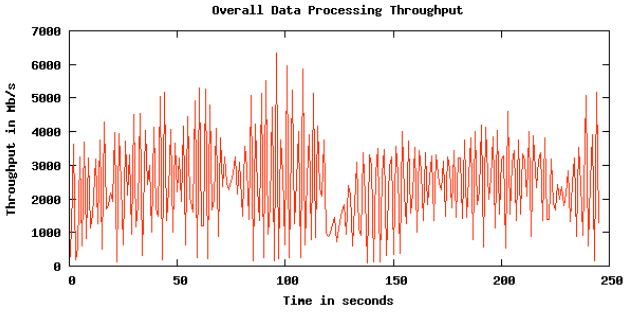

Figure 9 . Analysis of SDSS data using BESH. *We transferred SDSS data from Chicago to Tokyo via 10Gb/s link and used BESH*

to analyze the distribution of the brightness of the stars. This figure show the brightness histogram.

US121 Experimental Results. During the US121 demo, we transferred the entire BESTDR3 release of the SDSS data from the iGrid floor to nodes in Daejeon (Korea), Tokyo (Japan), and Chicago. The results are reported in Table 1. As mentioned above, the data consisted of 64 compressed files, each about 12.463 GBs, and in total comprising 797 GB of compressed data. When uncompressed, the data was about 1.5 TB.

For example, we transferred this data from San Diego to Daejeon in approximately 2.5 hours. The average transfer speed was 1027 Mb/s and the peak speed was over 1200 Mb/s. This was the first time that an astronomy data set of this size was transferred from disk to disk at this speed across the Pacific. With conventional networks and network protocols this transfer would not have been practical. A portion of the SDSS transfer throughput is shown in Figure 10.

Note that in this demonstration the disk IO speed is one of the major bottlenecks.

Table 1. This table summarizes three transfers of the Sloan Digital Sky Survey (SDSS) Release 3 data. *The data consisted of 64 files, each about 12.463 GB in size, and compromising about 797 GB in total. All results are reported in Mbps. The mean, median, standard deviation, minimum and maximum are computed from the 64 different transfers.*

Transfer: from iGrid to	Mean	Median	Standard Deviation	Min	Max
Chicago	653	712	255	128	1008
Kisti	1027	1160	229	312	1280
Tokyo	398	416	96	88	448

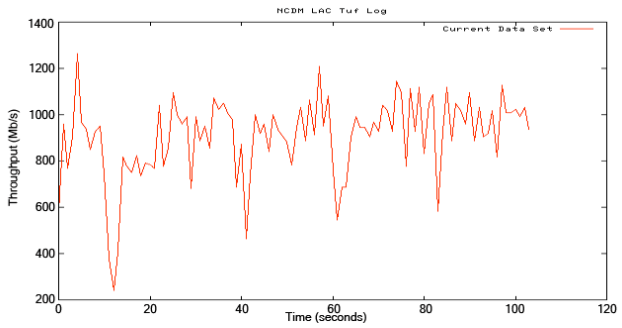

Figure 10. Throughout of SDSS data transfer from San Diego to KISTI, Korea. *This snapshot illustrates the disk-disk data transfer throughput (Mb/s) over time (seconds).*

6. RELATED WORK

Moving large data sets over high-speed wide area networks has been recognized as a challenging task for many years. During iGrid 2002, various groups demonstrated prototypes of several different tools for high performance data transport [2, 3, 9, 16, 18, 21].

Since then, various new data transport protocols or related congestion control algorithms [8, 10, 12] have been designed and developed. Comparison between different protocols is now commonly regarded as a complicated topic, as each protocol has both advantages and disadvantages and no single protocol has proved superior [15].

Since 1999, we have continued to develop a high performance data transport protocol based upon UDP: The first version was called SABUL [14]. SABUL used TCP as a control channel and was demonstrated at iGrid 2002. The next version [12] was completely implemented in UDP in order to gain efficiency. The current version [10] is called Composible-UDT and was used at iGrid 2005. One of the advantages of UDT is that it is easy to deploy since it can be deployed at the application level and does not require changing the kernel.

Perhaps the most widely deployed tool for bulk data transport is GridFTP [1], which uses parallel TCP to transfer data. In contrast, UDT uses UDP to transport the data and adds reliability and congestion control. We note that an upcoming release of GridFTP is expected to be integrated with a UDT driver enabling GridFTP to transfer data using UDT.

Tools for high performance data transport that have been widely adopted have tended to provide a more convenient user interface than that provided by a raw socket API. For this reason, UDT-Gateway provides a HTTP interface and hides the details of the UDT protocol.

We turn next to related work involving high volume data streams. Although streaming data mining is an area of active research, most of the work focuses on sensor networks and traditional Internet environments where the data transfer speed is much lower than what we saw during iGrid 2005.

Various approaches have been proposed for histogram computation on streaming data [5, 6, 17, 20]. These methods basically fall into two classes. One is to use different strategies to dynamically split and merge the buckets. The other is to construct a summary structure on the data stream and build histograms from the summary structure. Our method belongs to the first class. As far as we are aware, our implementation of the BESH algorithm over UDT is the first time that histograms have been computed on streaming data at the speeds seen at iGrid 2005.

More detailed analysis of histograms and streaming data processing is beyond the scope of this paper. Some general streaming data processing issues are discussed in [4, 7].

7. CONCLUSIONS

In this paper, we have described two demonstrations at iGrid 2005 that use data transport middleware and data mining middleware tools that we have developed.

For this first demonstration, we used the UDT-Gateway file transfer utility to transfer astronomical data from the iGrid 2005 conference to Korea. We transferred over 797 GB of data at a mean rate of 1027 Mb/s. This was the first time that we are aware of that astronomical data of this size has been transported across the Pacific.

For the second demonstration, we computed histograms on four high volume data flows that were streamed from Chicago, Korea, and Japan to the iGrid conference. We used an algorithm we designed and implemented called BESH. The average processing rate was about 8 Gb/s, with a peak speed of 14 Gb/s. This is one of the highest rates at which histograms have been computed on data distributed around the world that we are aware of.

Both applications were built over Composibile-UDT [10], a recent implementation of the UDT protocol that is composible in the sense that different congestion control algorithms can be easily implemented.

Finally, both of these demonstrations show the practicality of building useful, distributed data intensive applications using UDT-enabled middleware.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grant ANI 9977868, the Department of Energy under grant DE-FG02-04ER25639, and the U.S. Army Pantheon Project.

9. REFERENCES

[1] Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S. Data Management and Transfer in High Performance Computational Grid Environments. Parallel Computing. 2001.

[2] W. Allcock, J. Bresnahan, J. Bunn, S. Hegde, J. Insley, R. Kettimuthu, H. Newman, S. Ravot, T. Rimovsky, C. Steenberg, L. Winkler, "Grid-enabled particle physics event analysis: experiences using a 10 Gb, high-latency network for a high-energy physics application", Future Generation Computer Systems 19(6):983-997 August 2003

[3] A. Antony, J. Blom, C. de Laat, J. Lee and W. Sjouw. Microscopic Examination of TCP Flows over Transatlantic Links Future Generation Computer Systems, Volume 19(6), 2003.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems, *in Proc. of the 2002 ACM Symp. on Principles of Database Systems (PODS 2002),* June 2002.

[5] Donko Donjerkovic, Yannis E. Ioannidis, Raghu Ramakrishnan. *Dynamic Histograms: Capturing Evolving Data Sets,* Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 2000.

[6] Filippo Furfaro, Giuseppe M. Mazzeo, Domenico Saccà, Cristina Sirangelo: Hierarchical binary histograms for summarizing multi-dimensional data. SAC 2005: 598-603.

[7] Mohamed Medhat Gaber, Arkady B. Zaslavsky, Shonali Krishnaswamy: Mining data streams: a review. SIGMOD Record 34(2): 18-26 (2005).

[8] Mathieu Goutelle (editor), Yunhong Gu, Eric He (editor), Sanjay Hegde, Rajkumar Kettimuthu, Jason Leigh, Pascale Vicat-Blanc/Primet, Michael Welzl (editor), and Chaoyue Xiong. A Survey of Transport Protocols other than Standard TCP. In Global Grid Forum Data Transport Research Group. February 2004.

[9] Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong, Dave Lillethun, Jorge Levera, Joe Mambretti, Marco Mazzucco, and Jeremy Weinberger, Experimental Studies Using Photonic Data Services at IGrid 2002, Journal of Future Computer Systems, 2003, Volume 19, Number 6, pages 945-955.

[10] Yunhong Gu and Robert Grossman, Supporting Configurable Congestion Control in Data Transport Services, SC 2005, Seattle, Nov. 2005.

[11] Yunhong Gu and Robert Grossman, Optimizing UDP-based Protocol Implementation. PFLDNet 2005, Lyon, France, Feb. 2005.

[12] Yunhong Gu, Xinwei Hong, and Robert Grossman, Experiences in Design and Implementation of a High Performance Transport Protocol, SC 2004, Nov 6 - 12, Pittsburgh, PA, USA.

[13] Yunhong Gu, Xinwei Hong and Robert Grossman, An Analysis of AIMD Algorithms with Decreasing Increases, First Workshop on Networks for Grid Applications (Gridnets 2004), Oct. 29, San Jose, CA, USA.

[14] Yunhong Gu and Robert L. Grossman, SABUL: A Transport Protocol for Grid Computing, Journal of Grid Computing, 2003, Volume 1, Issue 4, pp. 377-386.

[15] Sangtae Ha, Yusung Kim, Long Le, Injong Rhee, and Lisong Xu, A Step toward Realistic Performance Evaluation of High-Speed TCP Variants, PFLDNet 2006, Nara, Japan.

[16] C. de Laat, E. Radius, S. Wallace, The rationale of current optical networking initiatives, Future Generation Computer Systems, Vol. 19, Number 6, August 2003, pp. 999-1008.

[17] G S Manku and R Motwani, Approximate Frequency Counts over Data Streams, VLDB 2002 (28th VLDB), p 346-357, August 2002.

[18] J. Mambretti, J. Weinberger, J. Chen, E. Bacon, F. Yeh, D. Lillethun, R. Grossman, Y. Gu, M. Mazzucco, The Photonic TeraStream: Enabling Next Generation Applications Through Intelligent Optical Networking at iGrid 2002, Journal of Future Computer Systems, Elsevier Press, Volume 19, Number 6, pages 897-908.

[19] A. Szalay, J. Gray, A. Thakar, P. Kuntz, T. Malik, J. Raddick, C. Stoughton. J. Vandenberg: *The SDSS SkyServer - Public Access to the Sloan Digital Sky Server Data*, ACM SIGMOD 2002.

[20] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In Proc. SIGMOD, 2002.

[21] Chong Zhang, Jason Leigh, Thomas A. DeFanti, Marco Mazzucco and Robert Grossman, TeraScope: Distributed Visual Data Mining of Terascale Data Sets Over Photonic

Networks, Journal of Future Computer Systems, 2003, Volume 19, Number 6, pages 935-943.

[22] Teraflow Testbed, http://www.teraflowtestbed.net.

[23] UDT: UDP-based Data Transfer Protocol, http://udt.sf.net.