Photonic Data Services: Integrating Data, Network and Path Services to Support Next Generation Data Mining Applications

Robert L. Grossman, Yunhong Gu,
Dave Hanley, Xinwei Hong, Jorge Levera and Marco Mazzucco
Laboratory for Advanced Computing
University of Illinois at Chicago

Dave Lillethun, Joe Mambretti, and Jeremy Weinberger
iCAIR
Northwestern University

May 11, 2004

This is a draft of the paper Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong, Dave Lillethun, Jorge Levera, Joe Mambretti, Marco Mazzucco, and Jeremy Weinberger, Photonic Data Services: Integrating Path, Network and Data Services to Support Next Generation Data Mining Applications, Data Mining: Next Generation Challenges and Future Directions, H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, editors, AAAI Press, 2004.

Abstract

We describe an architecture for next generation, distributed data mining systems which integrates data services to facilitate remote data analysis and distributed data mining, network protocol services for high performance data transport, and path services for optical paths. We also present experimental evidence using geoscience data that this architecture scales the remote analysis of Gigabyte size data sets over long haul, high performance networks.

1 Introduction

A fundamental challenge for next generation data mining is to develop systems for *remote* data analysis and *distributed* data mining which scale to large and

very large data sets. The data may be at *rest* in the sense that it resides on remote disks and tapes or it may be in *motion* in the sense that it is collected and streamed from a remote instrument.

The analysis and mining of this type of data is difficult for several reasons:

1. The majority of prior work has focused on agent-based systems in which local models are built on data in place and at rest, the models moved, and then combined at a central location.

2. TCP-based data transport itself becomes a bottleneck when working with remote and distributed data, even in high bandwidth networks, due to TCP's current congestion control mechanism [1]. This is because the congestion control mechanism for TCP behaves poorly for flows over links with a high bandwidth delay product. The bandwidth delay product can be approximated by multiplying the round trip time for a packet by the maximum bandwidth of the least capable link that a packet transverses. As an example, a standard TCP flow over a 155 Mb/s OC-3 link connecting Chicago and Amsterdam is in practice limited to about 5 Mb/s unless extensive network tuning is undertaken.

3. In the last few years, we have gained a better understanding of the primitives required to integrate data mining with databases. On the other hand, we do not yet have a good understanding of the primitives required for distributed, high performance data mining.

In this paper, we introduce an architecture for remote data analysis and distributed data mining which integrates services to set up optical paths, network protocols designed for high performance networks, and data services supporting the remote analysis and distributed mining of large data sets. We also show experimentally the speedup gained with this approach for some typical data mining algorithms such as computing simple correlations for streaming data.

We believe that the work described here is novel for the following reasons:

1. This is the first paper that we know of describing an architecture for integrating a) data services supporting primitives to facilitate remote data analysis and distributed data mining, b) network protocols designed to move packets efficiently over high performance networks, and c) services designed to set up paths on demand for photonic networks when required by applications. Integrated services like these can provide the foundation for scaling distributed data mining to large data sets. We call this architecture Photonic Data Services.

2. This is the first experimental study that we are aware of demonstrating the feasibility of the *distributed* mining of Gigabyte size data sets that are separated by thousands of miles and over a hundred milliseconds in packet round trip time.

In Section 2, we describe related work. In Section 3, we describe the basic idea. In Section 4, we describe the architecture we introduce called Photonic Data Services or PDS. In Sections 5-7, we describe the three main layers of PDS. In Section 8, we describe the testbed we use for our experiments. In Section 9, we describe our experimental studies involving distributed mining of geoscience data. Section 10 is the summary and conclusion.

We are currently continuing the experiments described in this paper and preparing an expanded version of this paper [13].

2 Background and Related Work

In this paper, we are concerned with supporting remote data analysis and distributed data mining applications with high performance data transport services. In addition, many applications will also require high performance compute services, which we do not address. Today, these would be typically provided by local compute clusters or by virtual compute clusters accessed via a computational grid [9] or computational middleware architecture [29].

There have been three main architectural approaches to date for distributed data mining: agent based systems, data grid based systems, and data web based systems. We consider each in turn.

The first approach is to use agents over commodity networks to move data, remotely control the data mining algorithms at the different sites, and to collect the intermediate results and models. Systems with this architecture include the JAM system developed by Stolfo et al. [33], the BODHI system developed by Kargupta et al. [20], the Kensington system developed by Guo et al. [5], and the Papyrus system developed by Grossman et al. [12].

The second approach is to use cluster middleware. Systems with this architecture include those developed by Subramonian et al. [30], Moore et al. [27], and Grossman et al. [12]. More recently, Globus has emerged as the dominant middleware for working with distributed clusters [9]. The Globus infrastructure for data intensive computing is called the data grid, and includes services for parallel TCP striping (GridFTP), and data replication services (Globus Replica Catalog and Globus Replica Management) [6].

Other grid middleware services that have been used for data mining include the DataCutter developed by Saltz et al. [2] and Discovery Net developed by Guo et al. [32]. For example, Du and Agrawal recently used the DataCutter for some distributed data mining experiments [7].

The third approach, and the one described in this paper, is to use data webs which are web based infrastructures for data [10]. Unlike grid middleware which is built over authentication, authorization and access (AAA) control mechanisms for rationing and scheduling presumably scarce high performance computing resources [9], data webs are built using W3C standards and emerging standards for web services and packaging (SOAP and XML). Data webs in contrast to data grids are designed to encourage the open sharing of data resources without AAA controls, in the same way that the web today encourages the sharing of

document resources without AAA controls [34].

For small data sets, data webs use W3C standards and emerging standards to manage both the data and metadata. These include HTTP, DWTP (Data Web Transport Protocol) and other emerging standards for transport [11], and SOAP and XML for packaging [34]. For large data sets, this infrastructure is used just for the *meta-data*, while specialized network protocols and data services (the photonic data services described below) are used to manage the *data* itself. Providing separate mechanisms for control paths and data paths is an old idea in high performance computing going back to at least the IBM High Performance Storage System (HPSS). Developing the appropriate data web services and protocols to work with large remote and distributed data is a fundamental research challenge.

As mentioned above, the performance of data flows with large bandwidth delay products (BDP) is usually quite poor in practice [1].

There have been several approaches for dealing with this problem. One approach to improving TCP performance for data intensive applications is to adjust the TCP window size to be the product of the bandwidth and the RTT delay of the network [18]. This approach requires modifying and tuning the kernel of each of the operating systems transporting the packets and ensuring that the networking hardware can support these large or jumbo packets.

Another approach to overcoming the limitations of TCP is to stripe TCP over several standard TCP network connections. In contrast to the first approach, this can be done at the data middleware or application level. This approach has been implemented several times, including PSockets [16] and GridFTP [3]. It has been observed that effectively utilizing high performance links can require dozens to hundreds of sockets. This can create an overhead, limiting the usefulness of this approach. In addition, the window size must be carefully tuned, as with the first approach.

Another approach is to use a protocol that combines a UDP-based data channel with a TCP-based control channel. UDP can effectively transport data at high rates even over high BDP paths. The TCP control channel can be used to create a reliable algorithm, while the appropriate rate and congestion control algorithms can be used in the control channel so that the algorithm is friendly to other flows. PDS uses a protocol called SABUL [15], which takes this approach and is described in more detail below. Since SABUL is based upon TCP and UDP, it can be deployed as an application library without making any changes to the existing network infrastructure.

Another approach is to improve TCP in various ways. High Speed TCP [8], Scalable TCP [22], and FAST [19] are examples of this approach. Although these approaches all appear to be promising, work is still required to understand their friendliness, performance, and scalability. In addition, new TCP variants require significant changes to the current network infrastructure.

Another approach is to create entirely new protocols, such as Explicit Control Protocol (XCP) [21] and the Datagram Congestion Control Protocol (DCCP) [23]. Again, deploying these new protocols will take some time due to the significant changes required to the current network infrastructure.

3 The Basic Idea

Today data intensive applications working with remote and distributed data are generally based upon standard networking (IP) and transport (TCP) protocols. For data mining applications running on commodity networks analyzing small data sets these protocols work extremely well. Data mining applications involving large, distributed data sets have generally used specialized networks such as NSF's vBNS network or the Internet 2's Abilene network. In practice, very large bandwidth applications have to be scheduled on these networks and require the use of specialized transport protocols [16].

As optical networking architectures become more common, a new possibility is emerging. A bandwidth demanding application can request an optical connection between the data sources and the data sinks for a specific application. More specifically, the application can request the set up, status and tear down of the required optical paths. Clearly there is a cross over point: for short transfers of small data, TCP is clearly preferable, while for long transfers of very large data, a dedicated optical path might be preferable.

For the purposes here, the layered network model we use is a standard extension of the standard 5-layer model in which we split the top layer into a layer providing specialized data services for remote data analysis and distributed data mining and a top application layer. More generally, two additional layers would be added between layers 5 and 6 below: one for the description of data services (for example, WSDL) and one for the discovery of data services (for example, UDDI) [34].

1. Physical Links. We assume that the physical links are provided by multi-channel wavelength-division multiplexed (WDM) communications, as well as by Ethernet, and other technologies.

2. Path Services Layer. We assume that there are services allowing us to set up paths between devices, tear down paths, check the status of paths, set up routing, etc.

3. Internet Layer. This layer provides a common network addressing and routing across multiple networks. For our applications, we use the Internet Protocol (IP) in this layer.

4. Network Protocol Services Layer. We assume that there are transport services including TCP, UDP, and other more specialized protocols providing high performance over the paths. Our applications use specialized high performance protocols in this layer.

5. Data Services Layer. We assume that there are standard services for moving data such as SOAP-based web services, as well as more specialized data services designed for performance networks.

6. Application Layer. We assume that the remote data analysis and distributed data mining applications can request standard and specialized network services depending upon the applications' requirements.

In this paper, we describe specialized integrated services for layers 2, 3 and 5 and illustrate their use on the analysis of distributed geoscience data.

Layer	Description	Implementation
6	Application	DWTP applications
5	Data Services	SOAP, DWTP
4	Network Protocol	TCP, UDP, SABUL
3	Internet Protocol	IP
2	Path Services	ODIN
1	Physical Links	WDM, Ethernet, ...

4 Photonic Data Services

In this paper, we introduce the idea of integrating 1) specialized photonic path services; 2) high performance network protocols and 3) high performance data services providing data mining primitives for remote data analysis and distributed data mining. We call these integrated services *photonic data services* or PDS.

As an example, consider a distributed data mining application in which 1.8 GB of vegetation data over a region specified by latitude and longitude coordinates will be correlated with 1.8 GB of climate data over the same region. Assume both data sets are in the US in different locations, but that the client doing the correlation is in Amsterdam.

Assume that both data sources are connected to the client by an OC-12 network operating at 622 Mb/s. Today, the data would be moved to a common location using a standard network protocol such as TCP, merged, and then correlated. Without the correlation, this process takes over 3000 seconds, as we will see in Section 9.

In general an experimental OC-12 is not available. Using the photonic data services described below, a photonic path can be set up in less than a minute and the two 1.8 GB streams transported and merged in less than 70 seconds, as we will see in Section 9. As the path services software matures, we expect the minute set up time to be reduced substantially, so that a data mining computation that today requires about an hour could be done in about a minute.

In the next three sections, we describe the three service layers we have implemented and integrated to create photonic data services to support data mining. Our implementation of the path services is called ODIN [24]; our implementation of the network protocol services is called SABUL [15]; our implementation of the streaming merge which is the data service or data mining primitive for the example above is called the continuously generated merge or CGM [26]. The work described in this paper is the first time we have integrated these three service layers and performed experimental studies using them.

5 Path Services

The path services used in PDS are called the Optical Dynamic Intelligent Network Service Layer or ODIN [24]. We now describe these systems following [24]. ODIN receives requests for circuits by applications, which for PDS are usually from the data service layer, and contacts the required network switches, including both optical-domain DWDM switches and traditional Ethernet switches and IP routers, to set up the circuits. ODIN also provides services to tear down paths and to check their status.

ODIN consists of two sub-systems: one, called the TeraScale High Performance Optical Resource Regulator or THOR, interfaces to the optical fabric; while the other, called the Dynamic Ethernet Intelligent Transit Interface or DEITI, interfaces to the traditional Ethernet/IP fabric.

ODIN is designed to dynamically provision and control global light paths. The ODIN subsystem THOR is based on new signaling methods for dynamically provisioning light paths. These light paths can be used to create optical VPNs (OVPNs), as well as to extend these light paths to edge resources through other types of dynamically provisioned paths, such as vLANs.

Currently, ODIN sets up paths only within a single administrative domain. In future work, similar path services are planned for multiple administrative domains.

6 Network Protocol Services

In this section, we describe a network protocol designed for high performance data transfer called the Simple Available Bandwidth Utilization Library or SABUL following [15]. We emphasize that several of the other network protocols mentioned above could also be used. We chose to use SABUL since as an application level library no change to the existing network infrastructure was required. In addition, SABUL does not require the sometimes delicate tuning required by IETF RFC 1323.

The idea behind SABUL is simple. SABUL combines the UDP protocol in order to send data at a high rate with the TCP protocol in order to do this in a reliable fashion. UDP has no flow control, rate control, or reliable transmission mechanisms. SABUL implements these control functions in a separate TCP control channel. This approach is in contrast to the approach of other high performance protocols such as NETBLT [4] which combine the data and control channels.

In SABUL, the packets on the UDP channel consist of the usual UDP header plus a 32 bit field for a sequence number. On the TCP channel, each packet consists of: a list of lost data packets, a field stating the requested data rate, and a field reserved to report the state of the receiver's available buffer size. We define the *communication state* information to be the information contained in these TCP packets.

The flow is assumed to be unidirectional. Data is sent to the receiver over

the UDP channel, while current communication state information is sent over the TCP channel, from the receiver to the sender. Since the communication state information is passed over TCP, its arrival is ensured; since the amount of this information is relatively small, it has a negligible effect on the overall performance of SABUL.

One of the advantages of SABUL is its continuous updating of state information. In contrast, NETBLT uses a mechanism that sends buffers of data at a fixed rate. At the end of transmission of each buffer, the receiving side of NETBLT sends the sender a list of packets that were lost in the transmission of this buffer. The sender then resends these packets; the process continues until all packets in the buffer are accounted for. Then the next buffer can be transmitted by NETBLT. NETBLT needs to block until all packets are accounted for on the sending side before sending another buffer. This process can be further delayed since packet loss information is transmitted unreliably by the receiver to the sender (since this information is sent over UDP). Another deficit of NETBLT is that it needs to wait for at least one round trip time to get each update of packets lost.

In SABUL, however, each time the receiver notices at least one missing packet, it uses the TCP channel to transmit to the sender a list of packets that were lost. It does not have to block the sending of packets over the UDP channel to wait for an incoming packet containing the communication state information. This allows for changing the rate and flow of data, and retransmission of any missing packets during the transmission of the data. The list of missing packets is updated every time a missing packet is received. If during a predefined amount of time no packet was lost, and thus no transmission sent to the sender on the TCP channel, the receiver sends a notification of this fact to the sender with communication state information. This allows the sender to empty its buffer of packets which have successfully been received and adjust the rate and flow if necessary.

7 Data Services

In this section, we describe data services designed to be component services or primitives for distributed data mining applications. This section is adapted from [11].

The data model, access model, and query model for PDS are based upon data webs. Data webs are web based infrastructures designed to facilitate the analysis and mining of remote and distributed data [11]. Data webs use a protocol for working with remote data called the Data Web Transfer Protocol or DWTP (formerly known as the DataSpace Transfer Protocol or DSTP) in the same way that the standard web uses HTTP to access remote documents. Data webs also support access to remote data using SOAP as the packaging protocol.

The PDS experiments described below use a data web implementation called DataSpace, which we sketch briefly below [11].

- *Distributed Columns of Numerical Data.* The data model for PDS is simple. Data is divided into rows (data records) and columns (data fields or data attributes). Both may be distributed over the web. Access to the data itself is through a DWTP server. The current DataSpace DWTP servers can also access data using SOAP. Physically, the data itself may be stored as files, in databases, or using other specialized storage mechanisms. Logically, data is just a distributed collection of columns.

- *Universal Correlation Keys.* A Universal Correlation Keys (UCK) is a globally unique id (GUID) and is used for relating columns of data on two different DWTP servers. Each column of data is associated with at least one column of UCKs.

- *Multi-Dimensional UCKs.* UCKs may be combined to provide multi-dimensional keys. This is essential for working with scientific and engineering data, such as the geoscience data used in the experiments below. For example, this data uses latitude and longitude as the UCKs.

- *Column Based Meta-Data.* Associated with each column of data is attribute meta-data and with each data set (a collection of columns) data set meta-data. DWTP applications may or may not use this meta-data. On the other hand, this meta-data is essential for building and deploying statistical models. DWTP servers provide a simple mechanism for associating metadata to columns and collections of columns.

Universal correlation keys enable distributed columns to be correlated in the following fashion: Pairs (k_i, x_i), where k_i is a UCK value and x_i is an attribute value, on DWTP Server 1 can be combined with pairs (k_j, y_j) on DWTP Server 2 to produce a table (x_k, y_k) in a DWTP client. The DWTP client can then, for example, find a function $y = f(x)$ relating x and y. This simple mechanism of distributed columns identified by UCKs (perhaps vector valued) is sufficient information for many data mining algorithms.

Depending upon the request, DWTP servers may return one or more columns, one or more rows, or entire tables. DWTP uses XML to describe the metadata. On the other hand, for efficiency and scalability, by default data *itself* is transmitted as records delimited by carriage returns, with fields delimited by commas. As an alternative, data may also be transmitted using SOAP. The DWTP client may also indicate that a specialized high performance protocol such as SABUL should be used for the data channel. To summarize, the DWTP protocol uses XML for metadata and small data, while data is typically streamed, with large amounts of data streamed using SABUL or other high performance network protocols.

The DWTP protocol includes commands for retrieving metadata, retrieving UCKs, retrieving data and subsets of data, and mechanisms for sampling, working with missing data, and merging by UCKs.

8 Physical Testbed

We assume that our network consists of Dense Wavelength-Division Multiplexed optical devices together with standard Ethernet/IP devices. For our experiments we used the Chicago area OMNInet [25] and the global Terra Wide Data Mining Testbed [35].

OMNInet is an optical networking testbed deployed in the Chicago metropolitan area. OMNInet currently provides 1 GE and 10 GE services between Northwestern, the University of Illinois at Chicago, and the StarLight facility in Chicago. OMNInet is operated by a research consortium consisting of iCAIR at Northwestern, the Electronic Visualization Laboratory at the University of Illinois at Chicago, Argonne National Laboratory, SBC, and Nortel.

The Terra Wide Data Mining Testbed (TWDM) is a testbed built on top of DataSpace for the remote analysis, distributed mining, and real time exploration of scientific, engineering, business, and other complex data. Currently, the TWDM Testbed consists of five geographically distributed workstation clusters linked by optical networks through StarLight in Chicago. These sites include StarLight itself, the Laboratory for Advanced Computing at UIC, iCAIR at Northwestern University, SARA in Amsterdam, and Dalhousie University in Halifax. SARA is connected to StarLight via the Netherlands' Surfnet network and Dalhousie is connected to StarLight via Canada's CANARIE network.

The experimental setup was as follows. Data servers were located at the SARA research facility in Amsterdam and at the University of Illinois at Chicago and connected via an OC-12 network. The merge was done at the StarLight Facility in Chicago. StarLight and the University of Illinois at Chicago are located several miles a part. The machine performing the distributed merge was connected by OC-12 paths to both remote data sources.

The machine in Amsterdam was a dual P4, 1700 Mhz, with 512M RAM. The machines in Chicago were dual PIIIs, 1000Mhz, with 512M RAM. The machines were all running Linux, with the 2.4.x kernels. The network traffic was over SurfNet and OMNInet with routing providing 622 Mb/s of maximum bandwidth.

9 PDS Application: Lambda Joins

In this section, we describe a sample distributed data mining application developed using photonic data services. The core application is the merging of distributed geoscience data from NCAR [28]. We have described previously the remote analysis of small NCAR data sets with DWTP clients and servers [11] over the commodity internet

For the work described in this section, we integrated the data services provided by DWTP servers, the network protocol services provided by SABUL, and the path services provided by ODIN. To support the correlation of distributed data in the NCAR format, we separately developed a continuous merge algorithm for streaming data over high performance networks called the Con-

tinuously Generated Merge or CGM [26]. Once the streaming data has been merged, simple counts using a finite buffer can be done in a variety of ways [14].

This approach is quite general and, for example, could be used to merge and do simple analyses of other distributed data using multi-dimensional keys.

Continuously Generated Merge (CGM) Algorithm. We now briefly review the CGM algorithm following [26]. In the CGM algorithm we assume the data is partially presorted. Without loss of generality, assume there are two data streams, A and B, being drawn into a client in approximately ascending order and we are trying to merge on one UCK. The CGM algorithm depends upon two parameters: a parameter N determining the number of records in a window, which is used to buffer the streaming data, and N_h, the number of entries in two auxiliary hash tables. The algorithm has an even step and an odd step. The even steps of the algorithm are as follows:

1. The client grabs some fixed number of records N, from both stream A and stream B and places them in window A and window B respectively (each has room for exactly N records).

2. A hash is done on the value of each UCK in window A and the record is placed in the appropriate location in hash table A, overwriting any previous record.

3. A hash is done on the value of each UCK in window B and if the value hashes to an occupied location in hash table A, both the records are merged. If the value does not hash to an occupied location in hash table A, then the record is placed in the appropriate location in hash table B, overwriting any previous record.

In the odd steps of the algorithm, the above algorithm is executed, but reversing the roles of A and B.

Experimental Results. The first results below are from the CGM algorithm running using TCP as the network protocol and DWTP as the data service protocol. Each data stream was 300 MB in size. The CGM algorithm used a hash table size of 50,000 and a window size of 10,000. The data was atmospheric data from NCAR. The randomization was done by replacing every n'th row (for example, for 10 percent every 10th row) with a random row which was within 50,000 lines of the current row.

As can be seen from the table, the average speed varied between 4-5 Mb/s, despite the fact that each link had a maximum available bandwidth of 622 Mb/s. We note that this type of result is typical.

Rand %	Match %	Time *sec*	Data Rate *Mb/s*
2	96.6	513	4.68
10	89.9	540	4.44
20	81.5	531	4.52
33	73.1	563	4.26

The next two results below are from the CGM algorithm running SABUL as the network protocol, DWTP as the data service protocol, and ODIN to provide path service. In this experiment, ODIN was called statically, not dynamically by the application. *The data size this time was 1.8 GBs so that in total 3.6 GBs of data were merged by the algorithm.* The average speed varies between 400-500 Mb/s. This means that CGM over SABUL was about 600x faster on average, since the amount of data was 6x greater and the elapsed time was about 100x greater.

When testing the algorithm we realized the largest single affect on the performance of the merge was the length of the record. The longer the record size the memory copying required, the greater the merge time. To illustrate this we ran two tests. In the first, both data files contain 1 UCK and 1 attribute; in the second, both data files contain 1 UCK and 7 attributes.

Rand %	Match %	Time *sec*	Data Rate *Mb/s*
2	99	53.3	550
10	91	52.4	550
20	83	56.2	512
33	78	54.6	527

Rand %	Match %	Time *sec*	Data Rate *Mb/s*
2	99	66.3	434
10	92	65.7	438
20	82	64.2	449
33	79	65.1	442

10 Summary and Conclusion

In this paper, we have introduced an architecture called Photonic Data Services or PDS which integrates data services, network protocol services, and path servers. With intelligent path services, distributed data mining applications can intelligently signal for a special photonic path, use this for distributed data mining, and then release it for use by other applications. With high performance network protocols, data mining applications can work effectively with remote Gigabyte size data sets over high performance networks. These types of protocols are sometimes several hundred times faster than traditional protocols over the same high performance networks. With specialized data services such as streaming merges, distributed data mining services can effectively correlate distributed Gigabyte size data sets.

In this paper, we have provided experimental evidence that our implementations scale to remote Gigabyte size data sets that can be distributed over thousands of miles and accessed via long haul networks with packet round trip times (RTT) of a hundred milliseconds or more. Compared to current implementations of data mining primitives for merging two data streams and computing counts, our Photonic Data Services are significantly faster. For example, to

stream two 1.8 Gigabyte data streams of geoscience data using latitude and longitude as keys across the Atlantic, merge the results by key, and compute simple counts required over an hour with conventional services and less than a minute using the photonic data services described in this paper. We emphasize that both experiments used the same high performance network.

Data webs built with this architecture complement data grids which require authentication, authorization and access controls supplied by Globus and other grid middleware, and the various custom agent based distributed data mining systems which have developed over the past several years. Data webs, and indeed data grids, built over photonic data services are one means of meeting the challenges posed by the large distributed and streaming data sets which will become more common with next generation data mining applications.

References

[1] Joseph Bannister, Andrew Chien, Ted Faber, Aaron Falk, Robert Grossman, and Jason Leigh. Transport protocols for high performance: Whither tcp? *Communications of the ACM*, to appear.

[2] M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J.Saltz. Distributed processing of very large datasets with datacutter. *Parallel Computing*, 27(11):1457–1478, 2001.

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

[4] D. Clark, M. Lambert, and L. Zhang. Netblt: A high throughput transport protocol. *Frontiers in Computer Communications Technology: Proceedings of the ACM-SIGCOMM '87*, pages 353–359, 1987.

[5] John Darlington, Yike Guo, Janjao Sutiwaraphun, and Hing Wing To. Parallel induction algorithms for data mining. *Lecture Notes in Computer Science*, 1280, 1997.

[6] Globus data grid. Retrieved from http://www.globus.org/datagrid/, September 2, 2002.

[7] Wei Du and Gagan Agrawal. Using general grid tools and compiler technology for distributed data mining: Preliminary report. In Parthasarathy et al. [31], pages 51–61.

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

[9] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, California, 1999.

[10] Robert Grossman, Mark Hornick, and Gregor Meyer. Data mining standards initiatives. *Communications of the ACM*, 45(8):59–61, 2002.

[11] Robert Grossman and Marco Mazzucco. Dataspace - a web infrastructure for the exploratory analysis and mining of data. *IEEE Computing in Science and Engineering*, 2002.

[12] Robert L. Grossman, Stuart Bailey, A. Ramu, Balinder Malhi, Harinath Sivakumar, and Andrei Turinsky. Papyrus: A system for data mining over local and wide area clusters and super-clusters. In *Proceedings of Supercomputing 1999*. IEEE and ACM, 1999.

[13] 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*, to appear.

[14] Robert L. Grossman, Jorge Levera, and Marco Mazzucco. Aggregate queries on streams of data using a small buffer. UIC Laboratory for Advanced Computing Technical Report, 2002.

[15] Robert L. Grossman, Marco Mazzucco, Harinath Sivakumar, and Yiting Pan. Simple available bandwidth utilization library for high-speed wide area networks. *Journal of Supercomputing*, to appear.

[16] Robert L Grossman, Harinath Sivakumar, and S. Bailey. Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Supercomputing*. IEEE and ACM, 2000.

[17] D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors. *Proceedings the Third International Conference on the Knowledge Discovery and Data Mining*, Menlo Park, California, 1997. AAAI Press.

[18] V. Jacobson, R. Braden, and D. Borman. Tcp extensions for high performance. *IETF RFC 1323*, May, 1992.

[19] 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. Singh. Fast tcp: From theory to experiments. *IEEE Communications Magazine*, submitted for publication.

[20] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining using an agent based architecture. In Heckerman et al. [17], pages 211–214.

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

[22] Tom Kelly. Scalable tcp: Improving performance in highspeed wide area networks. *submitted for publication*, 2002.

[23] Eddie Kohler, Mark Handley, Sally Floyd, and Jitendra Padhye. Datagram congestion control protocol (dccp). http://www.icir.org/kohler/dcp/, retrieved on January 10, 2003.

[24] Dave Lillethun, Joe Mambretti, and Jeremy Weinberger. Odin: Path services for optical networks, in preparation. *www.icair.org*, 2002.

[25] Joel Mambretti. Omninet *www.icair.org/omninet*, 2002.

[26] Marco Mazzucco, Asvin Ananthanarayan, Robert L. Grossman, Jorge Levera, and Gokulnath Bhagavantha Rao. Merging multiple data streams on common keys over high performance networks. In *Proceedings of Supercomputing 2002*. IEEE and ACM, 2002.

[27] R. W. Moore, C. Baru, R. Marciano, A. Rajasekar, and M. Wan. Data-intensive computing. In *The Grid: Blueprint for a New Computing Infrastructure* [9], pages 105–129.

[28] National Center for Atmospheric Research, Community Climate Model. Retrieved from www.cgd.ucar.edu/cms/ccm3/, April 10, 2002.

[29] NSF middleware initiative. Retrieved from www.nsf-middleware.org, September 2, 2002.

[30] S. Parthasarathy and R. Subramonian. Facilitating data mining on a network of workstations. *Advances in Distributed and Parallel Knowledge Discovery*, 2000.

[31] Srinvasan Parthasarathy, Hillol Kargupta, Vipin Kumar, David Skillicorn, and Mohammed Zaki, editors. *High Performance Data Mining*, Philadelphia, Pennsylvania, 2002. SIAM.

[32] Patrick and Yike Guo. The design of a platform for distributed kdd components. In Parthasarathy et al. [31], pages 63–78.

[33] S. Stolfo, A. L. Prodromidis, and P. K. Chan. Jam: Java agents for meta-learning over distributed databases. In Heckerman et al. [17].

[34] W3c semantic web. Retrieved from www.w3.org/2001/sw/, September 2, 2002.

[35] Terra wide data mining testbed. Retrieved from www.ncdm.uic.edu/testbeds.htm, September 2, 2002.