Experimental Studies Using Photonic Data Services at IGrid 2002

Robert L. Grossman and Yunhong Gu and Don Hamelburg
and Dave Hanley and Xinwei Hong and Jorge Levera
and Dave Lillethun and Marco Mazzucco and Joe Mambretti
and Jeremy Weinberger

December, 2002

Abstract

We describe an architecture for remote and distributed data intensive applications which integrates optical path services, network protocol services for high performance data transport, and data services for remote data analysis and distributed data mining. We also present experimental evidence using geoscience data that this architecture scales to long haul, high performance networks.

1 Introduction

A fundamental research challenge is to develop systems for remote data analysis and distributed data mining which scale to 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. For fixed window size, the bandwidth of reliable network protocols such as TCP, which requires acknowledgments for each packet, is roughly approximated by $1/RTT$, where RTT is the round trip time for a packet to travel between the client and server. For an application across the Atlantic that gives a rough limit of approximately 5 Mb/s, even if the path has a capacity many times greater, such as 1 Gb/s link.

2. 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 remote data analysis and distributed 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 providing simple primitives supporting the remote analysis and distributed mining of large data sets. We also describe experiments showing the speedup gained with this approach for some typical data mining algorithms such as computing simple correlations for streaming geoscience data.

In Section 2, we describe related work. In Section 3, we describe the architecture we introduce called Photonic Data Services or PDS. In Sections 4–6, we describe the three main layers of PDS. In Section 7, we describe two experimental studies involving PDS applications. Section 8 is the summary and conclusion.

A preliminary version of this paper describing a prior version of the architecture and experiments with an OC-12 network appeared in the conference proceedings [1].

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 [2] or computational middleware architecture [3].

It has long been recognized that TCP is not appropriate for high performance applications on long haul networks. The reason is simple: the TCP protocol requires acknowledgment of each packet. This limits the bandwidth to be a function of $1/RTT$, where RTT is the time required to send a packet and receive an acknowledgment [4], page 186.

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 [4]. 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 [5] and GridFTP [6]. It has been observed that the performance of striped TCP begins to level off after about 12-20 sockets, effectively limiting its usefulness to OC-3 or OC-12 networks.

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. [7], the BODHI system developed by Kargupta et. al. [8], the Kensington system developed by Guo et. al. [9], and the Papyrus system developed by Grossman et. al. [10].

The second approach is to use grid-based middleware. Systems with this architecture include those developed by Subramonian et. al. [11], Moore et. al. [12], and Grossman et. al. [10]. More recently, Globus has emerged as the dominant middleware for working with distributed clusters [2]. 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) [13].

Other grid middleware services that have been used for data mining include the DataCutter developed by Saltz et. al. [14] and Discovery Net developed by Guo et. al. [15]. For example, Du and Agrawal recently used the DataCutter for some distributed data mining experiments [16].

The third approach and the one used in this paper is to use data webs, which are web based infrastructures for data [17]. Unlike grid middleware which is built over authentication, authorization and access (AAA) control mechanisms for rationing and scheduling presumably scarce high performance computing resources [2], 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 [18].

For small data sets, data webs use W3C standards and emerging standards to manage both the data and metadata. These include HTTP, DSTP (DataSpace Transport Protocol) and other emerging standards for transport [19], and SOAP and XML for packaging [18]. 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.

3 Photonic Data Services

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 [5].

This paper describes applications based upon the following specialized services:

Intelligent signaling for paths by applications. 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 is clearly preferable.

With today's MEMS technology, reconfiguration of a light path in an optical add/drop multiplexer (OADMs) or optical cross connects (OXCs)can be done in milliseconds. New cross connect technologies such as semiconductor optical amplifiers (SOAs) would shorten this to nanoseconds [20]. Today, as we will see below, applications requiring moving > 0.5 GBs for remote data analysis and distributed data mining can benefit from requesting specialized lambda paths. In the future, the cross over point will be much smaller.

Specialized network protocols. It is clear that TCP does not currently perform well for moving large data sets over long distances. Recently there has been a flurry of activity investigating alternative protocols [5], [6], [21], [13], and [22]. Our assumption is that one or more of these alternative protocols will be commonly used by data intensive applications, supplementing TCP when required.

Specialized data services. Moving data is different than moving bits. In addition, we assume that data intensive applications will also make use of specialized protocols and services for working with data, services which support metadata operations, record and attribute selection, missing data, sampling, and related 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, in Section 7.3, we describe a distributed data mining application in which 1.8 GB of vegetation data over a region specified by latitude and longitude coordinates is correlated with 1.8 GB of climate data over the same region. Both data sets are in the US in different locations, but the client doing the correlation is in Amsterdam.

Assume that both data sources are connected to the client by 10 Gb/s links. Today, the data would be moved to a common location using a standard network protocol such as TCP, merged, and then correlated. Merely moving the data across the Atlantic takes over 3000 seconds, as we will see in Section 7.3.

Our assumption is that in the future dedicated links will not be common, but the ability to set up specialized photonic paths on a per application basis will be. 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 7.3. As the path services software matures, we expect the set up time to be reduced substantially, so that a data mining computation that today requires approximately an hour could be done in approximately a minute.

For the purposes here, the layered network model we use is a simple extension of the standard 5-layer model in which we split the top layer into two. One of these provides specialized data services for remote data analysis and distributed data mining and the other is the top application layer. Here is the layered model we are using:

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 the next three sections, we describe the three service layers we have implemented and integrated to create photonic data services to support data analysis and data mining. Our implementation of the path services (Layer 2) is called ODIN [23]; our implementation of the network protocol services (Layer 4) is called SABUL [22]; our implementation of data services (Layer 5) uses high performance implementations of the DataSpace Transport Protocol (DSTP) we have developed [24]. Using these three services, we developed an application for merging two high bandwidth data streams of earth science data employing a join algorithm called the continuously generated merge or CGM [25].

The work described in this paper is the first time we have integrated these three service layers and performed experimental studies using them.

Integrating Photonic Data Services with web services is straightforward and will be described in a forthcoming publication. To do this, we simply added two additional layers between PDS layers 5 and 6: one for the description of data services (for example, WSDL) and one for the discovery of data services (for example, UDDI) [18].

4 Path Services - ODIN

The path services used in PDS are called the Optical Dynamic Intelligent Network Service Layer or ODIN [23]. ODIN receives requests for circuits by applications, which for PDS can be from layer 4 or 5 services, 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 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. We now describe these systems following [23].

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 vLANs.

ODIN can be used to establish "services on demand" and, as noted, not only dynamically allocated light paths, but also dynamically allocated transient (or permanent) Optical Virtual Private Network (OVPNs). In part, ODIN accomplishes its functions through interactions with lower layer optical services such as THOR.

ODIN is designed to work within a single administrative domain and to provide a predefined set of path services. Since the only allocation of paths is through ODIN, it has complete knowledge of the topology and current resource allocations within the administrative domain. ODIN accepts requests for path services from services and applications over the network. When resources are allocated to fulfill requests, ODIN communicates with the requisite network switches to configure them as required. These switches can be optical-domain DWDM switches, Ethernet switches and/or IP routers.

With the current implementation of PDS, layer 4 or 5 services determine whether a photonic path is required. If so, a request is sent to the ODIN server. The ODIN server 1) finds the closest optical endpoints for the specific source and destination; 2) establishes the optical connection; and 3) activates other related network components to provide the required upper layer services, such as MPLS and Tagged VLAN switching.

5 Network Protocol Services - SABUL

In this section, we follow [22] and describe a network protocol we have developed called the Simple Available Bandwidth Utilization Library or SABUL. SABUL is designed for high performance data transfer and is especially useful on long haul networks.

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 [26] 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.

6 Data Services - DataSpace Transfer Protocol

In this section we follow [19] and describe data services designed to be component services or primitives for distributed data mining applications. The services are part of a protocol called the DataSpace Transfer Protocol, or DSTP, which we have developed. Access to DSTP data can also be gained through proxy servers, such as SOAP based servers. The experiments below used high performance DSTP servers which we have developed [24].

Distributed Columns of Numerical Data. The data model for DSTP Version 2 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 DSTP server, or through a proxy service, such as through a SOAP based server. 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. Version 3 of DSTP may support a more complex data model.

Universal Correlation Keys. A Universal Correlation Key (UCK) is a globally unique id (GUID) and is used for relating columns of data on two different DSTP 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. DSTP applications may or may not use this meta-data. On the other hand, this meta-data is essential for building and deploying statistical models. DSTP servers provide a simple mechanism for associating metadata to columns and collections of columns. DSTP applications often access metadata using SOAP/XML based web services.

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 DSTP Server 1 can be combined with pairs (k_j, y_j) on DSTP Server 2 to produce a table (x_k, y_k) in a DSTP client. The DSTP client can then, for example, find a function $y = f(x)$ relating x and y. This simple mechanism of distributed columns identified by UKCs (perhaps vector valued) is sufficient information for many data mining algorithms.

Depending upon the request, DSTP servers may return one or more columns, one or more rows, or entire tables. DSTP uses XML to describe the metadata.

DSTP applications can also access small data sets using XML so that DSTP is compatible with SOAP. On the other hand, for efficiency and scalability, the default is for data *itself* to be transmitted as records delimited by carriage returns, with fields delimited by commas. The DSTP client may also indicate that a specialized high performance protocol such as SABUL should be used for the data channel. To summarize, the DSTP 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 DSTP 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.

We note that some of the IGrid demonstrations, such as browsing data from the Protein Data Bank, required 30 or more seconds when using SOAP/XML for the metadata while less than a second when using specialized DSTP streaming protocols. This is due to the overhead to parse XML, which becomes more of an application burden as the number of attributes grows.

7 Experimental Studies

7.1 IGrid 2002 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 [27] and the global Terra Wide Data Mining Testbed [28].

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 with StarLight in Chicago as the optical interchange point. These sites include StarLight itself, the Laboratory for Advanced Computing at UIC, iCAIR at Northwestern University, SARA in Amsterdam, and CANARIE in Ottawa. SARA is connected to StarLight via the Netherlands' Surfnet network, Ottawa is connected to Starlight via Canda's CANARIE network, and UIC and iCAIR are connected via OMNInet.

The setup for the experiments in the next two sections was as follows. Three node DSTP clusters were located at the SARA research facility in Amsterdam, at the University of Illinois at Chicago, and at the StarLight Facility in Chicago. StarLight and the University of Illinois at Chicago are located several miles a

part. The SARA cluster and the StarLight cluster were connected via a 10 Gb/s Surfnet link. The UIC cluster and StarLight cluster were connected via a 10 Gb/s OMNInet link.

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.

7.2 PDS Application: Lambda FTP

Our first series of experiments measured an application we developed called Lambda FTP, a high performance implementation of FTP using SABUL. The testing was done between two three-node clusters, one located at StarLight in Chicago and one located at SARA in Amsterdam. The results are reported in the table below. Standard TCP provided about 4.36 Mb/s of bandwidth, while each of the three SABUL streams provided about 900 Mb/s of bandwidth, so that the aggregate SABUL bandwidth was about 2.7 Gb/s between the two three node clusters.

We note that DSTP servers can stream data using SABUL so that these performance numbers are also applicable to streaming DSTP data. In general, though, DSTP clients perform some type of computation so that actual bandwidth is dependent upon the particular data web application. An illustration of this is contained in the next section.

File Transfer between Chicago and Amsterdam (Mb/s)				
TCP Stream	SABUL Stream 1	SABUL Stream 2	SABUL Stream 3	Aggregate SABUL Stream
4.36	902.8	902.9	907.1	2712.8

7.3 PDS Application: Lambda Joins

Our second series of experiments involved testing a basic operation in distributed data mining, merging two data streams by a common key. For this series of experiments, we merged two NCAR data sets [29]. One of the data sets was located at UIC in Chicago and one was located at SARA in Amsterdam. The merging was done at StarLight in Chicago.

The two DSTP streams of data were merged by an algorithm called the Continuously Generated Merge, or CGM, which we developed for merging two high bandwidth data streams [25]. In this case, we merged the two streams using latitude, longitude, and time as the common vector valued key. Once distributed streaming data has been merged in this way, simple statistical counts using a finite buffer can be done in a variety of ways to support the interactive exploration of large remote data sets [30].

Merging distributed data streams by common keys is a basic operation in distributed data mining. Other examples include merging satellite imaging data

of different modalities by latitude/longitude and merging network route dumps by source address.

We now briefly review the CGM algorithm following [25]. 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.

The results in the table below are from the CGM algorithm running using TCP as the network protocol and DSTP 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 randomized by replacing every n'th row (for example, for 10 percent every 10'th 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.

Merging two data streams without PDS			
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 and DSTP as the data service protocol. One DSTP Server per cluster was used to send and receive data, so that the maximum bandwidth between the two clusters was 1 Gb/s. *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 less.

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 containing 1 UCK and 1 attribute; in the second, both data files containing 1 UCK and 7 attributes.

Merging two data streams with PDS - 1 attribute			
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

Merging two data streams with PDS - 7 attributes			
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

8 Summary and Conclusion

In this paper, we have introduced an architecture called Photonic Data Services or PDS which integrates path services, network protocol services, and data services. 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 this approach is practical and useful and that our implementations scale to remote Gigabyte size data sets that are distributed over thousands of miles. 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 across the Atlantic, merge the results using latitude, longitude and time as keys, 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.

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

1. This is the first description describing an *architecture* for integrating path services for photonic networks, network protocols designed for high performance networks, and data services supporting primitives to facilitate remote data analysis and distributed data mining. Integrated services such as 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 integration which allows applications (in our case data services middleware) to signal to optical networks requests to set up, check the status, and tear down photonic paths. The ability to configure paths and related services on an as-needed basis is essential if these services will one day move from today's experimental networks to tomorrow's production networks. Think of this as intelligent application signaling.

3. This is the first experimental study 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.

There are three main areas of work for the future:

1. First, we plan on scaling photonic path services to multiple administrative domains. The experiments reported here used photonic paths services within the OMNInet domain, but paths between OMNInet and Surfnet were set up by hand. This is a Photonic Data Service Layer 2 research problem.

2. Second, we plan on investigating the performance of SABUL on larger clusters. The experiments reported here were limited to three node clusters. This is a Photonic Data Service Layer 4 research problem.

3. Third, we plan on developing additional remote data analysis and distributed data mining algorithms using DSTP primitives. This is a Photonic Data Service Layer 5 research problem.

References

[1] R. L. Grossman, Y. Gu, D. Hanley, X. Hong, J. Levera, M. Mazzucco, D. Lillethun, J. Mambretti, J. Weinberger, Photonic data services: Integrating path, network and data services to support next generation data mining applications, in: Proceedings of the Next Generation Data Mining Conference, Kluwer, 2003.

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

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

[4] J. Walrand, P. Varaiya, High Performance Communication Networks, Morgan Kaufmann, San Francisco, California, 2000.

[5] R. L. Grossman, H. Sivakumar, 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.

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

[7] S. Stolfo, A. L. Prodromidis, P. K. Chan, Jam: Java agents for meta-learning over distributed databases, in: Heckerman et al. [32].

[8] H. Kargupta, I. Hamzaoglu, B. Stafford, Scalable, distributed data mining using an agent based architecture, in: Heckerman et al. [32], pp. 211–214.

[9] J. Darlington, Y. Guo, J. Sutiwaraphun, H. W. To, Parallel induction algorithms for data mining, Lecture Notes in Computer Science 1280.

[10] R. L. Grossman, S. Bailey, A. Ramu, B. Malhi, H. Sivakumar, A. 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.

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

[12] R. W. Moore, C. Baru, R. Marciano, A. Rajasekar, M. Wan, Data-intensive computing, in: The Grid: Blueprint for a New Computing Infrastructure [2], pp. 105–129.

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

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

[15] Patrick, Y. Guo, The design of a platform for distributed kdd components, in: Parthasarathy et al. [31], pp. 63–78.

[16] W. Du, G. Agrawal, Using general grid tools and compiler technology for distributed data mining: Preliminary report, in: Parthasarathy et al. [31], pp. 51–61.

[17] R. Grossman, M. Hornick, G. Meyer, Data mining standards initiatives, Communications of the ACM 45 (8) (2002) 59–61.

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

[19] R. Grossman, M. Mazzucco, Dataspace - a web infrastructure for the exploratory analysis and mining of data, IEEE Computing in Science and Engineering .

[20] M. Veeraraghavan, R. Karri, T. Moors, M. Karol, R. Grobler, Architectures and protocols that enable new applications on optical networks, IEEE Communications Magazine (2001) 118–127.

[21] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, Journal of Network and Computer Applications 23 (2001) 187–200.

[22] R. L. Grossman, M. Mazzucco, H. Sivakumar, Y. Pan, Simple available bandwidth utilization library for high-speed wide area networks, submitted for publication .

[23] D. Lillethun, J. Mambretti, J. Weinberger, Odin: Path services for optical networks, in preparation, www.icair.org .

[24] S. Bailey, E. Creel, R. L. Grossman, S. Gutti, H. Sivakumar, A high performance implementation of the data space transfer protocol (dstp), in: Large-Scale Parallel Data Mining, Springer-Verlag, 2000, pp. 55—64.

[25] M. Mazzucco, A. Ananthanarayan, R. L. Grossman, J. Levera, G. B. Rao, Merging multiple data streams on common keys over high performance networks, in: Proceedings of Supercomputing 2002, IEEE and ACM, 2002.

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

[27] J. Mambretti, Omninet, www.icair.org/omninet .

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

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

[30] R. L. Grossman, J. Levera, M. Mazzucco, Aggregate queries on streams of data using a small buffer, UIC Laboratory for Advanced Computing Technical Report (2002).

[31] S. Parthasarathy, H. Kargupta, V. Kumar, D. Skillicorn, M. Zaki (Eds.), High Performance Data Mining, SIAM, Philadelphia, Pennsylvania, 2002.

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