Open DMIX: High Performance Web Services for Distributed Data Mining

Robert Grossman[*], Yunhong Gu, Chetan Gupta,
David Hanley, Xinwei Hong, and Parthasarathy Krishnaswamy

January 15, 2004

This is a draft of the paper Robert L. Grossman, Yunhong Gu, Chetan Gupta, David Hanley, Xinwei Hong, and Parthasarathy Krishnaswamy, Web-Service Based Data Mining Middleware for Grid Computing Environments, 7th International Workshop on High Performance and Distributed Mining, in association with the Fourth International SIAM Conference on Data Mining, to appear.

Abstract

In this note, we introduce Open DMIX, an open source collection of web services for the mining, integration, and exploration of remote and distributed data. We also describe some preliminary experimental results using Open DMIX. Open DMIX is layered: the top layer provides templated data mining and statistical algorithms, such as those defined by the Predictive Model Markup Language [8]. The middle layer provides access and integration of remote and distributed data using the DataSpace Transfer Protocol (DSTP) [6]. The bottom layer provides specialized network protocols designed to work with large distributed data sets over wide area networks, which may have high bandwidth delay products (BDPs). Open DMIX clients interact with Open DMIX servers using a version of web services designed for high performance applications, which we call SOAP+.

[*]Robert Grossman is also with Open Data Partners.

1 Introduction

Open DMIX is an open source collection of web services for the mining, integration and exploration of remote and distributed data. Open DMIX is layered: the top layer provides templated data mining and statistical algorithms, such as those defined by the Predictive Model Markup Language [8]. The middle layer provides access and integration of remote and distributed data using the DataSpace Transfer Protocol (DSTP) [6]. The bottom layer provides specialized network protocols designed to work with large distributed data sets over wide area networks, which may have high bandwidth delay products (BDPs). Open DMIX clients interact with Open DMIX servers using a version of web services designed for high performance applications, which we call SOAP+.

We believe that Open DMIX is novel for the following reasons:

- Open DMIX enables the high performance exploration, integration, and mining of remote and distributed data using a data web paradigm instead of a data grid paradigm [6].
- Open DMIX integrates specialized network protocols with data access services so that very large remote and distributed data sets can be accessed and analyzed. Traditionally, data mining using traditional TCP-based services does not perform well on networks with high bandwidth delay products. Recall that the bandwidth delay product is the product of the bandwidth and the round trip time (RTT) of a packet.

- Open DMIX uses a scalable version of web services, which we describe in more detail below. Traditional web services do not provide adequate performance for many common data mining queries, as is also discussed below.

Section 2 describes related work. Section 3 describes the design of Open DMIX. Section 4 describes Open DMIX network protocols. Section 5 describes Open DMIX web services. Section 6 describes some Open DMIX data mining applications. Section 7 describes experimental studies using Open DMIX. Section 8 is the conclusion.

2 Background and Related Work

Recently, several different infrastructures have been used for distributed data mining:

- *Data Grids.* Data grid services combine authentication, authorization, and access (AAA) controls with resource managers so that arbitrary computations can be done using distributed computational and data resources belonging to a virtual organization [3]. Good examples of data grids are the data grids developed by physicists [13] and astronomers [4] to process the data collected by the collaboration's instruments. Data in data grids is stored in files and transported using GridFTP [1].

- *SOAP/XML-based Web Services.* Web services based upon SOAP and XML are a rapidly maturing infrastructure for accessing XML-based data [14]. SOAP enables the serialization of XML-data so that it may be transported using TCP or HTTP. SOAP-based services can be described using the Web Services Description Language, or WSDL, while Universal Description, Discovery, and Integration (UDDI) provides a simple mechanism for the discovery of web services. SOAP/XML-based web services are designed to deal with general XML-based data.

- *Remote Access to Databases.* SQL is the standard query language for databases and ODBC and JDBC are widely deployed protocols for accessing remote data resident in databases. In contrast to data grids, the ODBC and JDBC protocols support the full power of SQL and enable attribute level access to records, as well as a rich range of different types of SQL-based selections.

- *Hybrid Approaches.* There are also efforts to combine these approaches. The Open Grid Services Architecture is a standards effort that integrates grids with web services [12]. The OGSA Database Access and Integration Services (OGSA DAIS) [11] combines grid services with web services for remotely accessing databases.

Data grids are file based infrastructures that support general computation; in particular, they do not provide attribute based access to remote data, nor do they provide specific functionality directly related to data and its attributes. Databases were not designed to scale in the fashion of data grids or to work with remote and distributed data. Finally, web services today neither have the scalability of data grid applications nor do they provide specific data related functionality, although they do provide (using XPath for example) access to attributes.

The Open DMIX Services we introduce have i) the ability to work with attributes like databases and SOAP/XML web services, but lacking in data grids; ii) the scalability of data grids, which is lacking in remotely accessed databases and web services; iii) direct support for remote and distributed resources, which is provided by data grids and web services; and iv) explicit support for the common operations required to analyze and mine remote and distributed data, which is missing in all three approaches.

3 Design of Open DMIX

As described above, Open DMIX consists of three layers. A top layer containing templated data mining operations, a middle layer containing data access and integration services, and a bottom layer containing specialized network protocols.

The data access and integration services are based upon the DataSpace Transfer Protocol or DSTP [6]. DSTP is a protocol designed for accessing, exploring, and integrating remote and distributed data. DSTP has specific support for keys, metadata, and data. DSTP also supports missing values and provides various mechanisms for selecting data.

Security and policy are not part of the architecture per se, but are added as required for each application. The Open DMIX data services are based upon web services, and any web service compliant security mechanism, such as TLS, can be used.

DSTP has always provided explicit support for distributed joins [6]. Briefly, any attribute can be associated with a key called a universal correlation key (UCKs), which, in turn, is associated with a globally unique ID (GUID). Any two (distributed) attributes with the same GUID can be joined. DSTP servers provide explicit support for UCKs.

Metadata is stored in the server along with the data itself and may be searched. The data itself may be physically stored using one of several backends. Data may be stored in delimited ASCII files, in flat binary files, in a SQL database, in netCDF files, or in HDF files. Regardless of the backend, the data will be accessed in an SQL-like syntax. This syntax is a subset of SQL, and allows selection of data elements based on data and row range criteria, as well as statistical decimation of data.

This is achieved by recording what format the file is stored in (in XML meta-data), and by having a generalized parser that delivers an SQL-like query to plug-in modules that access that specific file format. The parsing of the query is handled by a higher level of the server than the data access module, which converts the query to a simple format.

For example, once we know there is a dataset "temperatures" on a server with the attributes "temperature," "humidity," and "time," we may write the following C++ code:

```
Client c;
c.setRange(0,100000);
c.setDecimation(50);
Vector<char*> v =
   c.soapQuery( "select
      temperature, time
      from temperatures
      where humidity>50" );
```

This will limit the query to the first 100,000 rows, and return 50% of the records that match our query. Note that the return mechanism in this example is via SOAP. Since SOAP must parse every data item to/from XML, its performance necessarily suffers. Additionally, many implementations require that the entire return set be resident in the memory of the client machine on completion of the call. This obviously limits both potential performance and the size of the return set.

Open DMIX also support a high performance mode which we call SOAP+. SOAP+ employs separate control and data channels. The data channel is not required to use XML and TCP. Also multiple threads can be used to stripe data.

# records	SOAP/XML Mode (sec)	SOAP+(sec)
10,000	0.65	0.21
50,000	2.57	0.72
150,000	11.13	2.05
375,000	51.18	5.01
1,000,000	352.10	13.43

Table 1: Open DMIX servers have two modes for data access. One mode uses SOAP/XML, which works for small data sets, and (small) metadata. The other mode uses uses a separate data and control channel. The data channel is not required to use TCP and XML.

The table above contains some performance measurements comparing XML/SOAP and SOAP+. The data used for this is a simple ten column dataset of random integers.

As the table makes clear, the SOAP/XML mode does not scale linearly with query size, and, in fact, breaks with sufficiently large queries. The 1 million row SOAP query consumed 99% of the CPU and much of the RAM on the server, then on the client, in its marshalling and demarshalling.

4 Open DMIX Network Protocol Services

There are several problems that are the result of TCP's window-based congestion control algorithm when it is used on networks with high bandwidth delay products (BDP). The bandwidth delay product is defined to be the product of the bandwidth and the round trip time (RTT) of a TCP packet. First, TCP's congestion control algorithm is not fair to flows with different RTTs. Flows with high BDPs will generally get less of the available bandwidth than flows with lower BDPs when passing through a common bottleneck. In particular, the theoretical upper limit of throughput decreases as the BDP of the link increases. Second, the AIMD (Additional Increase Multiplicative Decrease) algorithm used by the congestion control algorithm to set the sending rate can take a very long time to discover the available bandwidth on high BDP networks. Third, dropped packets due to physical errors on the links (not due to congestion) can prevent TCP from obtaining a high throughput.

We performed a simple series of tests to determine the performance of TCP over a 1 Gb/s link between Chicago and Amsterdam. The throughput of a single TCP stream without tuning the buffer size is about 4.5 Mb/s. The throughput of a single TCP stream is about 30 Mb/s when the TCP buffer is set to 12 MB, the approximate BDP value of the link. With parallel TCP streams, a maximum throughout of 320 Mb/s was obtained using 64 TCP flows, each with a 2 MB buffer. In another series of experiments, we set up two separate flows terminating in a common node in Chicago, one from a local node in Chicago and one from a remote node in Amsterdam. The former obtained 890 Mb/s and the latter 3.5 Mb/s.

Since many Open DMIX queries require multiple flows, it is very important that Open DMIX use a network protocol that is fair to each of the data flows, so that each flow obtains approximately equal bandwidth.

These results show that TCP tuning or parallel TCP is not sufficient for data intensive grid applications. Several new transport protocols have been introduced to address these issues, but few of them are widely used in grid computing; there are several reasons for this. First, many of the new protocols require that the existing network infrastructure be changed, for example, by requiring a specially tuned operating system kernel or by requiring modification to the routers. There is a large cost for these types of changes, and problems may arise in collaborative networks. Second, some of the new protocols are fast, but not fair. This is adequate when a single flow is used for bulk data transport on a network without congestion, but not adequate for data intensive computing on grids, when multiple flows are needed.

In Open DMIX, we integrate a new network protocol, called SABUL/UDT, that we have described previously [5, 7]. SABUL is a uni-directioal application level data transfer protocol. Data is transferred from one side (the sender) to the other side (the receiver). It uses UDP to transfer data with rate control and TCP to feed back control information for reliability and rate control. UDT is a variant of SABUL that uses UDP for the control channel as well. SABUL/UDT also uses a constant rate control interval and selective acknowledgments. Other protocols similar to SABUL have been proposed, including Tsunami [15], FOBS [2], and RBUDP [10].

It is well understood that TCP does not work well on high bandwidth, long delay networks. SABUL/UDT uses rate controlled UDP to solve the problem. The rate control that tunes the interpacket time produces a smoother data flow than TCP's window based control. Instead of RTT (round trip time), SABUL/UDT uses a fixed rate control interval (0.01 second) to remove the fairness bias caused by network delay. The control algorithm combines AIMD (additive increase multiplicative decrease) and MIMD (multiplicative increase multiplicative decrease) to satisfy fairness (including TCP fairness) while keeping high performance.

The higher the current sending rate, the higher SABUL/UDT increases it. This aggressively uses available bandwidth. In addition, SABUL/UDT does not decrease its sending rate for every single loss report, and when it does decrease it, it does so by only 1/8.

Since SABUL/UDT uses a constant rate control interval, TCP will increase quickly on short

RTT links (RTT < 0.01 sec). In addition, SABUL/UDT increases more slowly for lower bandwidth links. In this sense, it is friendly in traditional low bandwidth delay product networks. In high bandwidth delay product networks, where TCP does not perform well, SABUL/UDT can utilize the bandwidth more efficiently than TCP.

5 Open DMIX Data Access and Integration Services

A Web service listens on a port, accepting simple sockets, or it may reside as a service (for example, a CGI) in a web server, relying on the web server for data handling. When running under a web server, a SOAP server can take advantage of firewall tunneling, although performance will be reduced. The service accepts XML that describes an action for the server to perform, and returns XML to the client describing the result of the operation. It is possible to maintain state between operations, and operations are essentially non-streaming, due to the encoding rules of XML.

The primary problem with web services in the context of high performance data mining is that, due to the overhead of XML encoding and parsing, there is a limit to the speed of the data transmission and the total size of the return set. This is caused by the need to retain the entire dataset in local storage due to XML encoding and decoding rules. The specific issue is that redundant parts of XML documents can and should refer to the other similar parts of documents. This requires that the entire document be maintained for lookup purposes. Therefore, all data transport mechanisms that are truly XML compliant are in essence non-streaming. While a server could, in theory, safely ignore this rule in transmission, when there are no circular data structures, a compliant client cannot safely do so.

Web services are also limited by the performance of TCP sockets, as was discussed in the section above.

Our approach to high-performance web services is to create a fully compliant SOAP implementation, yet let the client and server negotiate a high-performance data channel. Results may be returned via the standard mechanism, or they may be returned via high performance protocols requested in the call and described in the return. When using normal SOAP return mechanisms, the size of the return set is limited so as to not cause an excessive storage/CPU burden on any one system.

MB	SOAP (sec)	SOAP+ (sec)
1.08	39.78	00.87
5.03	178.65	04.72
18.55	807.36	07.56
50.62	1776.99	25.16

Table 2: Comparing standard SOAP, with a SOAP+, a SOAP implementation we developed for Open DMIX, which uses a SOAP/XML control channel and a streaming data channel.

The simplest high performance mechanism is to return text rows as a stream in a normal TCP socket. This approach is simple and allows reasonable speed over local distances. The problems with this approach are the overhead of parsing and encoding text data and the use of TCP, which does not scale to long distances. This is a streaming approach, and only needs to consume a fixed amount of storage on the client and server, rather than storage proportional to the size of the dataset.

Data may also be returned as binary records; in this case, the return of the SOAP call describes the encoding of the binary record. These records are generally significantly more compact than text, and much faster to parse. In fact, as the client may request a specific endianess, no parsing may be needed. This approach has speed advantages even when the source is non-binary, although the performance is far greater with binary data sources.

We conducted a test that queried data from the Sloan Digital Sky Survey via both SOAP, and, as mentioned above, what we call SOAP+, which uses a SOAP/XML control channel and a streaming data channel.

6 Open DMIX Data Mining Applications

Currently, Open DMIX is integrated with the open source R statistical system. We have also begun to integrate some specialized streaming algorithms into Open DMIX. For example, we integrated a

	Query 1	Query 2	Query 3
text	3.63	1.49	5.42
binary	6.61	0.27	0.03
sql	9.11	1.08	1.18

Table 3: Open DMIX performance data for a data set containing 3 columns and one million rows. Time in seconds.

streaming clustering algorithm called GenIC [9] into the top layer of Open DMIX.

As a simple example, we set up two data sets of network intrusion data in Amsterdam. We then transported, integrated, and clustered the data using GenIC in Chicago over a network with a 110 ms RTT. Using standard web services the test took about 2400 seconds. The same test using Open DMIX took only 42 seconds. The next section describes some additional experimental studies using GenIC.

We are currently beginning to develop data mining primitives to simplify the integration of additional data mining algorithms into Open DMIX.

7 Experimental Results

In this section, we describe some experimental results using the Open DMIX server. Unless otherwise mentioned, all tests took place between a cluster in Amsterdam and a cluster in Chicago connected with a 1 Gb/s network having a 110 ms RTT. We used two data sets: a synthetic atmospheric data set and a network intrusion data set consisted of Snort alerts we collected over several months.

First, we examined three queries on a million row data set. The first query returns all the rows and columns. The second query returns rows 999000-999500 using a simple range selection. For our data set containing one million rows, it returns 0.5% of the data. The third query selects one column to return when one of the other columns is less than 650. For our dataset this results in returning less than 0.5% of the data.

Although there are obviously significant gains when streaming protocols are used for data transport, the XML/SOAP mechanism is still used for

	Query 1	Query 2	Query 3
text	32.88	1.47	56.3
binary	62.89	0.45	0.28
sql	85.05	1.08	11.76

Table 4: Open DMIX performance data for a data set containing 3 columns and 10 million rows. Time in seconds.

	Query 1	Query 2	Query 3
text	5.87	3.66	11.5
binary	14.24	0.32	0.41
sql	13.78	1.64	1.75

Table 5: Open DMIX performance data for a data set containing 10 columns and 1 million rows. Time in seconds.

	Query 1	Query 2	Query 3
text	17.61	16.21	46.00
binary	53.01	1.23	0.07
sql	41.9	4.72	4.79

Table 6: Open DMIX performance data for a data set containing 50 columns and 1 million rows. Time in seconds.

	Query 1	Query 2	Query 3
text	41.1	32.09	90.77
binary	105.67	2.23	0.12
sql	94.84	9.06	8.549

Table 7: Open DMIX performance data for a data set containing 100 columns and 1 million rows. Time in seconds.

Protocol	Test 1	Test 2	Test 3
UDT (secs)	81.85	80.77	81.39
TCP (secs)	634.40	635.11	634.73

Table 8: Comparing Open DMIX using UDT and TCP for a query which returns all the data. Time in seconds.

Trial #	100MB	1GB
1	248	2452
2	250	2437
3	249	2448
4	259	2457
5	262	2437

Table 9: Five trials using Open DMIX to retrieve and integrate data using TCP sockets. Time in seconds.

the transport of meta data and for small queries to enhance interoperability and the potential user base.

For the next series of tests, we used Open DMIX servers on a cluster in Amsterdam, which we accessed using a single client in Chicago. The goal was to compare retrieving data using Open DMIX with UDT and using Open DMIX with TCP. The data set contained 10 million rows of data and comprised 320 MBs. This test retrieved all the data using a select * query.

We then performed tests to ascertain the stability and scalability of DMIX. These tests involved running several trials of DMIX to show conisistent and linear performnce over a number of trials. We performed tests with both TCP and UDT used as transports and achieved similar results with each protocol.

We also performed tests that varied several internal parameters in the GenIC algorithm. In particular, we varied the number of cluster centers.

We also performed a three-hour test that clustered over 20GB of data. In this test we examined the changes in the sizes of the clusters over time; the resultant cluster sizes are shown as a scatter plot in Figure 1.

Trial #	100MB	1GB	10GB
1	4.31	42.8	421
2	4.88	43.0	418
3	4.49	42.9	425
4	4.12	43.1	419
5	4.34	42.7	416

Table 10: Five trials using Open DMIX to retrieve and integrate data using UDT sockets. Time in seconds.

Trial #	100MB	1GB
1	322	3213
2	317	3227
3	323	3205
4	299	3222
5	303	3230

Table 11: Five trials using Open DMIX to retrieve, integrate and cluster network intrusion data using TCP sockets. Time in seconds.

Trial #	100MB	1GB
1	123	1230
2	123	1216
3	124	1213
4	125	1221
5	122	1223

Table 12: Five trials using Open DMIX to retrieve, integrate and cluster network intrusion data using UDT. Time in seconds.

# Centers	Average time in seconds
4	122
8	107
12	115
16	238

Table 13: The results of varying the number of cluster centers when using Open DMIX to retrieve, integrate and clusters four streams of network intrusion data. About 100 MB of data was used. The median times of three runs is reported in seconds.

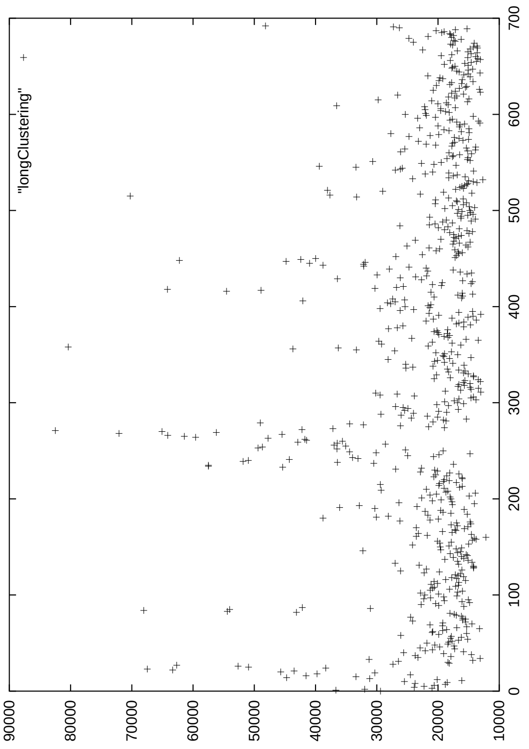

Figure 1: The sum of squares of sizes of clusters for a stream of about 15 GB of network intrusion data. Peaks in the graph correlate to network incidents such as worms. The X axis is the cluster sizes, the X axis is the cluster report index.

8 Conclusion

In this note, we have provided a preliminary description of Open DMIX. Open DMIX is a layered system for exploring, integrating and mining remote and distributed data. The top layer supports templated data mining operations such as clustering. The middle layer provides data access and integration services based upon the DataSpace Transport Protocol (DSTP). The bottom layer provides alternate network protocols to support high performance queries on networks with high bandwidth delay products.

Open DMIX uses an alternative to SOAP we call SOAP+, which uses a SOAP/XML based control channel and a separate data channel, which can employ alternate network protocols such as SABUL/UDT [5].

In our preliminary experiments, Open DMIX has proven to be useful, easy to use, and scalable. We are currently preparing our first release of Open DMIX and continuing our experiments.

References

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

[2] Fobs. omega.cs.iit.edu/ ondrej/research/fobs, retrieved on April 16, 2003.

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

[4] Jim Gray and Alexander S. Szalay. The world-wide telescope. *Science*, 293:2037–2040, 2001.

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

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

[7] Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong, Dave Lillethun, Jorge Levera, Joe Mambretti, Marco Mazzucco, and Jeremy Weinberger. Experimental studes using photonic data services at igrid 2002. *Journal of Future Generation Computer Systems*, 19(6):945–955, 2003.

[8] Data Mining Group. Predictive model markup language (pmml). http://www.dmg.org, January 10 2003.

[9] Chetan Gupta and Robert L. Grossman. Genic: A single pass generalized incremental algorithm for clustering. SIAM, 2004.

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

[11] Norman W. Paton, Malcolm P Atkinson, Vijay Dialani, Dave Pearson, Tony Storey, and Paul Watson. Database access and integration services on the grid. 2002.

[12] The Globus Project. Towards globus toolkit 3.0: Open grid services architecture. http://www.globus.org/ogsa/, retrieved on January 10, 2003.

[13] The GriPhyN Project. Griphyn - grid physics network. http://www.griphyn.org, retrieved on April 1, 2003.

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

[15] Tsunami. www.anml.iu.edu/anmlresearch.html, retrieved on April 4, 2003.