

Experimental Studies Scaling Web Services For Data Mining Using Open DMIX: Preliminary Results

Robert Grossman and David Hanley

Abstract

We have developed Open DMIX, which is an open source collection of web services for accessing, exploring, integrating, and mining remote and distributed data. Open DMIX clients interact with Open DMIX servers using a version of web services designed for high performance applications, which we call SOAP+. In this paper we describe experimental studies comparing SOAP and SOAP+ based data mining applications on high performance, wide area networks. For these applications, Open DMIX using SOAP+ can be significantly faster than traditional SOAP-based applications.

1. Introduction

Today, most data mining takes place in one of two ways. In the first way, a client server or 3-tier data mining application accesses and analyzes local data. In the second way, data mining is embedded in another application, either explicitly or implicitly. For example, today data mining is embedded into the databases marketed and sold by IBM, Microsoft and Oracle. Data mining is also commonly embedded into a variety of applications, for example in CRM and financial risk applications.

During the past several years, web services have matured to the point where it is now becoming practical to create distributed data mining infrastructures and platforms based upon them. Indeed, web services have the potential to change the infrastructure used to explore, analyze and mine data in a fundamental way. Consider the following: today, many people find it quicker to locate a preprint by using Google than to search for it on their own local disk. On the other hand, almost all data analysis is done using local data. As bandwidth becomes a commodity resource [5], accessing remote data and remote services will become easier, and one day it may be as easy to work with remote data as it is to work with local data.

However, even basic experiments using web services for data mining quickly identify some fundamental limitations. As we show in more detail below, mining remote and distributed data using SOAP/XML-based web services [22] does not scale to even moderate size data sets.

In this paper, we describe some experimental studies using Open DMIX, which is a collection of scalable web services for accessing, exploring, integrating and mining data. Open DMIX differs from prior work of which we are aware in three key ways.

- Open DMIX can use traditional SOAP/XML/TCP-based web services for small datasets and metadata. In addition, Open DMIX supports a new protocol called SOAP+ for larger datasets and metadata collections.

- SOAP+ has two channels. The first is a SOAP/XML/TCP-based control channel and the second is a data channel. The data channel can employ specialized network protocols and packaging formats.
- The data channel in SOAP+ can use a specialized network protocol we have developed for working with large remote and distributed data sets called UDT (UDP-based Data-Transfer Protocol).

In this paper we present some experimental studies comparing SOAP vs. SOAP+.

2. Background and Related Work

This section is based in part on [10]. It is convenient to think of data mining systems that were developed during the past decade as comprising three generations: 1) client-server systems; 2) component and agent-based systems; and 3) systems based upon web services.

The first generation of data mining system utilized local data, with either client-server or 3-tier architectures. With these systems, a client front end is used to access a server (possibly on the same machine) hosting the data mining application. With a client-server model, the server also manages the data; with a 3-tier model, the data is accessed from another source using ODBC, JDBC, or other related protocol.

The next generation of data mining systems was component-based. The components could be local, relying on Microsoft's COM or DCOM platforms, for example, or global, relying on systems such as Sun's J2EE platform. Angoss is an example of the former, and Kensington is an example of the latter.

More or less at the same time, various experimental agent-based data mining systems were developed. The basic assumption in these systems is that the data is distributed and agents are used to move the data, move the models produced by a local data mining system, or move the results of a local data mining computation. Today, very few agent-based systems are used in practice. This is probably because no agent-based infrastructure, over which an agent-based data mining system must be built, was ever widely adopted. Examples of agent-based distributed data mining systems include JAM [19], Papyrus [7], and BODHI [14].

Somewhat later, the next generation of service-based data mining systems began to emerge. These are generally built using W3C's standardization of web services. Examples include DataSpace [9] and data mining systems developed by IBM, Microsoft and SAS that employ the XML for Analysis standard [3].

More general service-based infrastructures, such as grids or data grids [4], are also used for data mining, especially when large computational resources are required. A data grid uses Globus, or an equivalent infrastructure, to provide a security infrastructure and resource management infrastructure so that distributed computing resources can be used. In addition, Globus provides a high performance data transport mechanism called GridFTP. Recently, the Grid community has begun an effort called the Open Grid Service Architecture, or OGSA, that provides a web service-based access to some grid services [17]. OGSA Database Access and Integration Services (OGSA DAIS) [16] combine grid services with web services for remotely accessing databases.

3. SOAP Based Web Services

A web service may be implemented as a standalone TCP server, or it may be accessed via a URL through a web server. When running under a web server, the SOAP service can take advantage of firewall tunneling, although performance will be reduced. The SOAP 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 marshaling rules of XML.

There are two fundamental problems when using web services for data mining of moderate to large size remote and distributed data sets.

First, 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 must refer to the other similar parts of documents. This requires that the entire document be maintained for lookup purposes. Therefore, all data packaging mechanisms that are truly XML compliant are in essence non-streaming. While a server could, in theory, safely ignore this encoding rule when there are no circular data structures, a compliant client cannot safely do so.

Second, web services are also limited by the performance of TCP sockets. As a simple example, when transporting data from our cluster in Amsterdam to our cluster in Chicago for the experiments reported below, TCP flows averaged between 3-4 Mb/s over a 1 Gb/s link. This is primarily due to the limitations of TCP when used on networks with high bandwidth delay products. Here the bandwidth delay product is 1 Gb/s x 110 ms or 13.75 MB. In contrast, UDT, the new network protocol we developed for Open DMIX, provides significantly higher performance. A single UDT flow can average 950 Mb/s over the same link.

4. SOAP+ Based Web Services

Open DMIX employs both standard SOAP/XML-based web services, as well as a high performance version, which we call SOAP+. SOAP+ uses separate data and control channels. The data channel can employ high-speed network protocols and alternatives to XML.

In a typical application, metadata and small data sets can be accessed using SOAP, while larger data sets can be accessed using SOAP+. The table below contains some performance measurements comparing SOAP and SOAP+ when accessing a synthetic data set containing 10 attributes and the indicated number of data records.

Record Count	SOAP using TCP/XML (secs)	SOAP+ using UDT/ASCII (secs)
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	352.1	13.43

Table 1. All times are in seconds. The tests were performed on a 1 Gb/s network linking a cluster in Chicago with a cluster in Amsterdam. The roundtrip time was 110 ms. This table shows the results of using SOAP with SOAP+, which employed UDT, and simple delimited ASCII text records.

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, and then on the client, in its marshalling and de-marshalling.

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 in order not to cause an excessive storage/CPU burden on the client or server.

The simplest high performance mechanism is to return ASCII text records 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 ASCII 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 can request a specific endianness, no client-side 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.

5. Alternate Network Transport Protocols

In prior work, we introduced an alternative to TCP for data mining called SABUL [8] and demonstrated that SABUL is significantly faster than TCP for mining remote and distributed data over networks with high bandwidth delay products [11]. Recall that the bandwidth delay product is the product of the bandwidth and the round trip time of the path. TCP throughput is directly limited by the bandwidth delay product of the connection it is using.

In further experiments, we found that, although SABUL works well when mining a single high volume flow of data, there were difficulties when transporting multiple high volume flows, as would be required, for example, when integrating two high volume flows from two geographically distributed data sets prior to applying a streaming data mining operation, such as streaming clustering.

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. Since Open DMIX queries use traditional SOAP/XML/TCP web services, it is also important that Open DMIX use a network protocol that is friendly to multiple TCP flows.

Recently, we have developed a new application level protocol called UDT and integrated it into Open DMIX. Since UDT is an application layer protocol, it is straightforward to use it for an application layer service, such as data mining. UDT is built on top of UDP and provides reliability (which UDP lacks) and congestion control to support the Open DMIX requirements of fairness (in order to support data mining operations on multiple UDT flows) and friendliness (in order to support TCP based control information). Other protocols similar to UDT have been proposed, including Tsunami [21], FOBS [2], and RBUDP [15].

For our experiments, we measured the performance of UDT over a 1 Gb/s network connecting a cluster in Chicago with a cluster in Amsterdam. The round trip time was 110 ms. The results are summarized in Table 1. TCP flows averaged 3-4 Mb/s. A single UDT flow averaged 950 Mb/s. Note that UDT is fair to multiple UDT flows. For example, four UDT flows share the bandwidth and average 169 Mb/s. This is important to support streaming data mining operations on multiple high volume data flows. Notice also that UDT is friendly to TCP in the sense that it essentially does not affect the performance of TCP flows--in general they average about 4.2 Mb/s until the link becomes congested. This is important since SOAP+ uses TCP for the control channel.

UDT			TCP			Overall Throughput
# Flows	Average	Aggregate	# Flows	Average	Aggregate	
1	667	667	50	4.23	212	878
2	315.5	631	50	4.19	210	841
3	222	666	50	4.05	203	869
4	169.3	677	50	3.81	191	868

Table 2. All measurements are in seconds. This table compares two network protocols UDT, which is used in SOAP+ for the data channel, and TCP, which is used with SOAP. SOAP+ uses TCP for its control channel. The tests were run between Chicago and Amsterdam on a 1 Gb/s network. The roundtrip time was 110 ms. This table illustrates the performance advantage of using UDT instead of TCP for mining high volume data flows. Notice that a single UDT flow is about 150x faster than a single TCP flow. Notice also that UDT is fair to multiple UDT flows and friendly to multiple TCP flows.

6. Alternative Packaging Formats

In addition to returning data in delimited ASCII fields (instead of XML), SOAP+ may also use a binary format, which is faster for several reasons. Marshalling overhead will be far lower compared to ASCII records, and may even be nonexistent. The records are more compact, and of a predictable size. Decoding will also be far more efficient. Table 3 compares text and binary records as packaging formats. Table 4 provides another comparison between using XML, delimited ASCII and binary formats.

It is worth observing that binary records have a 2:1 advantage with regards to transmission size, but have a speed advantage even larger, due to their more efficient encoding, which plays a role

even when no work is being done on the records, because there is no need to scan for end of string markers.

Record Count	SOAP time (seconds)	SOAP+ time using UDT and binary packaging
10000	0.07	0.07
50000	0.57	0.11
150000	8.24	0.15
375000	50.73	0.37
1000000	351.23	0.99
5000000	7400.47	4.66

Table 3. All times are in seconds. The tests were performed on a 1 Gb/s network linking a cluster in Chicago with a cluster in Amsterdam using data records with five attributes. The roundtrip time was 110 ms. This table compares SOAP and SOAP+. SOAP+ uses UDT as the network protocol and binary as the packaging formats.

Rows	ASCII time (sec)	ASCII size (MB)	Binary time (sec)	Binary size (MB)
1000000	1.57	48	0.52	28
2000000	2.99	96	0.97	56
5000000	7.31	240	2.32	140
10000000	14.56	480	4.61	280

Table 4. Note the significant advantages of binary compared to ASCII packaging formats. The tests were performed on a 1 Gb/s network linking a cluster in Chicago with a cluster in Amsterdam using data records with five attributes.

7. Clustering as a Web Service

In a final series of experiments, we examined the ability of Open DMIX to integrate two streams of data and apply a streaming clustering algorithm. The streaming clustering algorithm we used is described in [13].

Table 5 shows the time required to transport, integrate, and cluster 1.25 GBs of data using Open DMIX with TCP as the transport protocol, then using UDT as the transport protocol. The data was network intrusion data with 11 attributes. The clustering process using UDT was CPU bound or the differences using these two protocols would be even greater. The experiments were repeated five times. The results are in seconds. The remote data resided on a server in Amsterdam and was accessed over a 1 Gb/s route with a 110 ms RTT.

	SOAP	SOAP+
Experiment 1	1346	280
Experiment 2	1417	276
Experiment 3	1321	287
Experiment 4	1425	285
Experiment 5	1278	286
Average	1357	283

Table 5. All times are in seconds. Five experiments were run on a 1 Gb/s network between Chicago and Amsterdam with a 110 ms round trip time. Two 1.25 GB datasets were transported, integrated and clustered using a streaming algorithm using both SOAP and SOAP+. The process was CPU bound or the time using SOAP and SOAP+ would be even greater.

8. Conclusion

Web services are emerging as a standard mechanism for developing remote and distributed data mining applications. For commodity long haul networks, working with large data sets in this way can be very slow. On the other hand, the number of high performance networks is increasing, and wide area networks with 1 Gb/s and 10 Gb/s bandwidth are becoming available [23]. With these types of networks, it is practical to work with large (1 GB and larger) remote and distributed data sets if the appropriate network protocols and packaging formats are used.

We have developed an open source data mining, exploration and integration system called Open DMIX based upon high performance web services called SOAP+. SOAP+ uses a traditional SOAP-based control channel and a separate data channel, which can employ high performance network protocols and alternate packaging formats. In this paper, we report on experimental studies comparing SOAP and SOAP+ over wide area high performance networks. SOAP+ can provide significant performance advantages, sometimes as much as 10x-100x.

9. References

- [1] Mario Cannataro, Domenico Talia, and Paolo Trunfio. The Knowledge Grid: Towards An Architecture For Knowledge Discovery On The Grid. to appear.
- [2] Fobs. omega.cs.iit.edu/~ondrej/research/fobs, retrieved on April 16, 2003.
- [3] XML for Analysis Consortium. Xml For Analysis. retrieved from <http://www.xmla.org>, October 10, 2003.
- [4] I. Foster and C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco, California, 1999.
- [5] Ian Foster and Robert L. Grossman. Data Integration In A Bandwidth Rich World. *Communications of the ACM*, 46(11):50–57, 2003.
- [6] R. L. Grossman. Standards And Infrastructures For Data Mining. *Communications of the ACM*, 45(8):45–48, 2002.

- [7] R. L. Grossman, S. Bailey, A. Ramu, B. Malhi, H. Sivakumar, and A. Turinsky. Papyrus: A System For Data Mining Over Local And Wide Area Clusters And Super-Clusters. In Proceedings of Supercomputing. IEEE, 1999.
- [8] 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.
- [9] 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.
- [10] Robert L. Grossman. Standards, Services And Platforms For Data Mining: A Quick Overview. In Proceedings of the 2003 KDD Workshop on Data Mining Standards, Services and Platforms (DM-SSP 03), to appear.
- [11] 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 Generation Computer Systems, 19(6):945–955, 2003.
- [12] Data Mining Group. Predictive Model Markup Language (PMML). <http://www.dmg.org>, January 10 2003.
- [13] Chetan Gupta and Robert L. Grossman. Genic: A Single Pass Generalized Incremental Algorithm For Clustering. SIAM, 2004.
- [14] I. Hamzaoglu H. Kargupta and B. Stafford. Scalable, Distributed Data Mining Using An Agent Based Architecture. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, Proceedings of KDD '97, The Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, USA, pages 211–214. AAAI Press, August 1997.
- [15] E. He, J. Leigh, O. Yu, and T. DeFanti. Reliable Blast UDP: Predictable High Performance Bulk Data Transfer. In IEEE Cluster Computing, 2002.
- [16] Norman W. Paton, Malcolm P Atkinson, Vijay Dialani, Dave Pearson, Tony Storey, and Paul Watson. Database Access And Integration Services On The Grid. 2002.
- [17] The Globus Project. Towards Globus Toolkit 3.0: Open Grid Services Architecture. <http://www.globus.org/ogsa/>, retrieved on January 10, 2003.
- [18] R Project. Retrieved from <http://www.r-project.org>, January 10, 2003.
- [19] Salvatore J. Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, Dave W. Fan, and Philip K. Chan. JAM: Java Agents For Meta-Learning Over Distributed Databases. In Knowledge Discovery and Data Mining, pages 74–81, 1997.
- [20] W3c Semantic Web. Retrieved from www.w3.org/2001/sw/, September 2, 2002.
- [21] Tsunami. www.anml.iu.edu/anmlresearch.html, retrieved on April 4, 2003.
- [22] W3C. Semantic Web. retrieved from www.w3.org/2001/sw/, September 2, 2002.
- [23] A. Chien, T. Faber, A. Falk, J. Bannister, R. Grossman, and J. Leigh. Transport Protocols for High Performance: Whither TCP? Communications of the ACM, volume 46/11, pages 42-49, 2003.