Data Webs for Earth Science Data

Asvin Ananthanarayan,
Rajiv Balachandran, Yunhong Gu,
Robert Grossman, Xinwei Hong,
Jorge Levera, Marco Mazzucco
Laboratory for Advanced Computing
University of Illinois at Chicago

May, 2002; Revised June, 2003

This is a draft of a paper which later appeared in Parallel Computing, Volume 29, 2003, pages 1363-1379.

Abstract

We describe high performance data webs for earth science data which are designed for interactively analyzing small to moderate size remote data sets, as well as mining distributed data sets. Achieving high performance required developing specialized high performance transport services as well as specialized high performance middleware services for merging multiple data streams. Data webs complement data grids which are grid based infrastructures designed to support arbitrary distributed computation over distributed data using a trusted computing model.

Keywords: data webs, data grids, high performance web services, grid data, correlation keys

1 Introduction

Earth science data is of interest to a variety of communities, including scientists studying climate change, scientists in other disciplines interested in correlating earth science data with other data, and more casual users. There has been substantial work developing infrastructures for scientists who work with earth science data on a daily basis and are thoroughly familiar with the software and conventions required for working with it. In this paper, we describe a web based infrastructure designed to support the more casual exploration and use of earth science data.

We are in stage of transition with regard to how scientists and engineers use remote and distributed data. By and large, scientific and engineering data is still stored in data archives by file, retrieved by ftp, and analyzed locally on workstations. Often there is a web front end to the data archive which supports metadata queries and provides a convenient interface for identifying files of interest.

There is a growing consensus that over the next several years web-based interfaces to data archives will be supplemented by web-based and grid-based infrastructures for working with remote and distributed data.

In this paper, we describe data webs and how they can support scientists working with geographical data. We also describe some of the key ideas required for scaling data webs to work with larger data sets effectively, from the viewpoint of high end-to-end performance.

This paper describes the third generation of a data web we have developed for working with geographical and other multidimensional scientific and engineering data. The current generation understands multi-dimensional keys, which is essential for working with grid data. It relies on specialized high performance transport services and specialized high performance middleware services so that moderately large remote earth science data sets may be used as if they were local.

Section 2 describes related work. Section 3 describes data archives, data webs and data grids. Section 4 describes the key ideas behind data webs. Section 5 describes an architecture for high performance data web servers. Section 6 describes the implementation of a high performance data web server called Jupiter. Sections 7 describes experimental studies using Jupiter. Section 8 is the conclusion. There is also an appendix containing a typical interaction between a data web client and server.

2 Related Work

In this section, we describe some related work in rough historical order. First, we describe data exchange formats for earth science data. Second, we describe more recent XML based data formats for earth science data. Third, we describe some web and grid-based infrastructures for working with earth science data.

Scientific Data Interchange Formats. The goal of a data interchange format is to allow data to be used easily by different platforms and by different applications. There are dozens of different interchange formats, some of the most popular for earth science and atmospheric science data include:

1. Hierarchical Data Format (HDF) is a self describing data format orginally developed by the National Center for SuperComputing Applications (NCSA) and used for example, by the National Auronautics Space Administration (NASA) Earth Observing System (EOS).

2. Common Data Format (CDF) is a self describing data format for scientific data which supports scalar, vector and multidimensional data.

3. Network Common Data Form (NetCDF) is a self-describing, extendible file format for multidimensional data.

4. GRid In Binary (GRIB) is the World Meteorological Organization (WMO) standard for meteorological data in a grid format.

XML Based Interchange Formats. More recently, XML has begun to be used for providing a machine and platform independent language for earth science data. The problem with XML is that it is relatively verbose and is not ideal for vector and multidimensional data for that reason. On the other hand, XML has emerged as the format of choice for scientific metadata.

1. The Extensible Scientific Interchange Language (XSIL) is an XML language which defines common scientific data structures, such as tables, arrays, time, etc.

2. The Extensible Data Format (XDF) is an XML language which defines scalar and gridded data formats for scientific data.

3. The Earth Science Markup Language is an XML language for earth science data [8].

4. The Geography Markup Language (GML) is an XML language for geographic information, including both the geometry and properties of geographic features developed by the OpenGIS Consortium (OGC).

Web and Grid-Based Infrastructures. Both the data exchange formats and the XML languages described above are designed to work with archived data or data at rest. Web and grid-based services also provide a way to compute with data or to work with it interactively.

1. The Earth System Grid is a data grid for earth science data being developed by the National Center for Atmospheric Research (NCAR) and several Department of Energy (DOE) laboratories. The Earth Science Grid was built using the grid infrastructure [9] and more specialized data grid services [3]. Specialized data grid services include grid ftp for parallel data transport, the storage resource broker for access to hierarchical storage via relational metadata, and data replication services for accessing closer replicated data [5].

2. Digital Earth is a NASA led initiative to use an OpenGIS framework to allow GIS data residing on distributed servers to use each other's data [7].

3. Web Services [27] are a W3C standard based upon XML and Simple Object Access Protocol (SOAP) for defining middleware services described using the Web Services Description Language (WSDL). See [25] for related work on accessing and exchanging data using W3C standards.

3 Data Archives, Data Grids, and Data Webs

In this section, we describe how geographical data is currently accessed, in general, for analysis purposes and how it is likely to be accessed in the future. The foundation for this transition is 1) the geometric growth of network bandwidth and hard disk capacity and 2) the development of middleware services such as web services and grid services.

Data Archives. Most earth science data is stored in data archives today. The data is organized into data files, the data in the files is in HDF, NetCDF, or similar formats, and access to the data is via ftp at the file level. There is often a metadata database containing relational data describing the metadata and containing the name of the file. For ease of use, there may be a web based interface to the metadata database. For scalability, the files may be stored on a hierarchical storage system.

Data Webs. Data webs are designed for casual users to explore remote data and to do operations on distributed data that are template driven. In the same way that today's web enables the remote viewing of data with just a few clicks, tomorrow's data web will enable simple Exploratory Data Analysis (EDA) of remote and distributed data by casual users with just a few clicks.

Data Grids. Data grids are designed for power users who have specialized needs for large data sets or high end computational resources to distributed grid based computing applied to remote and distributed data. Because of the nature of the resources, they need to be scheduled and the middleware needs to Authenticate, Authorize, and schedule Access (AAA) to the data and computational resources. The AAA model is basically that of the mainframe or supercomputer. The architecture of data grids is described in [3] and [5]. More recently an effort has been made to integrate web services and grids [22], [10] and to support data derivation services [11].

Prior to the web, ftp sites served as document archives. Although it was counterintuitive at the time, allowing remote documents to be viewed using a browser with single click changed the way people worked, despite the fact that the alternative of using ftp and opening the file using a local word processor was not much more complicated.

Data is more complex than documents and the data archives are being replaced with two fundamentally different distributed platforms: data webs enable the remote exploratory analysis and distributed mining through templated operations of small to moderate data sets, while data grids enable arbitrary computations of moderate to large data sets. Since data grids are more demanding of resources, the resources must be authorized and scheduled and this is done through a middleware supporting authentication, authorization, access and scheduling. This is the mainframe or supercomputing framework for resource allocation. On the other hand data webs, like document webs, are designed for the casual exploration and browsing of remote and distributed data and typically are open to any interested party.

4 Data Webs - Basic Ideas

In this section, we give a quick introduction to some of the basic concepts underlying data webs. For more details, see [15]. Data webs are built on a few key concepts:

1. The basic data structure is a column of data. In data webs the basic data structure is a (distributed) column of data. Each column, or collection of columns, is associated with a key, called a universal correlation key or UCK.

2. Make it easy to merge distributed columns. One or more UCKS may be combined to create a vector valued UCK. UCKs are associated with globally unique IDs or GUIDs and data web services and applications use the GUIDs to determine whether two UCKs are in fact the same. As an

	Data Archives	**Data Webs**	**Data Grids**
Data Structure	remote files	distributed columns	distributed files with attribute valued access to data
Operations	retrieval	remote exploratory data analysis and distributed correlation	distributed computation
AAA	web model	web model	mainframe model
Protocols	ftp	dstp and web services	grid ftp
Design	simple access to files for casual users	simple access to data and EDA for casual users	high performance computing resources for power users
Middleware	ftp servers	dstp servers, data mining services	Globus

Table 1: Today, access to geographical data is primarily through data archives. Data archives are beginning to be supplemented by data webs and data grids. The table summarizes some of the basic distinctions between data archives, data webs, and grids. AAA is an abbreviation for authorization, authentication, and access.

example, a one degree by one degree latitude-longitude grid by six hour temporal grid used in a climate model would be identified by a triple of UCKs and corresponding GUIDs. Early versions of data web did not support vector valued UCKs which are essential for working with geographical data.

3. Separate data, metadata, and keys. Data web services and applications separate the keys, metadata and data. Given this separation, a data web application can express keys and metadata in XML, while the associated geographical data can be transported in a streaming protocol for efficiency and scalability. Other data web applications may make other choices, but data webs are designed for data and not documents and by supporting data, metadata and keys, certain operations can be implemented quite simply. For example, the metadata can specify a policy for specifying server side sampling of data.

4. Support basic, templated statistical operations. A typical sequence of data web operations is:

1. connect to two data web servers
2. set latitude, longitude and time as a vector valued UCK
3. retrieve precipitation from one server and ozone from the other
4. merge the precipitation and ozone fields using the UCKs
5. cluster the precipitation vs. ozone

Unlike semantic webs whose goal is to support general logical operations or data grids whose goal is to support general computation over data, the goal of a data web is to support certain templated operations such as those above.

5. Make it simple to add data. Setting up a data grid service is complicated because the AAA model requires the careful installation of complex middleware services. On the other hand, setting up a web document server is relatively easy. Data webs are modeled after the latter. Putting data into data webs consists of the following steps:

1. Install a data web server, which is a stand alone application.
2. Create the data file say, e.g., SNOW.dat containing rows and columns of data. For common data formats, such as NetCDF, there are drivers so that a data web server can directly access the data in its native format. Alternatively, the data may be imported into a relational database which is interfaced to a data web.

Command	**Description**
METADATA	retrieve different types of metadata including UCKs, file metadata, attribute metadata, and category metadata.
SET UCK	set the UCK
SET DataFileName	set the name of the data file
SET LINE	specify the first and last record to retrieve
SET SAMPLE RANDOM	specify a certain percentage of data to be sampled
SET TYPE	specify the type of connection, including text, binary socket, binary parallel socket, or binary SABUL connection
DATA	specify the data to be returned; a limited SQL like syntax is supported

Table 2: This table lists the commands used by the dataspace transfer protocol (dstp).

3. Append the name of the data file to one of the XML catalog files on a data web server. Catalog files contain the names of the data sets on a data web server. Catalog files may also be generated automatically.

4. Add a file SNOW.ds containing simple XML metadata about the data file, such as the UCKs, names of the attributes, their data types, etc., to the data web server.

A protocol called the DataSpace Transfer Protocol or DSTP has been developed to facilitate building data web applications using the simple ideas just described. For example, DSTP directly supports retrieving UCKs, metadata, and data; understands missing values; provides different server side sampling options; and, as we will see below, supports specialized transport protocols so that large data sets can be retrieved efficiently. A summary of DSTP can be found in Table 2 below [6]. A sample DSTP session can be found in Appendix A.

5 High Performance Data Webs

In the sections above, we argued that over time today's data archives will be replaced by data webs for templated operations on small to medium size data sets and by data grids for general computations on moderate to large size data sets.

End-to-End Performance. High performance data webs parallelize the data access and data transport of standard data webs running over commodity networks and using single workstations as the data servers. The goal is to parallelize the performance of the local i/o, the long haul network, and the application and also to maintain this performance through two interfaces:

1. First, the interface between the local file system holding the data web server data and the network.
2. Second, the interface between the network and the data web application.

The challenge is to make sure that neither the local i/o-to-network interface nor the network-to-application interface becomes the limiting factor in the application. This is sometimes referred to as the end-to-end network performance challenge. In this section, we describe an architecture to support this. In the sections below we describe some experimental studies based upon this architecture using a high performance data web server we have developed called Jupiter.

Our approach was to introduce two additional layers into a standard layered network model. The first new layer provided specialized, high performance transport services designed for long haul networks. The second new layer provided specialized high performance middleware services for efficiently striping and merging columns of data in order to exploit parallelism to improve performance. See Table 3.

High Performance Transport Protocols. It is common for initial implementations of wide area data intensive applications to have difficulty using the bandwidth of a high performance network effectively [19]. We have experimented with two specialized protocols to overcome this limitation.

The first is called PSockets and stripes data over multiple TCP based network sockets [12] to improve the effective bandwidth of an application. PSockets is used as an application library and does not require any changes to the operating system, such as changing the TCP window size.

Layer	**Protocols**	**Description**
Application	DSTP and P-DSTP applications	standard web and high performance web applications
Specialized Middleware	P-DSTP, P-Merge	specialized high performance middleware services over standard internet protocols
Standard Middleware	HTTP, DSTP, SOAP	standard and emerging internet protocols
Specialized Transport	PSockets, SABUL	provides high performance data transport over standard transport protocols
Standard Transport	TCP, UDP	standard transport protocols

Table 3: Our design of high performance data webs for geographical data introduced into new layers into a standard layered network model. The first new layer defines specialized high performance transport protocols over standard protocols such as TCP and UDP. The second new layer defines specialized high performance network services for working with multidimensional data over standard network services such as HTTP, SOAP, and newer protocols such as DSTP.

PSockets is limited, however, by its reliance on TCP. The major flaw of TCP for high performance applications is that its performance is affected by the latency inherent in global wide area networks.

To address this issue, we designed a second data transport protocol called Simple Available Bandwidth Utilization Library (SABUL) [21]. SABUL is a bulk data transfer library which uses rate controlled UDP to send data. What makes it novel is that it uses a TCP connection as a control channel to continuously update the communication state information of the data transfer. This allows for a more efficient exploitation of the available bandwidth, even when the bandwidth is changing over time.

The use of UDP in SABUL is a natural choice. Its header consists of destination and source addresses and has a field for checksum in most applications. UDP has no flow control, rate control, or reliable transmission mechanisms. Abstractly, it allows an application to work directly with the IP layer. The TCP channel is used for rate control, flow control, and reliable transmission. 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.

High Performance Middleware Services. We also developed two specialized middleware services to parallelize some of the common data web operations: Parallel DSTP (P-DSTP) is a parallel version of DSTP and P-Merge is a service which merges multiple data streams by UCKs.

P-DSTP is a minor extension of the DSTP protocol which enables a large data set to be served from multiple servers and received by one or more clients. The extension allows synchronization between servers and clients. It also uses a block transfer mode to enable fast reads from disk and quicker data transfers over a WAN. The metadata, provided by P-DSTP, enables the client to read the incoming blocked data stream. Thus from parallel disks, data can be served over parallel networks to parallel clients for processing.

The extension of DSTP to P-DSTP is natural. Since DSTP views data sets as columns of data, these columns can be distributed over multiple nodes. In turn, these multiple columns of data can be streamed over the network in parallel and merged on the client end with the use of UCKs.

Client DSTP applications often need to merge one or more columns of data by UCKs or columns extracted from multiple DSTP streams, say from P-DSTP. We also developed an algorithm for merging multiple DSTP data streams called Parallel Merge or P-Merge.

Here is a description of P-Merge from [16]. We assume the data sets are partially presorted. Without loss of generality, assume there are two data streams being drawn into a client and we are trying to join on one UCK. Fix integers K and N such that $K < N$. The client initially grabs two blocks, of some fixed size N, from both stream one and stream two. A sort is done on both blocks. A join is then attempted. Cursor a is placed on the first record of the first data block and cursor b is placed on the first record of the second data block. In step n, assume that cursor a is on record $m \leq n$. If cursor b points to a UCK whose value is greater than the UCK value plus t pointed to by cursor a than we move cursor a down one record. If cursor b points to a UCK value whose value is within t of the UCK value pointed to by a, then join the two records and move a and b down one. Finally, if a points to a UCK value which is greater than the UCK value plus the t pointed to by b, we move cursor b down one record. Clearly the complexity of this algorithm is $n \log(n) + n$. In practice, for partially sorted data, the cost can be much less.

Let M be the number of successful joins in the previous iteration. Define L to be the greatest of M and K. We refer to K as the *least window increment value.* It is the least amount of records that the window moves forward in one step. If either cursor a or b is on the N'th record of the block, we place L new records into the appropriate data block. The size of the data block remains unchanged. Without loss of generality, suppose cursor a is on the N'th record and the data is being received in ascending order. The new records are first placed in the memory locations of the successfully joined data records of data block one and then in the memory locations of the first $K - M$ records in data block one. Recall that the data in block one is sorted. Thus, by replacing the first $K - M$ records, we are replacing the records which have been in block one for the longest period of time. We continue this process until one or both data streams are exhausted.

6 The Design and Implementation of Jupiter

DataSpace is an open source implementation of a data web developed by the Laboratory for Advanced Computing at the University of Illinois at Chicago. It is designed using the DataSpace Transfer Protocol or DSTP, an open standard for data transport over data webs [6]. DataSpace can also use SOAP as an alternate data transport protocol [27].

In previous work, we designed and developed Mercury, a Java based

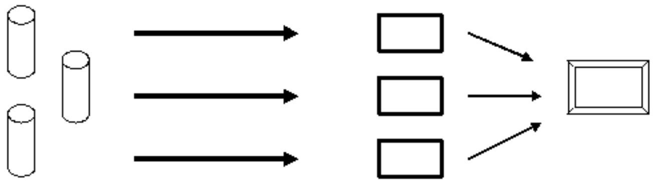


Figure 1: To achieve high end-to-end performance, the local parallel i/o streams are sent in parallel over a long haul network and managed by individual nodes of a cluster by the application. In the architecture described in this paper, the data transport is managed by specialized high performance transport services such as PSockets and SABUL. The striping of the data over DSTP and the merging of the data by UCKs is managed by specialized high performance middleware services as P-DSTP and P-Merge.

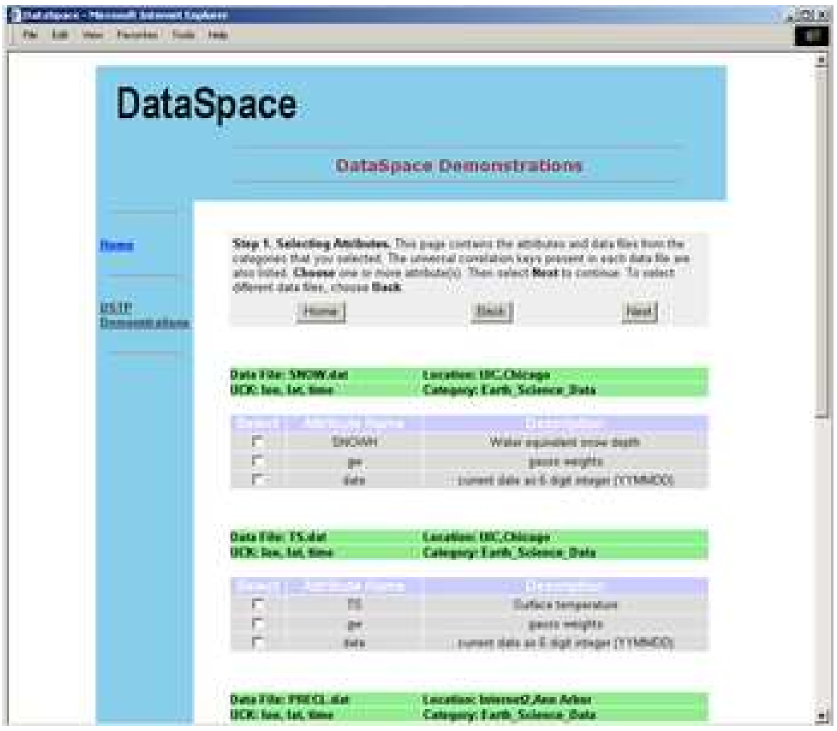
DSTP server for commodity networks [15]. Jupiter is a DSTP server written in C++ designed for high performance networks. Jupiter uses parallel i/o, parallel network transport, and specialized streaming operations to support the remote analysis and distributed mining of moderate to large distributed data sets requiring vector valued keys, such as geographical data.

Two different implementations of Jupiter were developed and compared. The first implementation uses MPIO [23] to read data striped over multiple nodes attached to a high speed interconnect. The data then is striped over the network with PSockets and P-DSTP. Each data node requires two network cards or one NIC card and one interconnect card supporting Scalable Coherent Interface (SCI) or myrnet. This places a high load on the system's PCI Bus and CPU.

The second implementation uses data nodes employing software Redundant Array of Independent Disks (RAID) and uses SABUL and P-DSTP for the transport over the long haul links. A server with this configuration is relatively inexpensive and can achieve transmission rates of over 500Mb/s. This solution has shown itself to be easier to scale, while requiring less hardware infrastructure. The second implementation is used for the experimental studies in the section below.

7 Experimental Studies using Jupiter

We developed a data web application using NCAR's Community Climate Model (CCM3) [4]. We imported several hundred Gigabytes of CCM3 data.


DataSpace

DataSpace Demonstrations

Home

DSTP Demonstrations

Step 1. Selecting Attributes. This page contains the attributes and data files from the categories that you selected. The universal correlation keys present in each data file are also listed. **Choose** one or more attribute(s). Then select **Next** to continue. To select different data files, choose **Back**.

Home Back Next

Data File: SNOW.dat **Location: UIC,Chicago**
UCK: lon, lat, time **Category: Earth_Science_Data**

Select	Attribute Name	Description
☐	SNOWH	Water equivalent snow depth
☐	gw	gauss weights
☐	date	current date as 6 digit integer (YYMMDD)

Data File: TS.dat **Location: UIC,Chicago**
UCK: lon, lat, time **Category: Earth_Science_Data**

Select	Attribute Name	Description
☐	TS	Surface temperature
☐	gw	gauss weights
☐	date	current date as 6 digit integer (YYMMDD)

Data File: PRECL.dat **Location: Internet2,Ann Arbor**
UCK: lon, lat, time **Category: Earth_Science_Data**

Figure 2: A sample page of the DSTP-based data web client.

The CCM3 data sets we used contained measurements of atmospheric and earth science properties on a 1 degree by 1 degree latitude-longitude scale for the past 120 years. The data was originally in a Network Common Data Format (NetCDF) [20] format, which we imported into a DSTP Server. The Data servers communicate with a client written in PHP that runs on a separate web server.

Communication between the client and server use DSTP. A typical interactive query might compare surface temperature data from one DSTP server with snow depth data from a second DSTP server. A typical query is of the form "Retrieve surface temperature and snow depth values for all latitude and longitude values for January 1998 and correlate them."

Figures 2 and 3 are sample output of our PHP client. The X-axis gives the latitude values, the Y-axis gives the Snow depth values and the Z-axis gives the Surface temperature values. The correlation was done on the UCKs latitude, longitude and time. Server side sampling was done from data files containing approximately 13 million rows. As seen in the figure, we have

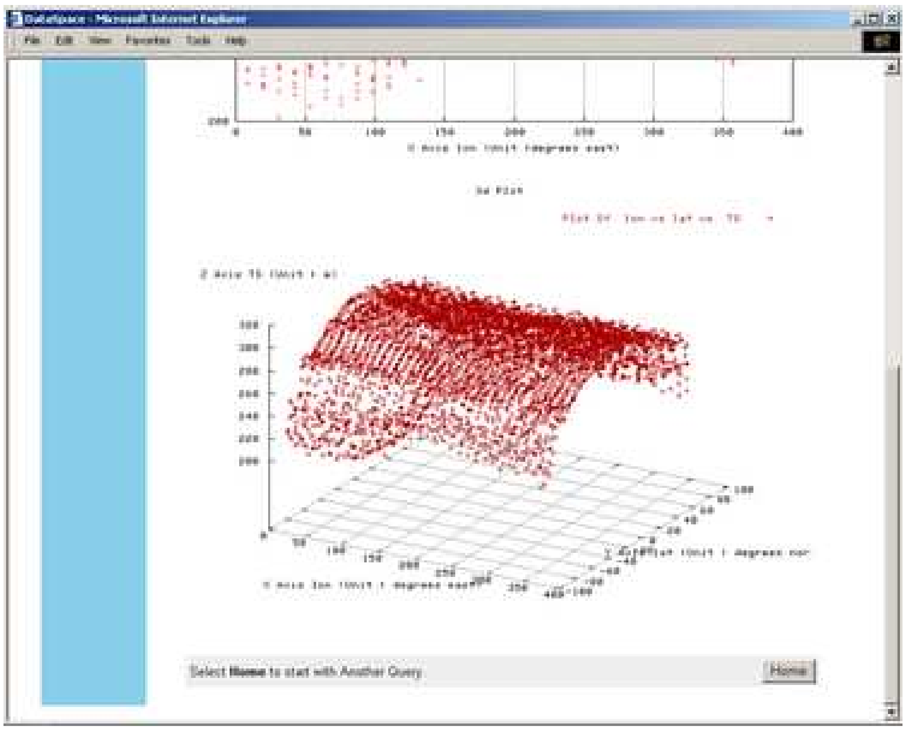

Dataspace - Microsoft Internet Explorer
File Edit View Favorites Tools Help
3d Plot
Select Home to start with Another Query
Home

Figure 3: A 3D plot from the DSTP-based data web client.

Records	**File Size** *MB*	**Get & Merge** *sec*	**Plot** *sec*	**Total Time** *sec*
10,000	0.39	2	≤ 1	3
20,000	0.79	4	≤ 1	5
30,000	1.18	5	2	7

Table 4: Performance timings for PHP web client

lower values of surface temperature and higher snow depth as we go towards the poles. Higher values of surface temperature and lower values of snow depth are observed as we move towards the equator. We have also used the PHP client to make distributed queries, such as comparing El Nino to cholera outbreaks, as in [2]. See Table 4 for some typical times for queries of this type using the PHP client and DSTP Server for data sets with records between 10,000 and 30,000 records.

Making queries like this interactive on distributed data sets containing millions of rows requires using the specialized high performance transport services and specialized high performance middleware services described above. In the remainder of this section, we describe some experiments showing the scalability of these services.

Table 5 from [21] details the results of using the specialized transport services PSockets and SABUL between a node at Ann Arbor, MI and a node at NCAR, Boulder, CO. Both nodes used the Linux 2.2.17 kernel. They had 256 MB of RAM and 100 Mb/s fast ethernet cards. These two machines are interconnected by Internet 2's Abilene network with an OC-3 uplink (155 Mb/s). The bottleneck in this experiment is the Fast Ethernet cards. In comparison, an application which does not exploit parallelism or tune the network in any way would typically see a rate of no more than 10 Mb/s.

Table 6 demonstrates the scalability of our P-Merge algorithm which merges multiple data streams by their UCKs. Our goal was to design an algorithm which could merge two or more data streams at high speed and trade accuracy for interactivity. As more work is required to sort the streaming data the performance degrades as expected. For preliminary exploratory data analysis, this is often acceptable.

8 Summary and Conclusion

In this paper we have described how data webs can support working with small to medium remote geographical data sets interactively. Data webs, unlike data archives and data grids are also designed to support the casual

Specialized Data Transport Used	**Rate in Mb/s**	**Packet Loss Percentage**
Application without window tuning or parallelism	< 10	-
Iperf with TCP window tuning	83.3	0.0
PSockets with 19 parallel sockets	85.27	0.085
SABUL	95.63	0.95

Table 5: This table summarizes the performance gain by using specialized transport services such as PSockets and SABUL when moving data between NCAR and Ann Arbor. Both Iperf and PSockets use TCP for data transmission. Using Iperf along with TCP window tuning we obtained a throughput of 83.3 Mb/s. PSockets detected the best number of parallel sockets to be 19. The maximum throughput obtained using PSockets was found to be 85.27 Mb/s. The maximum packet loss percentage we have observed during a TCP transmission over a well tuned Abilene network was 1 percent. Therefore SABUL was tuned to run with a maximum packet loss of 1 percent. The percentage of packet loss is given in the last column. Table 1 shows that SABUL's performance is superior to the throughput results obtained by Iperf and PSockets. The data is from [21].

Rand %	**Match %**	**Fetch Time**	**Merge Time**	**Total Time**
2	100	0.638	0.64	1.279
10	87	0.638	0.643	1.281
20	72	0.638	0.691	1.329
33	49	0.638	0.727	1.365

Table 6: This table summarizes the performance of the specialized middleware services we developed for merging multiple streams of data by their UCKs. The algorithm is a windowed merge from [16]. The overall processing rate for merging two data sets by latitude, longitude and time is about 170 Mb/s. If the data is ordered then merging can sometimes be done at line speed. As the data becomes more disordered (measured by the randomness column rand), either the processing speed must be reduced or some unordered records (measured by the match rate column) must be passed through. This is a basic tradeoff in exploratory data analysis, the tradeoff between interactivity and accuracy. The data is from [16]. All times are in seconds.

browsing of remote data and the casual mining of distributed data.

We developed and tested two different high performance servers, both of which used the open DataSpace Transfer Protocol or DSTP for communication. We operated a distributed testbed containing several hundred Gigabytes of NCAR CCM3 data and tested specialized high performance transport and middleware services.

Ease of use is facilitated by using simple mechanisms for importing data into data webs, assigning globally unique keys (UCKs) so that different data sets can be compared easily, providing simple access at the application level to data sets and attribute metadata, and supporting templated operations such as correlation and clustering.

Interactivity is achieved by: 1) building data web applications over specialized high performance data transport protocols such as PSockets and SABUL for moving data efficiently over high performance long haul networks; and 2) using specialized high performance middleware services such as P-DSTP and P-Merge so that several data streams can be transported in parallel and efficiently merged by the application.

Appendix A. A Sample DSTP Session

The Data Space Transfer Protocol(DSTP) is a protocol for transporting data over data webs that supports some basic data operations such as getting keys, getting metadata, range queries, server side sampling, missing values, etc. The DSTP server uses a stream connection and simple SMTP (Simple Mail Transfer Protocol) or NNTP (Network News Transfer Protocol) style commands and responses. It is designed to accept connections from hosts and to provide a simple interface to the data columns on the server. A DSTP server functions as an interface between DSTP applications and remote data.

Rather than describe the protocol and commands formally, this section contains an annotated session between a DSTP Server named S and a DSTP client named C (not their real names). First we see the client contacting the server on TCP port 5040

```
C: Trying 131.193.181.125...

C: Connected to ncdm125.lac.uic.edu (131.193.181.125).

S: Escape character is '^]'.
```

```
S: 200 ncdm125.lac.uic.edu DSTP server v2.0 ready

S: .
```

The client can now request which Universal Correlation keys (UCKs) the server is prepared to serve. Latitude is assigned the globally unique UCK ID 11111, Longitude is assigned 11112 and time is assigned 11113.

```
C: metadata

S: 210 MetaData follows

S:   <UCK NAME="Latitude" ID="11111" >

S:   </UCK>

S:   <UCK NAME="Longitude" ID="11112" >

S:   </UCK>

S:   <UCK NAME="Time" ID="11113" >

S:   </UCK>

S: .
```

The client requests that the server set these indexes for correlation.

```
C: set uck Latitude

S: 230 UCK selected

S: .

C: set uck Longitude
```

```
S: 230 UCK selected

S: .

etc.
```

Note that the client is connected to ncdm125.lac.uic.edu. Next, the client can ask what data files the server has which contains these UCKs.

```
C: metadata uck latitude server ncdm125.lac.uic.edu

S: 210 MetaData follows

S:  <SERVER  NAME="ncdm125.lac.uic.edu" LOCATION="UIC,Chicago" >

S:   <DATAFILE NAME="SNOW.1870-1998.dat"
         ADDRESS="/raid/125/sc2001/DATA/GridFlat/"
         NUMRECORDS="127303367"
         DSFILENAME="SNOW.1870-1998.ds"
         DATE="10/11/2001"
         SOURCE="NCAR"
         DESCRIPTION="Snow Data"
         TYPE="ASCII"
         DELIMITER=" " >

S:   </DATAFILE>

S:   <DATAFILE NAME="PS.1870-1998.dat"
         ADDRESS="/raid/125/sc2001/DATA/GridFlat/"
         NUMRECORDS="127303367"
         DSFILENAME="PS.1870-1998.ds"
         DATE="09/27/2001"
         SOURCE="NCAR"
         DESCRIPTION="Surface Pressure"
         TYPE="ASCII"
         DELIMITER=" " >

S:   </DATAFILE>
```

```
etc.
```

The client then selects the desired file, in our case that is SNOW.1870-1998.dat

```
C: set datafile SNOW.1870-1998.dat

S: 240 DataFile selected

S: .
```

After the data file is selected, the client can request the corresponding attribute metadata, as in the conversation below. In the same fashion, the server can request the data itself, using a SQL-like syntax.

```
C: metadata uck latitude server ncdm125.lac.uic.edu
   datafile SNOW.1870-1998.dat

S: 210 MetaData follows

S: <DATAFILE NAME="SNOW.1870-1998.dat"
         ADDRESS="/raid/125/sc2001/DATA/GridFlat/"
         etc.
         DELIMITER=" " >

S: <ATTRIBUTE-DESCRIPTOR
         NUMBER="1"
         NAME="lat"
         DATA-TYPE="real"
         UNIT="degrees_north"
         Note="latitude"
         UCKID="11111"
         UCKNAME="lat"
         MIN="-87.09652"
         MAX="87.86379" >
```

```
S: </ATTRIBUTE-DESCRIPTOR>

     etc.
```

References

[1] Data Mining Group, The Predicitive Markup Language (PMML), A Markup Language for Statistical and Data Mining Models, Version 2.0. Retrieved from www.dmg.org on January 10, 2002.

[2] Percedes Pascual, Xavier Rodo, Stephen P. Ellner, Rita Colwell, Menno J. Bouma, Cholera Dynamics and El Nino-Southern Oscillation, Volume 289, 2000, pp. 1766-1769.

[3] Protocols and Services for Distributed Data-Intensive Science. A. Chervenak, I. Foster, C. Kesselman, S. Tuecke. ACAT2000 Proceedings, pp. 161-163, 2000.

[4] National Center for Atmospheric Research's Community Climate Model 3. Retrieved from www.cgd.ucar.edu/cms/ccm3 on May 10, 2002.

[5] The Globus Data Grid Effort. Retrieved from www.globus.org/datagrid on May 10, 2002.

[6] DataSpace Transfer Protocol (DSTP). See www.dataspaceweb.net.

[7] Digital Earth. Retrieved from www.digitalearth.gov on May 10, 2002.

[8] Earth Science Markup Language (EMSL). Retrieved from esml.itsc.uah.edu on May 10, 2002.

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

[10] I. Foster, C. Kesselman, J. Nick and S. Tuecke, Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum, retrieved on June 22, 2002 from www.ggf.org.

[11] Ian Foster, Jens Vockler, Michael Wilde and Yong Zhao, Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation, 14th International Conference on Scientific and Statistical Database Management, 2002.

[12] 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, Proceedings of Supercomputing 2000, ACM, 2000.

[13] Robert Grossman, Emory Creel, Marco Mazzucco, Roy Williams, A DataSpace Infrastructure for Astronomical Data, in R. L. Grossman, C. Kamath, W. Philip Kegelmeye, V. Kumar, and R. Namburu, Data Mining for Scientific and Engineering Applications, Kluwer Academic Publishers, 2001.

[14] R. L. Grossman, Standards and Infrastructures for Data Mining, Communications of the ACM, 2002.

[15] Robert Grossman, and Marco Mazzucco, DataSpace - A Web Infrastructure for the Exploratory Analysis and Mining of Data, Computing in Science and Engineering, 2002.

[16] Marco Mazzucco, Asvin Ananthanarayan, and Robert L. Grossman, Jorge Levera, and Gokulnath Bhagavantha Rao, Merging Multiple Data Streams on Common Keys, submitted for publication.

[17] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, Iperf Version 1.6, May 2002. Retrieved from dast.nlanr.net/Projects/Iperf on May 20, 2002.

[18] Brian Kantor and Phil Lapsley, Network News Transfer Protocol, February 1986, RFC 977. Retrieved from www.w3.org/Protocols/rfc977/rfc977.html on January 10, 2002.

[19] H. Kargupta and P. Chan, editors, Advances in Distributed and Parallel Knowledge Discovery, AAAI Press/The MIT Press, Menlo Park, California, 2000.

[20] Unidata NetCDF. Retrieved from www.unidata.ucar.edu/packages/netcdf on May 10, 2002.

[21] H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, Q. Zhang, Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks, submitted for publication.

[22] The Globus Project, Towards Globus Toolkit 3.0: Open Grid Services Architecture, retrieved from http://www.globus.org/ogsa/ on January 10, 2003.

[23] ROMIO: A High-Performance, Portable MPI-IO Implementation. Retrieved from http://www-unix.mcs.anl.gov/romio on May 10, 2002.

[24] Rajeev Thakur, William Gropp, and Ewing Lusk, On Implementing MPI-IO Portably and with High Performance, in Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems, May 1999, pages 23–32.

[25] Web Architecture: Describing and Exchanging Data, W3C Note 7 June 1999. Retrieved from www.w3.org/1999/04/WebData on February 10, 2002.

[26] The Semantic Web. Retrieved from www.w3.org/2001/sw/ on February 10, 2002.

[27] Web Services Activity. Retrieved from www.w3.org/2002/ws on May 10, 2002.