Papyrus: A System for Data Mining over Local and Wide Area Clusters and Super-Clusters

S. Bailey, R Grossman[*], H. Sivakumar, and A. Turinsky
National Center for Data Mining
University of Illinois at Chicago

May, 1999

This is a draft of the following publication: R. L. Grossman, S. Bailey, A. Ramu, B. Malhi and H. Sivakumar, A. Turinsky, Papyrus: A System for Data Mining over Local and Wide Area Clusters and Super-Clusters, Proceedings of Supercomputing 1999, IEEE.

1 Introduction

Data mining is the semi-automatic discovery of patterns, correlations, changes, associations, and anomalies in large data sets. Traditionally, in a broad sense, statistics has focused on the assumption-driven analysis of data, while data mining has focused on the discovery-driven analysis of data. By discovery-driven, we mean the automatic search or semi-automatic search for interesting patterns and models.

With the explosion of the commodity internet and the emergence of wide area high performance networks, mining distributed data is becoming recognized as a fundamental scientific challenge. In this paper, we introduce a system called Papyrus for distributed data mining over commodity and high performance networks and give some preliminary experimental results about its performance. We are particularly interested in data mining over clusters of workstations, distributed clusters connected by high performance networks (*super-clusters*), and distributed clusters and super-clusters connected by commodity networks (*meta-clusters*).

As a motivating example taken from [7], consider the problem of searching for correlations between twenty five years of sunspot data archived on a server in Boulder and 80 years of Southern night marine air temperature data archived on a server in Maryland. The goal of this data mining query might be to understand whether sunspots are correlated with climatic shifts in temperature. Notice that

[*]Point of Contact: grossman@uic.edu, http://www.ncdm.uic.edu. R. Grossman is also with Magnify, Inc.

this is much different than a traditional search which would simply return all documents containing the keyword sunspots. In the recent past, these types of queries would have been impossible without first moving all of the data to a central location and then analyzing it within the memory of a single workstation. With next generation networks, some data mining queries can be done while leaving the data in place, as we have shown in several recent demonstrations [5]. This provides a fundamentally new technology, since, in fact, most data is distributed.

Concretely, data mining may be viewed as extracting a learning set from one or more (distributed) data warehouses and applying a data mining algorithm to produce a predictive model or rule set [8]. Different strategies are possible, depending upon the data, its distribution, the resources available, and the accuracy required:

MR (Move Results) Today's commodity networks can be used to move the results of local data mining computations to a central site.

MM (Move Models) Commodity networks can also be used to move predictive models from site to site. See [1] for descriptions of several systems employing these first two strategies.

MD (Move Data) Next generation broadband networks also create the possibility of moving large amounts of data [6].

The majority of the work in distributed data mining has focused on the first two strategies MR and MM [1]. In this paper, we introduce Papyrus, a distributed data mining system supporting all three strategies, as well as mixtures of the strategies. We also describe experimental studies for strategies MD and MR.

2 Background and Related Work

In this section, we provide some background material and discuss some related work in this area. With the exception of [17] and [12], the work we know of in this area is limited to data mining over commodity networks. This section is based in part on [7].

Several systems have been developed for distributed data mining. Perhaps the most mature are: the JAM system developed by Stolfo et. al. [16], the Kensington system developed by Guo et. al. [9], and BODHI developed by Kargupta et. al. [10]. These systems differ in several ways:

Data strategy. Distributed data mining can choose to move data, to move intermediate results, to move predictive models, or to move the final results of a data mining algorithm. Distributed data mining systems which employ *local learning* build models at each site and move the models to a central location. Systems which employ *centralized learning* move the data to a central location for model building. Systems can also employ *hybrid learning*, that is, strategies which combine local and centralized learning. JAM, Kensington and BODHI all employ local learning.

Task strategy. Distributed data mining systems can choose to coordinate a data mining algorithm over several sites or to apply data mining algorithms independently at each site. With *independent learning*, data mining algorithms are applied to each site independently. With *coordinated learning,* one (or more) sites coordinate the tasks within a data mining algorithm across several sites.

Model Strategy. Several different methods have been employed for combining predictive models built at different sites. The simplest, most common method is to use *voting* and combine the outputs of the various models with a majority vote [2]. *Meta-learning* combines several models by building a separate meta-model whose inputs are the outputs of the various models and whose output is the desired outcome [16]. *Knowledge probing* considers learning from a black box viewpoint and creates an overall model by examining the input and the outputs to the various models, as well as the desired output [9]. Multiple models, or what are often called ensembles or committees of models, have been used for quite a while in (centralized) data mining. A variety of methods have been studied for combining models in an ensemble, including Bayesian model averaging and model selection [15], stacking [20], partition learning [4], and other statistical methods, such as mixture of experts [21]. JAM employs meta-learning, while Kensington employs knowledge probing.

Papyrus is designed to support different data, task and model strategies. For example, in contrast to JAM and Kensington, Papyrus can not only move models from node to node (Strategy MM), but can also move data from node to node (Strategy MD) when that strategy is desired. In contrast to BODHI, Papyrus is built over a data warehousing layer which can move data over both commodity and high performance networks. Also, Papyrus is a specialized system which is designed for clusters, meta-clusters, and super-clusters, while JAM, Kennsington and BODHI are designed for mining data distributed over the internet.

All four systems make use of Java. JAM employs Java applets to move machine learning algorithms to distributed data. Kensington uses Java JDBC to mine distributed data. Papyrus uses Java aglets [11].

Moore [12] stresses the importance of developing an appropriate storage and archival infrastructure for high performance data mining and discusses work in this area. The distributed data mining system developed by Subramonian and Parthasarathy [17] is designed to work with clusters of SMP workstation and like Papyrus is designed to exploit clusters of workstations. Both this system and Papyrus are designed around data clusters and compute clusters. Papyrus also explicitly supports clusters of clusters and clusters connected with different types of networks.

3 The Fundamental Trade-Off

In distributed data mining, there is a fundamental trade-off between the accuracy and the cost of the computation. Our interest is in cost functions which reflect both computation costs and communication costs, especially the cost of wide area communications. At one extreme, we can ship all the data to a single node. We assume that this produces the most accurate result. In general, this is also the most expensive. At the other extreme, we can process all the data locally obtaining local results, and combine the local results at the root to obtain the final result. In general, this approach is less expensive, but also less accurate.

We take the viewpoint that the data mining process consists of applying data mining algorithms to learning sets to produce predictive models. We view the predictive model very concretely as consisting of a PMML file [8]. Continuing, the PMML file can be used to process data further to obtain results, which we view as a vector.

We assume that there are n different sites connected by a network. One of the nodes, say K^{th}, is the network *root* where the overall result will be computed. With this viewpoint, a node can employ one of three different strategies [19]:

MD Move Data. Ship raw data D across the network to another node for processing.

MM Move Models. Process the data locally and ship the predictive model M to another node for further processing.

MR Move Results. Process the data locally until a result is obtained and ship the result R to another node for further processing.

In general, this progression results in a loss of accuracy, but a decrease in the cost. The process moves data, models, and results from node to node until the root produces a final result.

For some problems, sufficient accuracy is desired so that all the data must be moved to a central root. For other problems, sufficient speed is desired so that all the computation is done locally. Recall that it takes approximately 24 hours to move a terabyte of data over a OC-3 network. Given terabytes of data distributed over a meta-cluster of clusters and super-clusters, the correct strategy can mean the difference between waiting minutes instead of hours.

To simplify the discussion, we only consider mixed strategies involving MD and MM; the general case can be handled similarly. See [19]. A *strategy* X as a matrix of numbers

$$X = [x_{ij}]_{i,j=1}^{n}$$

where x_{ij} is the amount of data D_i that is moved from the i^{th} node to the j^{th} node for processing. After the move, the amount of data at the j^{th} node becomes $\tilde{D}_j$. We assume that when data is processed into a predictive model, its amount is compressed uniformly for each node with a coefficient α.

The cost of processing data at i^{th} node into a predictive model is $\bar{c}_i$ dollars per Gigabyte and the optimal cost of moving data from i^{th} to j^{th} node via the cheapest route between the two nodes is c_{ij} dollars per Gigabyte. The *overall cost function* for a strategy X is easily computed as

$$C(X) = \sum_{ij} (c_{ij}x_{ij} + \bar{c}_j x_{ij} + c_{jK}\alpha x_{ij}) = \sum_{ij} \hat{c}_{ij} x_{ij}.$$

A critical observation is that in many cases the cost function is convex [19]. In such cases, the least possible error ϵ_0 occurs when all data is moved to a single node and the largest possible error $\epsilon_{\max}$ occurs when the data is evenly distributed among all nodes. Note that the vector $\tilde{D}(X) = \left(\tilde{D}_1, \ldots, \tilde{D}_n\right)$ defines the relevant data distribution, where each $\tilde{D}_j = \sum_i x_{ij}$. We use the following form of the *error function* for a strategy X:

$$\epsilon(X) = \epsilon_0 + \rho_D \left(1 - \frac{\|\tilde{D}(X)\|}{|D|}\right), \qquad \rho_D = \frac{\epsilon_{\max} - \epsilon_0}{1 - \frac{1}{\sqrt{n}}}$$

where $|D| = \sum_j \tilde{D}_j$ is the overall amount of data in the network, $\|\tilde{D}\|$ is the usual Euclidean norm of a vector $\tilde{D}(X)$, and ρ_D is a scaling coefficient. Other forms of $\epsilon(X)$ are possible, each being convex. A strategy must satisfy the condition $\epsilon(X) \leq \epsilon_{tol}$ where ϵ_{tol} is an error tolerance threshold.

It is easy to find examples [19] in which there are intermediate strategies that are more cost effective than the naive ones of either leaving all the data in place or moving all the data to a single fixed node.

Because of this, it makes sense to design distributed systems with the flexibility of moving data, moving predictive models, or moving the results of local computations. This is one of the key ideas behind the Papyrus system. Papyrus, as far as we know, is the first distributed data mining system to be designed with this level of flexibility.

4 Papyrus

Papyrus is a layered system of software tools and network services for distributed data mining and data intensive computing. Papyrus applications can be designed to access any of the following four layers, as required. In this paper, we restrict attention to applications that move data (MD) from node to node using the bottom layer or applications that move models (MM) or results (MR) from node to node using the top layer.

> *A data management layer called Osiris.* At the lowest level is a data management layer specifically designed to support data mining of clusters, meta-clusters, and super-clusters. This can be thought of as a global data warehouse. Osiris divides data into units called folios and divides folios into smaller units called segments. Osiris can then

move segments from node to node as required for a computation. In practice, it is only feasible to move segments within clusters and super-clusters, that is using broadband, high performance networks.

A data mining layer. We view data mining as the extraction of a learning set from a data warehouse and the semi-automatic production, with the appropriate data mining algorithm, of a predictive model or rule set. We have developed with others an XML mark up language for predictive models and rules called the Predictive Model Markup Language (PMML) [13]. Hence the input to the data mining layer are learning sets managed by Osiris and the outputs are PMML files managed by the predictive modeling layer Anubis. Currently, this layer consists of application specific code. Several of our applications use a slightly modified form of C4.5 [14].

A predictive modeling layer. Eventually, we plan to employ a separate layer above the data mining layer for managing predictive models. Papyrus currently does not have one and instead uses an agent layer described below for managing predictive models.

An agent layer called Bast. The role of this layer is to identify relevant clusters within a meta-cluster or super-cluster, relevant data sets within the clusters, relevant attributes for particular queries, and relevant strategies for moving data, models and results. The agent layer gains information about clusters and the data and information contained there by examining local files, which express the information using an XML mark up language called DSML (Data Space Markup Language) [3], which we have developed.

Status. A preliminary version of Papyrus was demonstrated using the Terabyte Challenge Testbed [18] at Supercomputing 98 in Orlando during November, 1998. Currently, Papyrus has two modes: it can use Osiris to move segments (Strategy MD) from node to node as required for computation. It can also use Bast to move predictive models (Strategy MM) from node to node. We are currently designing Papyrus to support optimal strategies, which given an acceptable error, moves data, models, and results to achieve the error with minimum possible cost.

5 Preliminary Experimental Results - Osiris

Four compute servers and four data servers were used for the experiments. The four data servers were located at the University of Illinois at Chicago. The four compute clusters were located at Highway 1 in Washington, D. C. The compute and the data clusters were connected by the NSF/MCI vBNS Network. Even though vBNS is an OC-3 network offering maximum bandwidth of 155 Mb/s, the end nodes at Highway 1 were connected via a DS-3 link, which limited the maximum bandwidth of the testbed to 45 Mb/s.

We tested Papyrus' implementation of a MD strategy with an application in high energy physics. The application consists of event collision data from the CDF detector at the Department of Energy's Fermi National Accelerator Laboratory. The data is stored as event objects using Osiris. The analysis routines were designed by the physicists to verify the existense of the Top Quark from the measurements.

To better understand the MD strategy, we populated the data using two different schemas. In the first case, all the event data and its metadata were stored as a single object. We call this a *Horizontal Store* since this can be viewed like the rows in a database. In the second case, the event data and metadata were stored attribute by attribute. We call this a *Vertical Store*, since this can be viewed like the columns of a database. The metadata was analyzed by the query to find out which attributes were required and only those attributes were moved.

Tables 1 and 2 show the performance results of running various processes on the four compute clusters. Note that for the Horizontal Store the entire data set was transferred to the compute cluster. This is advantageous in applications where the whole object is required to be analyzed. When we have a high speed network and these types of queries, the Horizontal Store works best. The queries using the Vertical Store ended up moving less data, since the particular query used relatively few attributes per record and attributes were moved by segment (and not by record) from the data to the compute server. Due to this fact, there were more events analyzed per second and the query completed faster with Vertical Store. Different queries favor different layouts.

One of the experiments that was conducted was to run multiple processes on the same query, with the processes working on different sections of the store. Note from Table 1 that we achieved near linear speed up by increasing the number of processes in this way.

Observe that the aggregate raw data bandwidth used by the application increased as we increased the number of processes. While using the Horizontal Store, we obtained an aggregate bandwidth of 34.93Mb/s of the acheivable 45Mb/s showing an efficiency of 77.6%. These experiments would be simply impractical with commodity networks.

CS	P/CS	ADR in Mb/s	TDT in Giga bytes	TTT in seconds	EPR in Events/second
1	1	3.06	4.4	11777.5	64
1	2	6.07	4.4	5926.59	253
1	4	10.05	4.4	3590.40	655
1	8	16.92	4.4	2132.05	2811
4	1	11.36	4.4	3170.7	947
4	2	19.44	4.4	1854.97	3231
4	4	23.32	4.4	1550.91	7731
4	8	34.93	4.4	1032.24	23245

Table 1: Horizontal Store : Data Servers used = 4, Store Size = 4.4 Giga bytes
CS - Compute servers,

P/CS - Processes per compute server,
ADR - Aggregate Data Rate,
TDT - Total data transferred,
TTT - Total time taken for completion of application,
EPR - Events processing Rate

CS	P/CS	ADR in Mb/s	TDT in Mega bytes	TTT in seconds	EPR in Events/second
1	1	1.39	269.5	1549.05	400
1	2	2.75	269.5	797.00	1554
1	4	3.81	269.5	566.42	4377
1	8	6.75	269.5	320.45	15482
4	1	4.34	269.5	496.32	4997
4	2	6.73	269.5	322.13	15392
4	4	9.74	269.5	223.05	44590
4	8	13.96	269.5	152.52	126918

Table 2: Vertical Store : Data Servers used = 4, Store Size = 4.0 Giga bytes

6 Preliminary Experimental Results - Bast

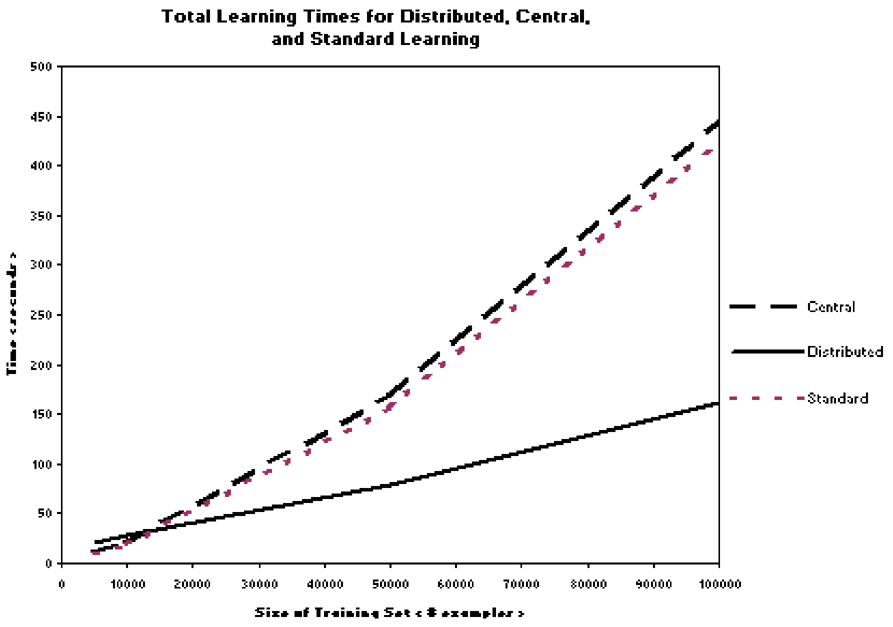
In this section we review our preliminary experiments when Papyrus employs the MM strategy using its agent layer Bast. The data consists of health care outcome data from the University Health Consortium (UHC). We distributed the outcomes data by zip code for this series of experiments, built models locally using the tree based classifier C4.5 [14], and assembled the results using Bast.

The tests were run on 24 IBM RS/6000 workstations distributed between the University of Illinois at Chicago, the University of Pennsylvania, and the University of Maryland. The workstations were part of the NSCP's Terabyte Challenge Testbed [18].

The UHC outcomes data was partitioned by zip code and distributed over the 24 nodes in order to simulate inherently distributed data. The data contained 115 attributes; we chose 29 attributes on which to build models. The data set has both continuous and discrete attributes. Eighty percent of the data was distributed over the 24 nodes and used for the (distributed) training set. Twenty percent of the data was stored at a single node and used for the validation set.

Bast was used to distribute queries to the 24 nodes, local models were built using C4.5 on the data available at each node, and the PMML models were returned by Bast and combined via a simple majority vote. See [6] for additional details.

The experiments were conducted on training distributed data sets ranging from 1000 records to 100,000 records. The experiments compared total learning time of distributed learning (Move Models) vs centralized learning (Move Data). For completeness we included the time required for standard learning, where we start with the data in a single location and build a single model. The total



learning time is the time it takes to transfer any data, build/transfer models, and score the validation set back at the control workstation.

Our preliminary results indicate that our Bast implementation of distributed learning is faster than both centralized learning and standard learning. It must be cautioned that error rates for distributed learning might be highly dependant on the partitioning of the data set and/or other factors. More extensive tests under currently under way.

7 Conclusion

Most data is distributed and most data is unexamined. Distributed data mining holds the promise of extracting some useful information from this data.

Currently, most experiments in distributed data mining have focused on strategies which build local models and move the models or the results of applying the models to data to a central location (Move Models, Move Results). With the emergence of broadband, high performance networks, it is becoming feasible also to move the data (Move Data).

In this paper, we point out that mixed strategies can outperform either the Move Model or Move Data strategy. We describe a distributed data mining system called Papyrus which supports Move Data, Move Models, Move Results, or mixed strategies. We also give preliminary experimental results showing the ability of Papyrus to effectively mine data using the Move Model or Move Data strategy.

References

[1] P. Chan and H. Kargupta, editors, Proceedings of the Workshop on Distributed Data Mining, The Fourth International Conference on Knowledge Discovery and Data Mining New York City, 1999, to appear.

[2] T. G. Dietterich, Machine Learning Research: Four Current Directions, AI Magazine Volume 18, pages 97-136, 1997.

[3] DataSpace, Protocols and Languages for Distributed Data Mining, http://www.ncdm.uic.edu.

[4] R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, "Data Mining and Tree-based Optimization," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.

[5] The Terabyte Challenge: An Open, Distributed Testbed for Managing and Mining Massive Data Sets, Proceedings of the 1998 Conference on Supercomputing, IEEE.

[6] Robert Grossman, Stuart Bailey, Simon Kasif, Don Mon, Ashok Ramu and Balinder Malhi, The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters, Meta-Clusters and Super-Clusters, Proceedings of the Workshop on Distributed Data Mining, The Fourth International Conference on Knowledge Discovery and Data Mining New York City, P. Chan and H. Kargupta, editors, to appear.

[7] R. L. Grossman, An Overview of High Performance and Distributed Data Mining Systems, submitted for publication.

[8] R. L. Grossman, S. Bailey, A. Ramu, B. Malhi, P. Hallstrom, I. Pulleyn, and X. Qin, The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML), Information and System Technology, Volume 41, pages 589-595, 1999.

[9] Y. Guo, S. M. Rueger, J. Sutiwaraphun, J.; and J. Forbes-Millott, Meta-Learnig for Parallel Data Mining, in *Proceedings of the Seventh Parallel Computing Workshop,* pages 1-2, 1997.

[10] H. Kargupta, I. Hamzaoglu and B. Stafford, Scalable, Distributed Data Mining Using an Agent Based Architecture, in D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings the Third International Conference on the Knowledge Discovery and Data Mining,* AAAI Press, Menlo Park, California, pages 211-214, 1997.

[11] D. B. Lange, M. Oshima, *Programming and Deploying Java(tm) Mobile Agents With Aglets(tm),* Addison-Wesley, Reading, Massachusetts, 1998.

[12] R. Moore, T. A. Prince, and M. Ellisman, Data Intensive Computing and Digital Libraries, Communications of the ACM, Volume 41, pages 56-62, 1998.

[13] See http://www.dmg.org.

[14] J. Ross Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.

[15] A. E. Raftery, D. Madigan, and J. A. Hoeting, 1996. Bayesian Model Averaging for Linear Regression Models, Journal of the American Statistical Association, Volume92, pages 179–191, 1996.

[16] S. Stolfo, A. L. Prodromidis, and P. K. Chan, JAM: Java Agents for Meta-Learning over Distributed Databases, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining,* AAAI Press, Menlo Park, California, 1997.

[17] R. Subramonian and S. Parthasarathy, An Architecture for Distributed Data Mining, to appear.

[18] The Terabyte Challenge, http://www.ncdm.uic.edu/tc

[19] R. L. Grossman and A. Turinsky, Optimal Strategies for Distributed Data Mining using Data and Model Partitions, submitted for publication.

[20] D. Wolpert, Stacked Generalization, Neural Networks Volume 5, pages 241-259, 1992.

[21] L. Xu, and M.I. Jordan, EM Learning on A Generalised Finite Mixture Model for Combining Multiple Classifiers, in Proceedings of World Congress on Neural Networks, Hillsdale, New Jersey, Erlbaum, 1993.