Scalable, Distributed Data Mining—An Agent Architecture

Hillol Kargupta and **Ilker Hamzaoglu** and **Brian Stafford**

Computational Science Methods Group, X Division, Los Alamos National Laboratory
P.O. Box 1663, MS F645
Los Alamos, NM, 87545
hillol@lanl.gov, i-hamzu@uiuc.edu, stafford@lanl.gov

Abstract

Scalability determines the potential in distributing both data and computation in data mining. The PADMA (PArallel Data Mining Agents) architecture will be described, along with experiments on text to address scalability. PADMA agents offer parallel data access, and hierarchical clustering, with results visualized through a JAVA web–interface.

Introduction

[1] Data mining involves extracting, analyzing, and presenting data in concise and useful forms. Data sources are tending to be distributed remotely from users. By offering remote access and analysis through multiple agents, a wider variety of sources can be accessed, and more computing power can be applied. When agents are located with data sources, a concise response will minimize network usage. This is a key to scaling as networks tend to be the biggest bottlenecks.

In PADMA, agents access local data to produce analyses orders of magnitude smaller than the original data. These results can be economically communicated for collaborative data analysis, and web based interactive visualization. This paper describes the initial implementation of PADMA for unstructured text data mining. Though PADMA is not domain specific, and is currently expanding for structured and numeric data.

The next section covers related works. An architecture section focuses on distributing data and work among agents. A section describes text analysis in agents. Web interface descriptions are followed by scalability experiments and conclusions.

Related Work

Two threads support the many domains influencing PADMA: (1) agent based information processing, and (2) parallel computing. This section briefly reviews these fields in the area of data mining.

An introduction to intelligent agents can be found in Maes (1994, Foner (1993). Demand for adaptive systems lead to agents in automated mail filtering (Maes, 1994), and meeting scheduling (Kozierok & Maes, 1993). The *Amalthaea* system (Moukas, 1996) uses agents to discover and filter the world-wide-web. User feedback on agent efficacy drives evolutionary learning algorithms which generate new agents. McElligott and Sorensen (1994) proposed an evolutionary, connectionist approach for filtering, using machine learning to find document representations under user supervision.

Parallel data mining of large databases is growing. Holsheimer et al. (1996) developed *Data Surveyor*, a mining tool and parallel database server with high level rule induction. Zaki et al. (1996) focuses on optimization of parallel rule induction. The *PARKA* project (Anderson et al., 1994) demonstrates large scale parallel mining. The *Conquest* system (Shek et al., 1996) mines distributed geoscientific data, exploiting parallel queries. *GA-MINER* (Radcliffe, 1995) determines data representations, then detects patterns with a parallel genetic algorithm. *GA-MINER* scalability was studied for shared and distributed memory machines.

PADMA Architecture

Figure 1 shows PADMA's overall architecture. The main components are, (1) data mining agents, (2) agent coordinating facilitator, and (3) user interface.

Data mining agents extract and share high level information. The initial data mining agents specialize in clustering (grouping) text, reporting members and the most representative words for each cluster.

The facilitator gives user requests and feedback to agents, and merges concise, high level results. Requests are in Structured Query Language. Feedback on representative word quality adds training to the initially unsupervised analysis. Results allow a graphic view, and a list of representative words for each cluster.

The graphical user interface is web-based, allowing remote interaction. An HTML/CGI/JAVA mix is moving to JAVA over socket communications.

Agents and facilitators use the Parallel Portable File System (PPFS), developed by the University of Illinois at Urbana-Champaign (Huber, Elford, Reed, Chien, & Blumenthal, 1995). MPI (Message Passing Interface) provides interprocess communication. PPFS and MPI form an extensible, object-oriented infrastructure in C++. PADMA currently runs on a Sun Sparc workstation cluster and on an IBM SP-2, but can port to

[1]Copyright ©1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

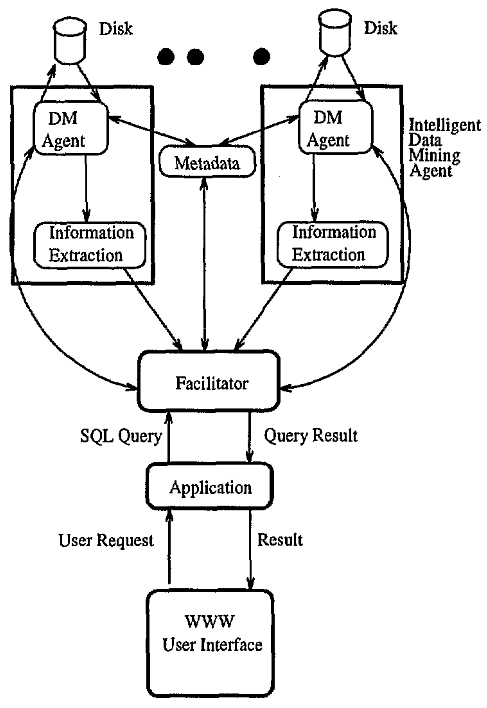

Figure 1: The PADMA architecture.

distributed machines with MPI and Unix file I/O. Parallel access, and analysis by agents is covered next.

Parallel Data Access

On a large scale, access time is a critical limiting factor. Parallel access may help decrease response times (Dewitt & Gray, 1992). Also, Inter agent communication is slower than memory access and should be limited.

Each agent has its own disk for local operations. This provides parallel data access. Striped and blocked algorithms are used to distribute documents among agents. The agents and the facilitator also maintain file caches. Cache management, e.g. FIFO replacement, write-back and prefetching, maximize cache benefits.

Agents in PADMA also provide parallel relational database functionality. Each corpus is stored as a relational database table with *document number, text, ngram vector* attributes. A subset of SQL is supported, including creation/deletion of tables and hash indices, and parallel select and join. PADMA uses intra-operator parallelism.

SQL queries allow a focus on special conditions. The $NGRAM = ELECTRON$ condition selects data sets with the feature $NGRAM$ instantiated to $ELECTRON$. In a query and analyze (cluster), a table subset is selected and reanalyzed—without creating a new table. Agents locally select and analyze data, only needing to communicate concise results to the facilitator. This scalable parallel select was used in experiments.

There are three major algorithms for table joins (Dewitt, 1991; Schneider, 1989). Nested-join cross compare tables with complexity $O(n^2)$. Sort-merge join reduces complexity to $O(n \log n)$ by sorting both tables on join attributes, then comparing by binary search. Hash-join can be fastest, but is ineffective for non-equijoin operations. PADMA uses sort-merge join.

A fragment and replicate strategy parallelizes the sort-merge join. Each agent joins a pair of fragments, then broadcasts to the other agents. After inter-agent comparisons, the facilitator merges results.

Parallel Data Analysis

In PADMA, agents primarily analyze local data and communicate a small or null "concept graph" to the facilitator. The facilitator combines concept graphs and sends results to the user interface. Scalability is achieved by minimizing communication and is best appreciated in the context of the text analysis modules.

The first type of analysis in PADMA's data mining agents is clustering. Clustering groups items based on similarity. A feature vector is formed to from the values of features extracted from items in a data base. Similarity is then measured geometrically, with closer vectors being more similar. Mathematics has an abstract definition of 'measure', which supports many ways to measure. For text, Euclidean distance is too sensitive to document size, but the angle between two feature vectors tells if two items have features in similar proportions. PADMA uses the *cosine* measure taken from a dot product of feature vectors, with a value of the cosine of the angle between feature vectors.

For scalability, data is partitioned into blocks, and each block is partially clustered. Partial clustering only considers a small portion of the greatest similarities in each pass, but reduces a block to a smaller number of items and clusters. In subsequent passes, blocks will combine in a pyramid hierarchy. Each agent can locally construct a hierarchy, and need only communicate a small amount of information (a concept graph) from the top of its hierarchy. For a large number of agents, facilitators may continue the hierarchy to reduce the information sent to the user interface.

As clusters grow, statistics can be used to characterize the features that are most important to similarity within a cluster. In the case of text, a set of representative words can be extracted. The clustering can start without prior knowledge of a set of texts, but can learn a set of most important words. The user may then supervise by giving feedback on the quality of the words judged important to clusters. Such training improves the systems judgment for future clustering, an can be applied to new data sets in the same domain.

Scalability is achieved by localizing data access, and limiting communication to small, distilled analyses. Other analytical methods may be added as long as they can extract information in a hierarchy. The user interface has the core capability of displaying hierarchies of information, and can be specialized to display results of future analytical tools.

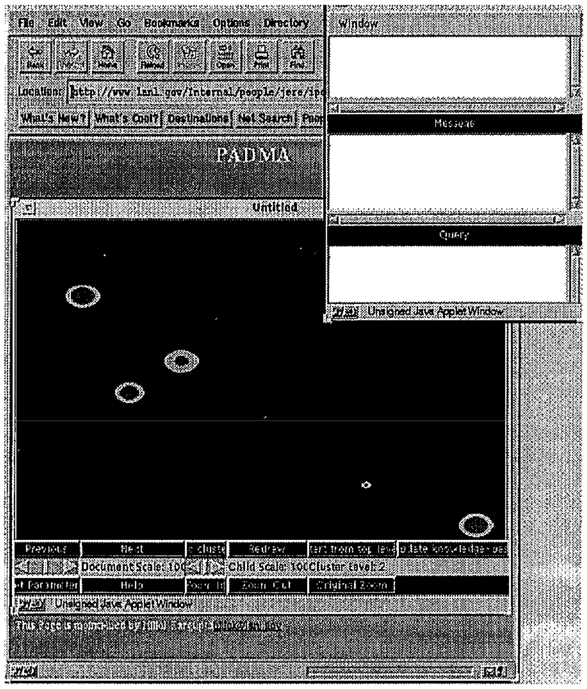
File Edit View Go Bookmarks Options Directory
Window
Location: http://www.lanl.gov/Internal/people/jere/
PADMA
Untitled
Message
Query
Previous
Next
Redraw
Document Scale
Child Scale
Cluster level: 2
Help
Zoom Out
Original Zoom
Unsigned Java Applet Window

Figure 2: Web based interface of PADMA.

Web Interface For Visualization

The interface is built on HTML and JAVA frames. Currently, JAVA controls remote C++ code through a CGI script, though we are migrating to socket communication. The initial parallel version described in this paper has five major operations: (1) create, (2) delete, (3) read, (4) query, and (5) cluster.

Clustering and Querying (with automatic clustering on the query results), provide text analysis. The area of a light blue outer circle reflects the number of documents in a cluster, the inner red dot shows how many children are connected from the next lower level. The user may move up or down in the hierarchy a level at a time, or move down to see only the children of a selected cluster. Representative words can be seen for any selected cluster. On the bottom level, the user may retrieve the documents in a cluster.

Experiments

As an initial performance study, we performed three different experiments to assess the performance and scalability of the PADMA system. We measured the execution times for clustering all the documents in a corpus as well as clustering a subset of the documents related through a select or join operation. Throughout the experiments PADMA agents and the facilitator are configured to use 2MB write-back caches. In all the experiments we used the TIPSTER text corpus of size 36MB containing 25273 text documents. It's striped across all agents with a striping factor of 1.

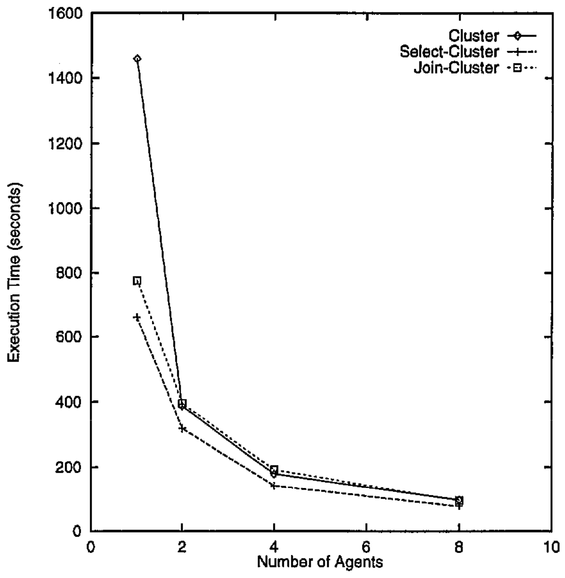

Figure 3: PADMA Performance

The experiments are carried out on the 128 node IBM SP2 at Argonne National Laboratory. On this machine, 120 nodes are used as compute nodes and the remaining 8 nodes are used as dedicated I/O servers. Each compute node has its own I/O subsystem which uses its own local disk, and the I/O servers have faster I/O subsystems. On this machine, all PADMA components run on the compute nodes. PADMA data mining agents use the input/output subsystem of the nodes they are executing on for storing and retrieving the documents. The IBM SP2 was in multi-user mode during the experiments.

The experimental results are presented in Figure 3. The graph corresponding to Cluster, shows the time it takes for clustering the whole corpus. Since each agent clusters its portion of the documents independently, there is no interprocess communication involved in clustering except sending the clustering results to the facilitator. As a result of this we got a linear speedup for the clustering algorithm which demonstrates its scalability. We even got a super-linear speedup when the number of agents is increased from one to two possibly due to memory effects.

Figure 3 also shows the graph corresponding to Select Cluster, which refers to the time it takes to apply a select query to a corpus and cluster the resulting documents. As we mentioned earlier, this combined operation helps the user to focus on the documents he wants to explore rather than considering all the documents in the whole corpus. In this experiment we used the following SQL query,

SELECT DOCNO,TEXT,NGRAM FROM TIPSTER WHERE NGRAM = ELECTRON where the *NGRAM = ELECTRON* condition refers to selecting the documents that are related with keyword *electron*. 15084 documents in the TIPSTER corpus matched this select query. Then these documents are clustered using the regular clustering algorithm. This process in done on the fly, i.e. on each agent as soon as matching documents are found they fed into the clustering module.

The graph corresponding to Join Cluster refers to the time it takes to apply a join query and cluster the resulting documents. In this experiment we used the following SQL query,

SELECT TIPSTER.DOCNO, TIPSTER.TEXT, TIPSTER.NGRAM, AUTHORS.CITY FROM TIPSTER,
AUTHORS WHERE TIPSTER.AUTHOR = AUTHORS.AUTHOR AND AUTHORS.CITY = LONDON AND TIPSTER.NGRAM = ELECTRON

where *TIPSTER.AUTHOR = AUTHORS.AUTHOR AND AUTHORS.CITY = LONDON AND TIPSTER.NGRAM = ELECTRON* condition refers to selecting the documents that are written by authors from London and related with keyword *electron*. Since we store each corpus as a separate relational database table, for this experiment we were able to add an AUTHOR attribute to the TIPSTER table which stores the names of the authors of the documents. In addition we used an AUTHORS table that has AUTHOR and CITY attributes. This table consists of 28 tuples. It's also striped across all agents with a striping factor of 1. 15084 documents matched this join query. Then these documents are clustered using the regular clustering algorithm. This process in done on the fly, i.e. on each agent as soon as matching documents are found they fed into the clustering module.

In this section we presented the initial experimental results about the performance of the PADMA system. These results demonstrated its scalability. In these experiments we used a single corpus of size 36 MB and two different select and join queries. In addition we only used striped data distribution and write-back caches, and we didn't perform any prefetching.

Conclusions And Future Work

This paper introduced PADMA, an agent based architecture for parallel data mining. The PADMA system demonstrated that agent based data mining tools are suitable for exploiting benefits of parallel computing. Main characteristics of PADMA are: (1) parallel query processing & data accessing, (2) parallel data analysis (3) interactive data/cluster visualization. PADMA is still under development. A module for supervised learning of piece-wise linear classifiers using feedback from the user is already developed and incorporated in PADMA. We are currently in the process of adding numeric data handling capabilities. A parallel/distributed genetic search engine will be added to support future machine learning algorithms.

Acknowledgments

National Center for Supercomputing Applications and United States Department of Energy funding. IBM SP2 time, granted by Argonne National Laboratory.

References

Anderson, W., Hendler, J., Evett, M., & Kettler, B. (1994). *Massively parallel matching of knowledge structures*. USA: AAAI/The MIT Press.

Foner, L. N. (1993, May). *What's an agent anyway? - a sociological case study*. ftp://medialab.media.mit.edu/pub/Foner/Papers/What's-an-Agent-Anyway–Julia.ps.

Holsheimer, M., Kersten, M., & Siebes, P. (1996). Data surveyor: Searching for nuggets in parallel. *Advances in Knowledge Discovery and Data Mining*.

Huber, J., Elford, C., Reed, D., Chien, A., & Blumenthal, D. (1995). *PPFS: A high performance portable parallel file system* (Technical Report UIUCDCS-R-95-1903). CS Dept., UIUC.

Kozierok, R., & Maes, P. (1993). A learning interface agent for scheduling meetings. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces* (pp. 81–88). ACM Press, New York.

Maes, P. (1994, July). Agents the reduce work and information overload. *Communications of the ACM* (Vol. 37, No. 7).

McElligott, M., & Sorensen, H. (1994). An evolutionary connectionist approach to personal information filtering. In *INNC 94 (Fourth Irish Neural Network Conference)* (pp. 141–146). ftp://odyssey.ucc.ie/pub/filtering/INNC94.ps.

Moukas, A. (1996). *Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem*. MIT Media Laboratory.

Radcliffe, N. (1995). *Ga-miner: Parallel data mining with hierarchical genetic algorithms final report* (Technical Report EPCC-AIKMS-GA-MINER-REPORT 1.0). Quadstone Ltd.

Shek, E., Mesrobian, E., & Muntz, R. (1996). On heterogeneous distributed geoscientific query processing. In *Proceedings of 6th International Workshop on Research Issues in Data Engineering: Interoperability of Nontraditional Database Systems* (pp. 107–116).

Zaki, M., Ogihara, M., Parthasarathy, S., & Li, W. (1996). *Parallel data mining for association rules on shared-memory multi-processors* (Technical Report 618). CS Dept., University of Rochester.