Scalable, Distributed Data Mining Using An Agent Based Architecture

Hillol Kargupta, Ilker Hamzaoglu, Brian Stafford

Computational Science Methods Group
X Division, Los Alamos National Laboratory
P.O. Box 1663, MS F645
Los Alamos, NM, 87545

Office: (505) 667-8945
Fax: (505) 665-4479
e-mail: hillol@lanl.gov

Abstract: Algorithm scalability and the distributed nature of both data and computation deserve serious attention in the context of data mining. This paper presents PADMA (PArallel Data Mining Agents), a parallel agent based system, that makes an effort to address these issues. PADMA contains modules for (1) parallel data accessing operations, (2) parallel hierarchical clustering, and (3) web-based data visualization. This paper describes the general architecture of PADMA and experimental results.

Scalable, Distributed Data Mining Using An Agent Based Architecture

Hillol Kargupta, Ilker Hamzaoglu, Brian Stafford

Computational Science Methods Group
X Division, Los Alamos National Laboratory
P.O. Box 1663, MS F645
Los Alamos, NM, 87545

LAUR-96-3491, shorter version published in the Proceedings of High Performance Computing'97 & Knowledge Discovery and Data Mining'97

Abstract

Algorithm scalability and the distributed nature of both data and computation deserve serious attention in the context of data mining. This paper presents PADMA (PArallel Data Mining Agents), a parallel agent based system, that makes an effort to address these issues. PADMA contains modules for (1) parallel data accessing operations, (2) parallel hierarchical clustering, and (3) web-based data visualization. This paper describes the general architecture of PADMA and experimental results.

1 Introduction

Data mining involves extraction, transformation, and presentation of data in useful form. As we move more and more toward a paper-less society, each of these components of data mining is likely to face the challenges of dealing with large volume of data. Apart from the sheer volume of the data, the very distributed nature of the data storage and computing environments is likely to play an important role in the design of next generation of data mining systems.

In this paper we explore the possibility of large scale data mining using a very distributed information processing architecture. We present PADMA (PArallel Data Mining Agents), an agent based parallel data mining system. In PADMA individual agents are responsible for local data accessing, collaborative data analysis, and web based interactive information visualization. Although PADMA architecture is not specific to any particular domain and currently PADMA is being enhanced to handle both text and numeric data, in this paper we describe only the initial implementation of PADMA for unstructured text data mining.

Section 2 introduces previous work on agent based software systems and parallel data mining. Section 3 presents a general overview of the PADMA system. The parallel relational database accessing operations of PADMA agents are described in Section 4. Section 5 describes the representation scheme of text documents and the hierarchical clustering algorithm incorporated in the agents. Section 6 describes the web-based user interface and visualization module of PADMA. Section 7 presents experimental results on grounds of scalability and summarizes some applications of PADMA in health informatics. Section 8 concludes this paper and identifies the on-going work.

2 Related Work

Although the motivation behind the initial development of PADMA came from many different domains, the main methodological approach was based on two growing fields of computing: (1) agent based information processing architecture and (2) parallel computing. In this section we briefly review previous efforts made in the area of data mining using the above mentioned technologies.

Interest in agent based software systems soared high during the last few years. An introduction to intelligent agents can be found elsewhere (Maes, 1994; Foner, 1993). The demand for adaptive and smarter software system lead to the incorporation of intelligent agent based technology for addressing many different problems, such as automated mail filtering (Maes, 1994; Lashkari, Metral, & Maes, 1994), meeting scheduling (Kozierok & Maes, 1993). Software agents are also used for aiding information retrieval and processing to extract higher level information. Moukas (1996) reported the *Amalthaea* system that uses agents to discover and filter information available in the world-wide-web. *Amalthaea* also used evolutionary learning algorithms for generating new agents. The efficacy of agents was evaluated by the feedback from the user. McElligott and Sorensen (1994) proposed an evolutionary, connectionist approach for information filtering. Their approach suggested using machine learning algorithms to learn suitable representation of the text documents and used feedback from the user for supervised learning.

Parallel data mining is a growing field that tries to exploit the benefits of parallel computing for mining large scale databases. Holsheimer, Kersten, and Siebes (1996) developed a parallel data mining tool, Data Surveyor, that consists of a mining tool and a parallel database server. It supported regular parallel database operations and mechanisms for higher level rule induction. Parallel algorithms for inducing association rules have been reported elsewhere (Zaki, Ogihara, Parthasarathy, & Li, 1996). They primarily focused on optimization issues for parallel rule induction algorithms. The PARKA project (Anderson, Hendler, Evett, & Kettler, 1994) is another example of exploiting the strengths of parallel computing for processing knowledge bases. Although the knowledge base of PARKA is not exactly same as the usual databases used in data mining, it effectively demonstrated the use of parallel computing technology for processing large amount of information. Shek, Mesrobian, and Muntz (1996) reported the Conquest system for parallel data mining of distributed geoscientific data. This system exploits parallel query processing, distributed data accessing capabilities for geoscientific data mining. A genetic algorithm based parallel data mining system, called GA-MINER is reported elsewhere (Radcliffe, 1995). This system first determines a suitable representation of data and then uses a parallel genetic algorithm to detect patterns in the data. The scalability of the system was investigated for shared and distributed memory machines.

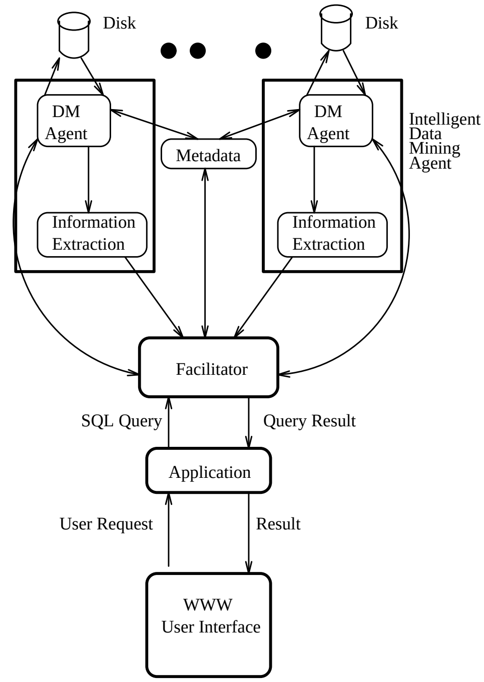

Figure 1: The PADMA architecture.

PADMA combines many features of the agent based and parallel data mining systems. The following section presents an overview of PADMA.

3 Architecture Of PADMA

The PADMA is an agent based architecture for parallel/distributed data mining. The goal of this effort is to develop a flexible system that will exploit data mining agents in parallel, for the particular application in hand. Although PADMA is not specialized for any particular kind of data mining domain, its initial implementation used agents specializing in unstructured text document classification. Figure 1 shows the overall architecture of PADMA. The main structural components of PADMA are, (1) data mining agents, (2) facilitator for coordinating the agents, and (3) user interface. Each of these items are described in the following.

Data mining agents are responsible for accessing data and extracting higher level useful information from the data. A data mining agent specializes in performing some activity in the domain of interest. In the current implementation, data mining agents specializes on text analysis and classification.

Agents work in parallel and share their information through the *facilitator*. The facilitator module coordinates the agents, presents information to the user interface, and provides feedbacks to the agents from the user.

PADMA has a graphical web-based user interface for presenting information extracted by the agents to the user. The facilitator accepts queries from the user interface in standard SQL (Structured Query Language) format; the queries are broadcasted to the agents. Agents comes up with the extracted information relevant to the query. Facilitator collects the information and presents it to the user.

The agents and facilitator of PADMA are developed using a Parallel Portable File System (PPFS). Parallel Portable File System (PPFS) user-level library was developed in the Computer Science department in University of Illinois at Urbana-Champaign (Huber, Elford, Reed, Chien, & Blumenthal, 1995), (Huber, 1995). The PADMA is designed in object-oriented style to provide an extensible infrastructure and coded in C++. MPI (Message Passing Interface) is used as the message passing substrate for interprocess communication. Each data mining agent uses the underlying unix file system on the machines they are executing on for carrying out their local input/output operations. PADMA currently runs on a cluster of Sun Sparc workstations and on IBM SP-2. However it is easily portable to any distributed memory machine provided that MPI is operational on this machine and a unix file system is used for serial input/output operations on its nodes. The user interface is written for Java sensitive browser. PADMA can be functionally decomposed into three different components: (1) parallel query processing and data accessing, (2) hierarchical clustering, and (3) interactive cluster/data visualization. Each of these components of PADMA will be further elaborated in the following sections.

4 Parallel Data Accessing Operations By Agents

Accessing data is an important aspect of data mining. In large scale data mining data access input/output performance becomes a critical factor in the overall performance of the data mining system. Accessing data in parallel may help decreasing the response time (Dewitt & Gray, 1992).

In PADMA, each data mining agent maintains its own disk subsystem to carry out input/output operations locally. This provides parallel data access for the whole system. Currently striped and blocked data distribution algorithms are used to distributed documents across data mining agents. Each agent and the facilitator also maintain a file cache for caching the documents that they access. Appropriate buffer management algorithms, e.g. FIFO replacement policy, write-back and prefetching, are employed to maximize the benefit obtained from these caches.

Data mining agents in PADMA also provide parallel relational database functionality. This is achieved by storing each corpus, which consists of a number of text documents, as a relational database table with *document number, text, ngram vector* attributes. Currently a subset of SQL (Structured Query Language) is supported by PADMA. These include table creation and deletion, hash index creation and deletion, parallel select and join operations. PADMA achieves parallel query processing through intra-operator parallelism.

This functionality is provided to help the users to select the subset of the documents they want to explore with clustering. PADMA provides a special condition in an SQL query which helps the users to select the documents related to a keyword. For example, $NGRAM = ELECTRON$ condition can be used to select the data sets with the *NGRAM* feature instantiated to the keyword *ELECTRON*. Using this special condition users can analyze data related to a keyword in two different ways. In the first method, PADMA can be used to create a new table based on the outcome of a query and then this new table can be analyzed by PADMA agents. The second method achieves the same functionality on the fly, i.e. without creating a new table. This is done by the query and cluster (we refer to the analysis part by *cluster operation*, since the initial implementation of PADMA had only unsupervised cluster analysis capability) operation which combines the querying and clustering operations by first executing the query operation on the agents and then feeding the selected data directly to the analysis module. This is a much more scalable algorithm compared to the first one, since it doesn't involve communicating with the facilitator except reporting the final results. We used this method in the performance experiments.

Parallel select operations in PADMA are carried out independently by each data mining agent without any interprocess communication. After each agent is carried out the select operation on its local data, the results are gathered by the facilitator which produces the final outcome of the select operation by merging these individual results.

There are three major algorithms for implementing join operations between two tables (Dewitt, Naughton, & Schneider, 1991), (Schneider & Dewitt, 1989). Nested-join involves comparing each tuple of the first table with all the tuples of the second table, and it's complexity is $O(n^2)$. Sort-merge join reduces the complexity to $O(nlogn)$ by sorting both tables based on the join attribute values. Then these sorted tables are compared using binary search. Hash-join algorithm partitions both tables into a number of buckets based on the join attribute values, and then matching is performed within each bucket independently. This reduces the complexity to $O(n)$. Hash-join algorithm performs better than the sort-merge join for equijoin operations unless the tables are already in sorted order. However it is ineffective for non-equijoin operations. In order to effectively support both equijoin and non-equijoin operations, sort-merge join algorithm is implemented in PADMA.

A fragment and replicate (broadcast) strategy is utilized to parallelize the sort-merge join algorithm. Each data mining agent initially sorts its part of both tables, and compares these parts. Each agent then broadcasts its part of the small size table to the other agents. After each agent compares its part of the larger table against the tuples of the small table it received from the other agents, the results are gathered by the facilitator which produces the final outcome of the join operation by merging these individual results.

5 Parallel Data Analysis By Agents

In PADMA data analysis is primarily done by the agents in a distributed fashion. Every agent returns a "concept graph" to the facilitator which could be null if an agent does not find anything relevant to the user's query. The facilitator is responsible for combining the concept graphs and present the result to the interface in a user transparent manner.

Although PADMA agents are currently being provided with numeric data analysis algorithms, experimental results reported here were produced using agents that are capable of analyzing unstructured textual data. PADMA agents uses both supervised learning and unsupervised hierarchical clustering techniques for generating the concept hierarchy of document clusters.

5.1 Text Data Mining Agents

The objective of text mining agents of PADMA is to identify statistically significant document clusters that may lead to identifying common patterns among the documents in a text corpora. Text mining involves two important steps: (1)

choosing/constructing the document representation and (2) finding of relations among the documents. PADMA uses a hierarchy of different representations. Relations among the documents are determined using both unsupervised hierarchical clustering algorithms and optional user feedback driven piecewise linear classifiers. Experiments reported in this paper did not use any user feedback driven supervised learning. Therefore, in the following sections we focus only on the unsupervised clustering analysis of text data.

5.2 Representation and similarity measure

The unsupervised component of the text mining in PADMA is primarily based on statistical analysis. A hierarchical clustering algorithm is used for generating a concept graph relating documents and clusters to each other. Usually clustering algorithms work from a representation of the underlying state space and a measure of similarity between any two points from the state space. Typical representation of text documents uses a vector of weighted word frequencies (Salton, Allan, Buckley, & Singhal, 1994) in the document. However, word frequency based representations are sometimes susceptible to spelling errors. An alternate representation called *n-gram* was proposed elsewhere (Damenshek, 1995). N-grams are n-letter strings. The set of 1-grams is just the alphabet. The set of 2-grams is the set of pairs of letters. With a 26 letter alphabet, there are 26^n possible n-grams. Spaces may be included to indicate the boundaries of words. N-grams have been successfully demonstrated for approximate text classification (Damenshek, 1995; Kimbrell, 1988; Cavnar, 1993). PADMA uses a two level representation for generating a hierarchical classification of the documents, namely: (1) n-gram representation for documents (2) or-grams for cluster representation. An or-gram of a cluster is simply a representation obtained by arithmetic *or* operation among the corresponding n-gram frequencies. This basically generates a vector of n-grams weighted by their relative frequencies within the set of documents in a cluster. We have used cosine of the angle between any two n-grams or any two or-grams as the similarity measure among documents and clusters respectively. Moreover, regular decision theoretic approaches using first and second order distribution statistics are adopted in order to minimize the decision errors. The following section describes the hierarchical clustering algorithm used in PADMA.

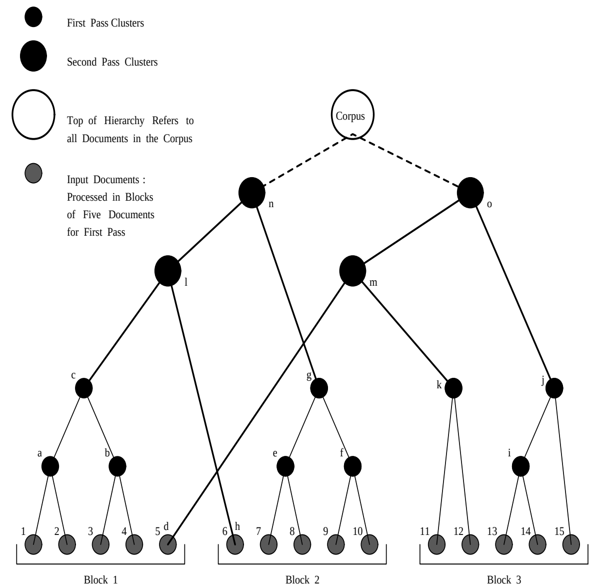

Figure 2: Cluster hierarchy of documents.

5.3 Hierarchical Clustering Over Blocks of Data

Figure 2 illustrates the idea of hierarchical clustering used in PADMA. First consider the algorithm running on one machine. One pass on the first block produces cluster c and document 5. Document 5 is considered as cluster d for future passes. The rest of the blocks are independently processed in a first pass with their results concatenated in a file. The total results of the first pass look like:

$c = \{1,2,3,4\}$, $d = \{5\}$, $h = \{6\}$, $g = \{7,8,9,10\}$, $k = \{11,12\}$, $j = \{13,14,15\}$.

Note that clusters are formed by binary combinations, but only the end clusters of a level are stored. There are three things to notice: intermediate clusters a, b, e, f, i are omitted, n ary clusters are recorded, and outlying documents or clusters may not be agglomerated until latter passes. The last point shows that documents can combine even if they do not start in the same block. For example, document 5 is several blocks away from 11 and 12 but merges higher up in the hierarchy in cluster m. A further detail can be seen in comparing clusters i, j and k. Clusters are formed in order of nearness of members rather than in an ordered scan of the input. Thus i was formed first as 13 and 14 were the closest items in their block. Likewise, j was formed before k because 13, 14, 15 are more similar to each other than were 11 and 12.

The second pass is able to form interrelations among documents from several blocks. If there are a large number of blocks then $O(\log n)$ passes will be needed to assure that all documents get the chance of being interrelated. However, in retrieval, every branch is searched downward until it can be eliminated or given a strong match. So if a document (say 5) can not be related to its ideal companion (say 15) until after it has been clustered with less desirable neighbors, then the or-gram can help. For a query that best matches 5 and 15, the centroid of o will show it has some promise, but its or-gram will say that a minority of its documents satisfy the query, so the search will continue into data produced in the first pass. Centroids for d, k, and j will be checked to show d and j as relevant. Cluster d will show a match with document 5, and j will show either a minority or majority match. In a minority match to j, likely only 15 will be returned. For a majority match to j, j will return all of its members. The rational of the majority return, even if 13 were not close to the query, is that 13 has enough in common with query matching documents 14, 15 that 13 is likely to be related to the query in some interesting way. Highlighting may be used to show either all strong similarities among the results or just the portions of the results which matched with the query.

In the case of multiple Data Mining Agents (DMAs), each agent works independently on its portion of the total documents and proceeds with clustering until some small number of clusters, five for example, are left at the top. Then the diagram can be seen as three Data Mining Agents feeding their end results to a client. The client can then cluster over a highly condensed amount of input. The client then passes its results for display. User queries can be passed from the client to the DMAs, with the results again passed up to the client which can display deeper clusters if there were too many matches, or display a reasonable number of matched documents. If the user clicks to select a particular cluster in a visual display, the centroid of the chosen cluster is sent as a query vector to each of the DMAs. Again, the combination of clustering over blocks and searching down in the resulting hierarchy can overcome deficiencies resulting from data being distributed among separate machines or CPUs.

The only work trade-off is that more extensive searches may be performed if nearest neighbors tend to be far apart in the input. In this case the data might have a cyclical pattern which can be taken to advantage. For example, quarterly data should be grouped by quarters for data that is seasonally adjustable, or arranged with common sources grouped together and sequenced in quarter order for time series related data. Lagged or phase shifted data can likewise be reordered according to the period of the lag or shift. The following section describes the web based user interface of PADMA.

6 Web Based User Interface For Data Visualization

PADMA provides a world wide web based user interface for visual interaction with the system. Users can specify the requested operation type from the PADMA homepage through an HTML form. This page is shown in Figure 3. The interface then communicates with the PADMA system through a cgi script which submits the request to PADMA. After the requested operation is carried out, the result is displayed to the user.

User interface currently supports five major operations. Create option is used to create a table out of unstructured text documents. Users should supply these documents to PADMA. Read option is used to read the contents of a table. Delete

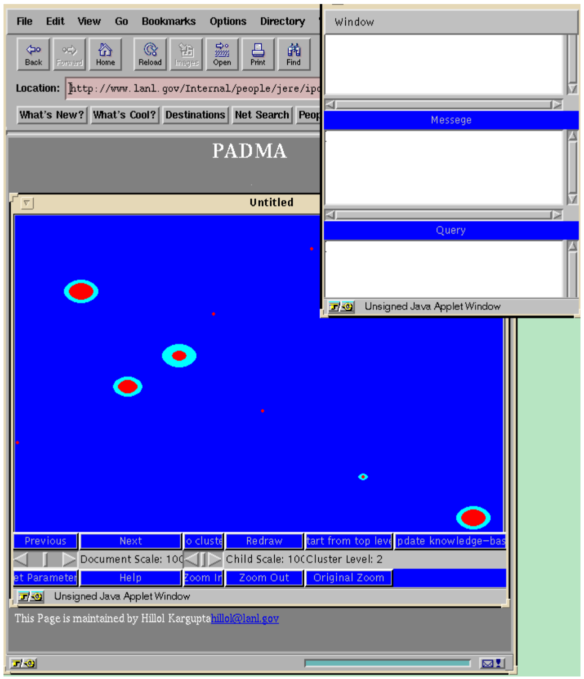

Figure 3: Web based interface of PADMA.

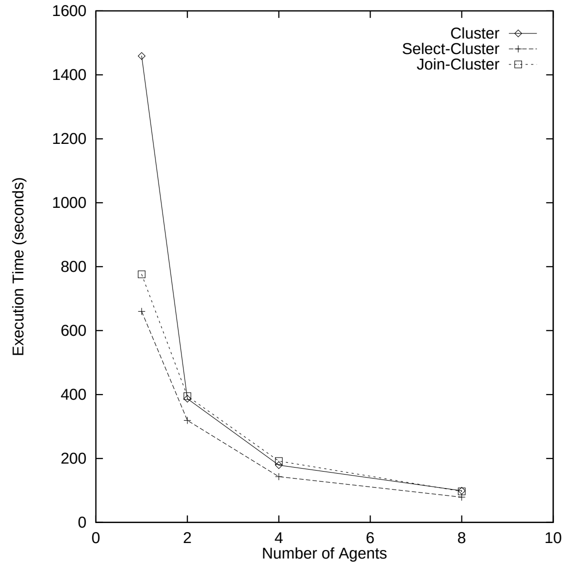

Figure 4: PADMA Performance

option is used to delete a certain table. Query option is used to query these tables. PADMA applies the SQL query submitted by the user to the appropriate tables and presents the result back to the user.

Clustering option is used to cluster all the documents in a single table as well as clustering a subset of the documents related through a select or join operation. In the latter case, the user also provides an SQL query. PADMA first applies this query to the appropriate tables and then clusters the resulting documents. This helps the user to focus on the documents he wants to explore rather than considering all the documents in a single table. The result of a clustering request is presented to the user in the form of a two dimensional cluster plot, which shows the nodes representing clusters at a certain level. The display operation is carried out by a java applet. Therefore a java aware web browser is needed to display the cluster plot. A color encoding scheme is used to represent the the degree of bushiness of each node in the graph of clusters.

PADMA user interface provides interactive iterative clustering. For an initial clustering request the top level of hierarchical clustering outcome is presented to the user. Users then can click on any one of the clusters presented in a cluster plot to further explore the documents in this cluster. Based on this request the clusters in the next level of hierarchical clustering outcome that belong to this top level cluster selected by the user are presented to the user in a new cluster plot. Users can continue examining the deeper levels of this clustering hierarchy by interactively clicking on the cluster they want to explore further. Finally the documents in the related cluster are presented to the user.

7 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.

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 4. 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 superlinear speedup when the number of agents is increased from one to two possibly due to memory effects.

Figure 4 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 *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.

8 Conclusions And Future Work

This paper introduced PADMA, an agent based architecture for parallel data mining. 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. However, PADMA is still under development and requires more work. A module for supervised learning of piece-wise linear classifiers using feedback from the user is already developed and incorporated in

PADMA. Also we are currently in the process of incorporating both numeric and text data handling capabilities in PADMA.

Acknowledgments

This work is supported by a grant from National Center for Supercomputing Applications and United States Department of Energy. We also acknowledge the computing time on IBM SP2, 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.

Cavnar, W. B. (1993). N-gram based text filtering. (pp. 171–179). National Institute of Standards and Technology.

Damenshek, M. (1995). Gauging similarity via n-grams: Language-independent categorization of text. *Science*, 267.

Dewitt, D., & Gray, J. (1992). Parallel database systems: The future of high performance database systems. *Communications of the ACM*, *35*(6), 85–98.

Dewitt, D., Naughton, J., & Schneider, D. (1991). *An evaluation of non-equijoin algorithms* (Technical Report Technical Report CS-TR-91-1011). Department of Computer Science, University of Wisconsin-Madison.

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1995). A density-based algorithm for discovering clusters in large spatial databases with noise.

Foner, L. N. (1993, May). *What's an agent anyway? - a sociological case study.* ftp://media-lab.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*, 447–470.

Huber, J. (1995). *Ppfs: An experimental file system for high performance parallel input/output* (Technical Report MS. Thesis). Department of Computer Science, University of Illinois at Urbana-Champaign.

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). Department of Computer Science, University of Illinois at Urbana-Champaign.

Kimbrell, R. E. (1988). Searching for text? send an n-gram. *Byte* (May), 297–312.

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.

Lashkari, Y., Metral, M., & Maes, P. (1994). *Collaborative interface agents.* Autonomous Agents Group, MIT Media Laboratory.

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.* Autonomous Agent Group, MIT Media Laboratory.

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

Rasmussen, E. (1992). Clustering algorithms. In Frakes, W. B., & Baeza-Yates, R. (Eds.), *Information Retrieval: Data Structures and Algorithms* (Chapter 16, pp. 419–442). Prentice Hall.

Salton, G., Allan, J., Buckley, C., & Singhal, A. (1994). Automatic analysis, theme generation, and summarization of machine-readable texts. *Science*, *264*(3), 1421–1426.

Schneider, D., & Dewitt, D. (1989). *A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment* (Technical Report Technical Report CS-TR-89-836). Department of Computer Science, University of Wisconsin-Madison.

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).

Willett, P. (1987). *Similarity and clustering in chemical information systems.* Letchworth,England: Research Studies Press LTD.

Willett, P. (1988). Recent trends in hierarchical document clustering: A critical review. *Information processing and management*, *24*(5), 577–597.

Winograd, T. (1972). *Understanding natural language.* New York: Academic Press.

Zaki, M., Ogihara, M., Parthasarathy, S., & Li, W. (1996). *Parallel data mining for association rules on shared-memory multiprocessors* (Technical Report Technical Report 618). Department of Computer Science, University of Rochester.