A Data Mining Architecture for Distributed Environments

Mafruz Zaman Ashrafi, David Taniar, and Kate Smith

School of Business Systems, Monash University
PO BOX 63B, Clayton 3800, Australia
{Mafruz.Ashrafi,David.Taniar,Kate.Smith}@infotech.monash.edu.au

Abstract. Data mining offers tools for the discovery of relationship, patterns and knowledge from a massive database in order to guide decisions about future activities. Applications from various domains have adopted this technique to perform data analysis efficiently. Several issues need to be addressed when such techniques apply on data these are bulk at size and geographically distributed at various sites. In this paper we describe system architecture for a scalable and a portable distributed data mining application. The system contains modules for secure distributed communication, database connectivity, organized data management and efficient data analysis for generating a global mining model. Performance evaluation of the system is also carried out and presented.

1 Introduction

The widespread use of computers and the advance in database technology have provided huge amounts of data. The explosive growth of data in databases has generated an urgent need for efficient data mining techniques to discover useful information and knowledge. On the other hand, the emergence of network-based distributing computing such as the private intranet, internet, and wireless networks has created a natural demand for scalable techniques of data mining that can exploit the full benefit of such computing environments.

Distributed Data Mining (DDM) aims to discover knowledge from different data sources geographically distributed on multiple sites and to combine it to build a global data-mining model [3,4,8]. However, several issues emerge when data mining techniques are used on such systems. The distributing computing system has an additional level of complexity compared with centralized or host-based system. It may need to deal with heterogeneous platforms and multiple databases and possibly different schemas, with the design and implementation of scalable and effective protocol for communication among the nodes, and the selective and efficient use of the information that is gathered from several nodes [9].

A fundamental challenge for DDM is to develop mining techniques without having to communicate data unnecessarily. Such functionality is required for reasons of efficiency, accuracy and privacy. In addition, appropriate protocols, languages, and network services are required for mining distributed data to handle the required metadata and mapping.

In this paper, we present a system architecture for developing mining applications for distributed systems. The proposed architecture is not focused on any particular data mining algorithms, since our intention is not to propose new algorithms but to suggest a system infrastructure that makes it possible to plug in any mining algorithm and enable it to participate in a highly distributed real time system. The system is implemented in Java because it supports portable distribute programming on multiple platforms. Java thread, socket and data compression, JDBC techniques were utilized.

2 Related Work

In this section, we provide some background material and related work in this area. Several system including JAM, PADMA, Papyrus, BODHI, Kensington, PaDDMAS, and DMA have been developed/proposed for distributed data mining.

JAM [3] is distributed agent-based data mining system that uses meta-learning technique. It was develops local patterns of fraudulent activity by mining the local databases of several financial institutes. Than final patterns are generated by combining these local patterns. It assumes that each data site consists of a local database, learning agents, meta-learning agents and configuration modules which perform the major task of distributing computing by sending and receiving different requests from different sites.

PADMA [7] is an agent-based architecture for parallel /distributed data mining. It is a document analysis tool that works on a distributed environment based on cooperative agents. It aims to develop a flexible system that exploits data mining parallels. The data-mining agents in PADMA perform several parallel relational operations with the information extracted from the documents. The authors report on a PADMA implementation of unstructured text mining although the architecture is not domain specific.

The Papyrus [4] system is able to mine distributed data sources on a local and wide area cluster and a super cluster scenario. It uses meta-clusters to generate local models, which are exchanged to generate a global model. The originator reports that the system can support the moving of large volumes of mining data. The idea is founded on a theory similar to JAM system. Nevertheless they use a model representation language (PMML) and storage system called Osiris.

The BODHI [8] is a hierarchical agent based distributed learning system. The system was designed to create a communication system and run time environment for Collective Data Mining. It employs local learning techniques to build models at each distributed site and then moves these models to a centralized location. The models are then combed to build a meta-model whose inputs are the outputs of various models.

Kensington [13] Architecture is based on a distributed component environment located on different nodes on a generic network, like the Internet or Intranet. Kensington provides different components such as user oriented components, Application servers and Third level servers. It warps the analysis algorithm as Enterprise Java Bean components. PaDDMAS [8] is a Parallel and Distributed Data

Mining Application Suite, which uses a similar approach as the Kensington but has extended a few other features like, support for third party components, and a XML interface which able to hide component implementation.

The mining of association rules in distributed database has also been examined by David W.C. *et al.* They presented Distributed Mining of Association Rules (DMA) algorithm, which takes advantage of the inherent parallel environment of a distributed database. It uses the local counts of the large item-sets on each processor to decide whether a large item-set is heavy (both locally large in one database partition and globally large in the whole database), and then generates the candidates from the heavy large item-sets.

The proposed system was developed to support data mining in a distributed or parallel environment but has some significant differences from the abovementioned systems or architecture. In contrast with JAM, PADMA, and Papyrus, our model not only generated a global model from the homogeneous database but also from heterogeneous database. We also employ some secure communication techniques that are required in distributed environment. The Kensington and PaDDMAS systems are component-based. In BODHI system local models are gathered into a centralized site from the different remote sites and then they are combined to generate a global model. In our approach every individual site is capable of doing the same task as the centralized site of BODHI. It allows us to overcome the single point of failure. Moreover, we designed a repository for each site, which allows each site to do further analysis if needed. In contrast with DMA, in our system we analyze the association rule not only with support and confidence but we also consider the total number of record.

3 Design Rationale

The architecture of a data mining system plays a significant role in the efficiency with which data is mined. A typical DDM involves two tasks: local data compression and/or analysis for the minimization of network traffic, and the generation of global data models and analysis by combining various local data and models [12]. To perform these tasks successfully, a DDM system depends on various factors such as data source, security, multiple results etc. In the following paragraphs we evaluate our proposed architecture of distributed data mining on the basis of these factors:

3.1 Data Sources

The distributed data mining applications must run on multiple architectures and different operating systems (for example Windows, Unix). To achieve this, we use Java programming language and hence eliminate incompatibilities. Another challenge of distributed mining application is to find mining rules from different sources of formatted or unformatted data with diverse semantics. Because there are many kinds of data and databases used in different applications, and one may expect that distributed data mining system should be able to perform efficient data mining on different kinds of data [2]. In our module we used JDBC ODBC technology to handle different sources of RDBMS, which are distributed in different locations.

3.2 Multiple Results

In distributed data mining application, different kinds of knowledge can be elicited from large amounts of data including patterns which can be established by examining different mining algorithms (for example Decision Tree, Association rule, Sequential Patterns) in the same set of data. Because discovering knowledge from large database involves huge amounts of data processing cost, it is important to produce an organized way to devise rules, which can be used in the future. On the other hand, technology is evolving day by day, which makes us to think about the future communication between the distributed applications. Extensible Markup Language (XML) has become the standard for communication between applications. With XML, an application defines markup tags to represent the different elements of data in a text file so it can be read and handled by any application that uses XML. In this module, the data mining rule repository stores rules in XML format to achieve the abovementioned goal.

3.3 Technological Issues

As mentioned earlier we used Java technology for eliminating incompatibilities. Java allows us to achieve this by using several techniques: such as RMI and socket. The primary goal is for the RMI to enable programmers to develop distributed Java programs with the same syntax and semantics used for non-distributed programs. To do this, RMI allows Java classes and objects in different machines to communicate and work in a single Java Virtual Machine (JVM). As a result, Java RMI has some communication and implementation overheads. Java Socket level programming (a socket is a software endpoint that establishes bi-directional communication between a server program and one or more client programs) allows us start the server program with a specific hardware port on the machine where it runs so any client program anywhere in the network can be communicated with the server program. As a result Java Socket have less communication and implementation overheads.

3.4 Security

The security of network system is becoming increasingly important as more and more sensitive information is stored and manipulated online [11]. Distributed applications, which are guaranteed to be ‘network friendly’, pose a larger threat than usual. Whenever a request comes from outside the local environment, it poses a threat to security and privacy. Consequently special care must be taken to handle those kinds of attack. The system should support authentication and message security. In the proposed module we use one of the primitive approaches to resolve the authentication problem. And message level security implementation can be obtained by using the Java Secure Socket Extension API.

3.5 Cost Effectiveness

The volumes of data in databases are increasing day-by-day. Large-scale data sets are usually physically distributed. Current methods can handle data in the tens-of-gigabytes range. Association rule mining algorithms do not appear to be suitable for the terabyte range [10]. On the other hand, the Distributed Data Mining Application involves transferring huge amounts data through the networks. This requires implementing some kinds of compression technology. In our module we use Java ZIP compression technology for reducing the data traffic cost.

4 Distributed Data Mining Architecture

Our proposed mining architecture is a client/server-based system developed for performing knowledge discovery from large distributed sources of data. Due to the diversity of mining algorithms and the diversity of data sources, it is difficult to generate a mining model by combining mining rules on different sites. Our proposed system works independently to combine result from different sites. This section describes the abstract architecture model of the Distributed Data Mining and the interaction between its various subsystems. The architecture has the following subsystems: communication, mining, analyzing, and database. Figure 1 shows the architecture and the relationship between the different subsystems.

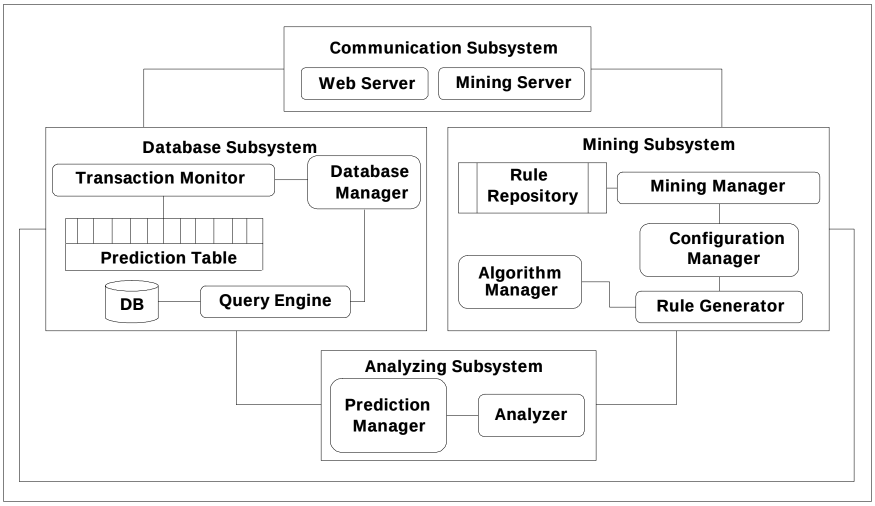

Fig. 1. Distributed Data Mining Architecture

4.1 Communication Subsystem

The communication subsystem is the heart of the network communication. It is responsible for sending and receiving requests to or from other sites registered with the local site. Because distributed systems are vulnerable, special care has been taken on this subsystem to handle unauthorized access. The following steps reduce vulnerability:

- Every time on the outside mining application wants to join with the local mining system, an object is generated which holds various information of that site and places that object in the active site table.
- Whenever any request arrives from that site a new object will be created and verify the active site table.

Sending mining rules to other sites is simple. It sends mining rules to those sites, which can be found on the active site table. Figure 2 shows the class diagram of the communication subsystem. The *MineServer* is an interface, which defines a certain set of functionality of the mining server. The *MineServerImpl* class implements the *MineServer* interfaces. It provides a coordinating facility between the other subsystems. The class *MineServerImpl* uses the Java thread model to concurrently handle multiple requests.

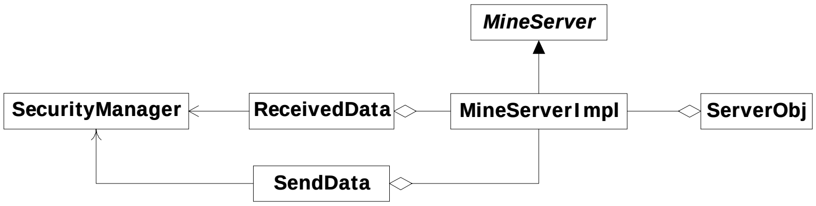

Fig. 2. Class Diagram of Communication Subsystem

The *Server* class is responsible for implements server sockets on a specified port. A socket is a software endpoint that establishes bi-directional communication between a server program and one or more client programs. The socket associates the server program with a specific hardware port on the machine on which it runs so that any client program anywhere in the network with a socket associated with that same port could be communicated with the server program. This class waits for requests to come in over the network. When it gets request from the authorized site, it opens the input stream for reading and saves it in a local file. The Server class reads the input stream as a ZIP file format. This class maintains a log file for management purposes. The *SendData* class is responsible for sending mining rules to the other sites. It sends data as a ZIP file format. The *ServerObj* class is responsible for registering new servers (that is wants to join with the data mining process) with the local sites. The *SecurityManager* class is responsible for verifying different security aspects.

4.2 Mining Subsystem

Figure 3 shows the class diagram of the Mining Subsystem. This is the core subsystem of the proposed distributed data mining system. It basically deals with the various data mining algorithms and manages the existing rules, in an organized way, into the repository.

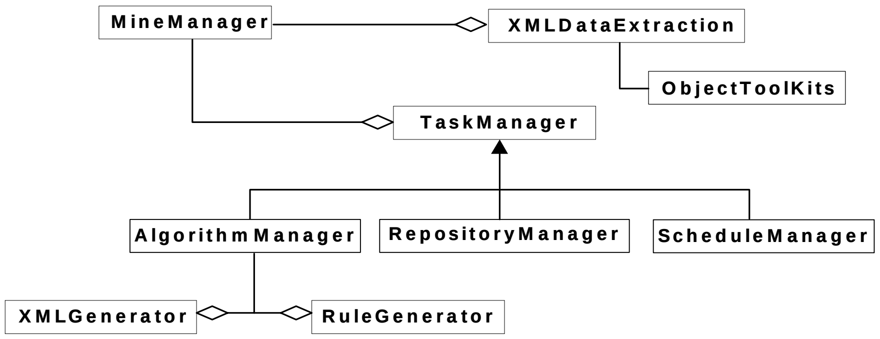

Fig. 3. Class Diagram of Mining Subsystem

The *MineManager* class is responsible for data processing and initiating different tasks. The *XMLDataExtraction* class deals with various incoming XML encoded data (received from the other sites), extracts them and saves them into a repository. The *XMLGenerator* class is responsible for encoding mining rules into the XML format. To define the legal building blocks of an XML document, we use a Document Type Definition (DTD). The DTD specification is actually part of the XML specification, which allows us to create a mining rule in a valid XML structure with a list of legal elements. This can be used it to ensure that the XML encoded data we read from the other site is indeed valid.

The *RuleGenerator* class is responsible for generating rules by using a specific mining algorithm on a particular data set. The *AlgorithmManager* class is responsible for implementing different mining algorithms that are part of the distributed data mining system. It generates rules based on those algorithms. The *RepositoryManager* class is responsible for maintaining the existing rules in an organized way. The *ScheduleManager* is responsible for performing different tasks on a routine basis.

4.3 Analyzing Subsystem

A successful DDM project involves several tasks including, examining and pruning the mining results and reporting the final result. Data mining results include classification, association, clustering, prediction, estimation, and deviation analysis. This subsystem is responsible for analyzing different data mining pattern gathered from multiple sites. It also generates a global model. Figure 4 shows the class diagram of this subsystem.

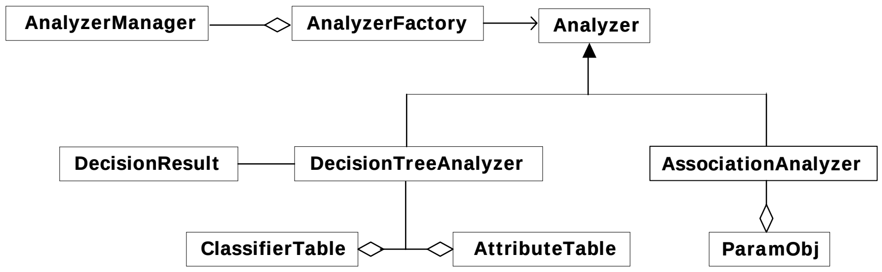

Fig. 4. Class Diagram of Analyzing Subsystem

The *AnalyzerManager* class initiates the global data-mining model generation task. Since the generation of global mining depends on various mining rules, we implemented a different rule analyzing class to achieve that. The *AnalyzerFactory* class returns an instance of a class depending on the data provided by *AnalyzerManager* class.

In this project we implemented two rules analyzed for two algorithms, the Rule Induction (Decision Tree) and the Association Mining. The former is a model that is both a predictive and a descriptive representation of a collection of rules. Rule induction is one of the most descriptive forms of knowledge discovery. It is a technique for discovering a set of "If / Then" rules from data in order to classify the different cases. Because it looks for all possible interesting patterns in a data set, the technique is powerful.

In the *DecisionTree* class we combined decision tree mining rules, each which has a classifier and a set of attributes. The classifier indicates the label or category to which the particular rule belongs. Attributes can be continuous that is, coming from an ordered domain, or categorical that is, coming from an unordered domain. We divided each rule on two parts, the classifier and the rule and represented them into two tables. The classifier table holds the classifier name and the corresponding rule number. The rule part is further divided into the attribute level and put into two different tables, the root and child, with the attribute name and rule number.

In a distributed computing environment, the database may fragment in different sites, as a result, can generate an overwhelming number of rules from several sites. To handle this kind of scenario we closely observed whether the attributes (root as well as child) of one rule fully or partly belongs to other rules or not and eliminated the fragmented rules. The rules in the rule induction form are independent and many may contradict each other. If we found any contradiction rule, we marked that rule as clash between the corresponding classifier. Human interaction is required to overcome such scenarios.

The association rule is used to find the set of all subsets of items or attributes that frequently occur in many database records or transactions, and additionally, to extract rules about how a subset of items influences the presence of another subset. The two important measures in the association rule are support and confidence.

The *AssociationAnalyzer* class analyzes different association mining rules received from the multiple sites and generates the global association-mining model. In a traditional (centralized-based) system, association rules are generated on the basis of local support and the confidence of the itemsets. In distributed environment the database may fragment, and the size of the database may vary from one site to another. This requires us to consider some additional parameter for generating a global model. This class generated global association mining model based is on four parameters: support, confidence, total support, and total confidence. The first two parameters provide the percentage of support and confidence of any particular itemset pattern. The parameter total support is measured by numbers of records present in the training set. Total confidence is measured by the numbers of times a particular item set with minimum confidence satisfies a particular pattern on it. In this class we implemented two different methods for generating a global model.

4.4 Database Subsystem

This subsystem is responsible for retrieving data from storage and saving it back to the database. To do this, it maintains a connection with a specific database. It has the capacity to generate a result by using SQL queries and stored-procedures within the context of a particular connection. Figure 5 shows the class diagram of this subsystem.

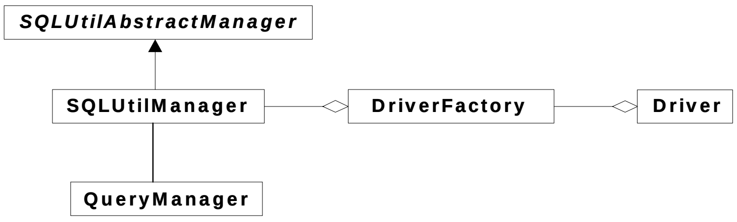

Fig. 5. Class Diagram of Database Subsystem

The *SQLUtilManager* class is the heart of this subsystem. It represents an individual session with the database and provides the methods necessary to save and retrieve objects it. It has the capability to support connections in various well-known databases and the *DriverFactory* class instantiate the corresponding driver for that database. The *QueryManager* class retrieves results from the database. The retrieve operation uses a simple SQL statement or calls a store procedure, which are, resides in the database.

5 Performance Evaluation

In this section we review the preliminary experiments of the proposed DDM architecture. We carried out a sensitivity analysis through a simulation. A sensitivity analysis is performed by varying performance parameters. The parameters were varied with different fragment schema, redundant rules, numbers of base classifier and total number of rules.

The experiments were run on a Windows 2000 server environment. The local rule model was generated from the data replicated and fragmented into three different sites. The local model consists of several thousand descriptive decision tree rules in If/Than format. We conducted this experiment by varying 5500 to 55500 of rules. The rule data contained a total number of 14 attributes. Some of the attributes are numeric, the rest categorical. The average length of each rule is 60 bytes. The experiment compared the total time of generating a global rule model by combining different local rules (that is generated each individual sites).

5.1 Result of Experiment

Figure 6 shows a comparative performance by varying the rules (received from three different sites) with a different base classifier. Each base classifier was equally distributed among the rules. In the first phase, each rule was scanned to identify the classifier of that rule and then to create the corresponding root and attribute table. The data are fragmented (both vertically and horizontally) and in a local rule model same rule may exist in a different format (that is the combination of attribute may appear differently).

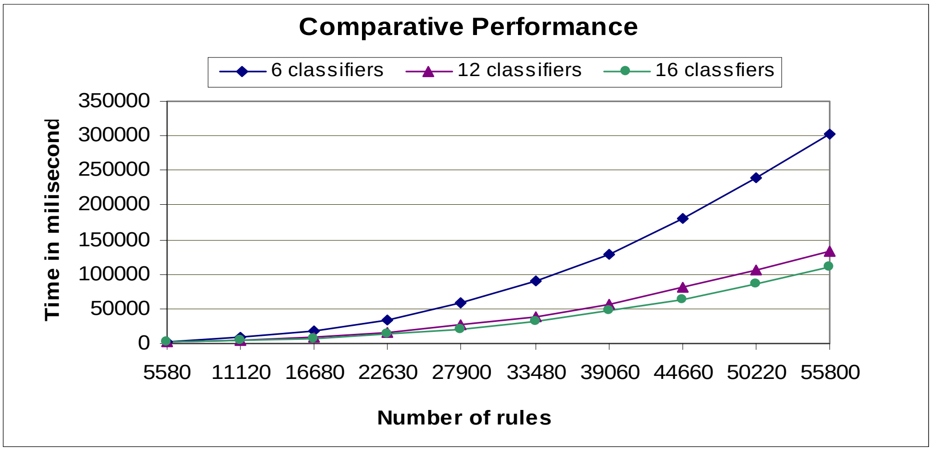

Fig. 6. Comparative Performance

The preliminary results indicate that the global rule model for the classifier set with 6 elements perform extensive data processing because its attribute table size increases with the proportion of rules. The major cost for scanning the data and finding rules with the same attributes. On the other hand, rules with elements of 12 and 16 classifier sets have smaller attribute tables compared with the classifier set of 6 elements. Hence, they scanned less data. On average, the classifier set with 16 elements is nearly two to three times faster then the classifier set with 6 elements.

6 Conclusions and Future Work

The distributed data mining uses communication networks to extract new knowledge from a large data set that exists in a distributed environment. It can enhance the computational time of knowledge extraction. In this paper we have defined and designed a system architecture for Distributing Data Mining, which allows us to combine local and remote patterns and to generate a global model for a different mining algorithm. The architecture is based on Java language. XML technique is used for data translation with support distributing computing. The secure socket layer is designed for communication between different sites. The Java thread is used to achieve parallelism.

Future work is being planned to investigate data security and privacy. It is important to consider when distributed knowledge discovery may lead to an invasion of privacy and what kind of security measures could be developed for preventing the disclosure of sensitive information.

References

1. David W. Cheung, Vincent T. Ng , Ada W. Fu, and Yongjian fu “Efficient Mining of Association rules in Distributed Databases”. *IEEE Transaction on Knowledge and Data Mining,* Vol. 8, No 6, December 1996.
2. Chen M.S., Han J., and Yu P.S. “Data mining: An overview from a database perspective”. *IEEE Transactions on Knowledge and Data Engineering,* Vol 8, No 6, pages 866-883, 1996.
3. Stolfo S.J., Prodromidis A.L., Tselepis S., Lee W., Fan D.W., and Chan P.K. “Jam: Java agents for meta-learning over distributed databases”. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 74-81, Newport Beach, CA, August 1997. AAAI Press.
4. Bailey S.M., Grossman R. L., Sivakumar H., and Turinsky A.L., “Papyrus: A System for Data Mining over Local and Wide Area Clusters and Super-Clusters”. *Technical report*, University of Illinois at Chicago.
5. Rana O., Walker D., Li M., Lynden S., and Ward M., ”PaDDMAS: Parallel and Distributed Data Mining Application Suit”. *Proceedings of the Fourteenth International Parallel and Distributed Processing Symposium, pages 387-392.*
6. Kusiak A., ”Decomposition in Data Mining: An Industrial Case Study”. *IEEE Transaction on Electronics Packaging Manufacturing, Vol. 23 No 4 October 2000.*
7. Kargupta H., Hamzaoglu I., and Stafford B. “Scalable, Distributed Data Mining An Agent Based Application”. *Proceedings of Knowledge Discovery And Data Mining, August, 1997.*
8. Kargupta, H., Park, B., Hershberger, D., and Johnson, E., (1999), "Collective Data Mining: A New Perspective Toward Distributed Data Mining". *Advances in Distributed and Parallel Knowledge Discovery*, 1999. MIT/AAAI Press.

9. Prodromidis, A., Chan, P., and Stolfo, S. (2000). “Meta-Learning in Distributed Data Mining Systems: Issues and Approaches”. *Advances in Distributed and Parallel Knowledge Discovery,* AAAI/MITPress.
10. Zaki M. “Parallel and Distributed Association Mining: A survey”. *IEEE Concurrency, special issue on Parallel Mechanisms for Data Mining, 7(4):14--25, December.*
11. Lee W., Salvatore J. S., Philip K. C., Eleazar E., Wei F., Matthew M., Shlomo H., and Junxin Z.. ”Real Time Data Mining-based Intrusion Detection.” *Proceedings of DISCEX II. June 2001*.
12. Sally M., “Distributed Data Mining”. *Proceeding of Intelligence in Industry,* Issue 3, 2001.
13. Chattratichat J., Darlington J, Guo Y., Hedvall S., Kohler M., and Syed J., “An Architecture for Distributed Enterprise Data Mining”. *HPCN, Amsterdam, 1999*.