AN OPTIMIZED DISTRIBUTED ASSOCIATION RULE MINING ALGORITHM IN PARALLEL AND DISTRIBUTED DATA MINING WITH XML DATA FOR IMPROVED RESPONSE TIME.

Dr (Mrs).Sujni Paul
Associate Professor
Department of Computer Applications,
Karunya University, Coimbatore 641114 , Tamil Nadu, India
sujni_paul@yahoo.com

ABSTRACT

Many current data mining tasks can be accomplished successfully only in a distributed setting. The field of distributed data mining has therefore gained increasing importance in the last decade. The Apriori algorithm by Rakesh Agarwal has emerged as one of the best Association Rule mining algorithms. Ii also serves as the base algorithm for most parallel algorithms. The enormity and high dimensionality of datasets typically available as input to problem of association rule discovery, makes it an ideal problem for solving on multiple processors in parallel. The primary reasons are the memory and CPU speed limitations faced by single processors. In this paper an Optimized Distributed Association Rule mining algorithm for geographically distributed data is used in parallel and distributed environment so that it reduces communication costs. The response time is calculated in this environment using XML data.

KEYWORDS

Association rules, Apriori algorithm, parallel and distributed data mining, XML data, response time.

1. INTRODUCTION

Association rule mining (ARM) has become one of the core data mining tasks and has attracted tremendous interest among data mining researchers. ARM is an undirected or unsupervised data mining technique which works on variable length data, and produces clear and understandable results. There are two dominant approaches for utilizing multiple processors that have emerged distributed memory in which each processor has a private memory; and shared memory in which all processors access common memory [4]. Shared memory architecture has many desirable properties. Each processor has direct and equal access to all memory in the system. Parallel programs are easy to implement on such a system. In distributed memory architecture each processor has its own local memory that can only be accessed directly by that processor [10]. For a processor to have access to data in the local memory of another processor a copy of the desired data element must be sent from one processor to the other through message passing. XML data are used with the Optimized Distributed Association Rule Mining Algorithm. A parallel application could be divided into number of tasks and executed concurrently on different processors in the system [9]. However the performance of a parallel application on a distributed system is mainly dependent on the allocation of the tasks comprising the application onto the available processors in the system.

Modern organizations are geographically distributed. Typically, each site locally stores its ever increasing amount of day-to-day data. Using centralized data mining to discover useful patterns in such organizations' data isn't always feasible because merging data sets from different sites into a centralized site incurs huge network communication costs. Data from these organizations are not only distributed over various locations but also vertically fragmented, making it difficult if not impossible to combine them in a central location. Distributed data mining has thus emerged as an active subarea of data mining research. In this paper an Optimized Association Rule Mining Algorithm is used for performing the mining process.

2. RELATED WORK

Three parallel algorithms for mining association rules [2], an important data mining problem is formulated in this paper. These algorithms have been designed to investigate and understand the performance implications of a spectrum of trade-offs between computation, communication, memory usage, synchronization, and the use of problem-specific information in parallel data mining [11]. Fast Distributed Mining of association rules, which generates a small number of candidate sets and substantially reduces the number of messages to be passed at mining association rules [3].

Algorithms for mining association rules from relational data have been well developed. Several query languages have been proposed, to assist association rule mining such as [18], 19]. The topic of mining XML data has received little attention, as the data mining community has focused on the development of techniques for extracting common structure from heterogeneous XML data. For instance, [20] has proposed an algorithm to construct a frequent tree by finding common subtrees embedded in the heterogeneous XML data. On the other hand, some researchers focus on developing a standard model to represent the knowledge extracted from the data using XML. JAM [21] has been developed to gather information from sparse data sources and induce a global classification model. The PADMA system [22] is a document analysis tool working on a distributed environment, based on cooperative agents. It works without any relational database underneath. Instead, there are PADMA agents that perform several relational operations with the information extracted from the documents.

3. ASSOCIATION RULE MINING ALGORITHMS

An association rule is a rule which implies certain association relationships among a set of objects (such as ``occur together'' or ``one implies the other'') in a database. Given a set of transactions, where each transaction is a set of literals (called items), an **association rule** is an expression of the form X Y , where X and Y are sets of items. The intuitive meaning of such a rule is that transactions of the database which contain X tend to contain Y [1].

3.1 Apriori Algorithm

An association rule mining algorithm, Apriori has been developed for rule mining in large transaction databases by IBM's Quest project team [3]. An *itemset* is a non-empty set of items.

They have decomposed the problem of mining association rules into two parts

- Find all combinations of items that have transaction support above minimum support. Call those combinations frequent itemsets.
- Use the frequent itemsets to generate the desired rules. The general idea is that if, say, ABCD and AB are frequent itemsets, then we can determine if the rule AB CD holds by computing the ratio r = support(ABCD)/support(AB). The rule holds only if r >= minimum confidence. Note that the

rule will have minimum support because ABCD is frequent. The algorithm is highly scalable [7]. The Apriori algorithm used in Quest for finding all frequent itemsets is given below.

```
procedure AprioriAlg()
begin

    $L_1$ := {frequent 1-itemsets};
    for ( k := 2; $L_{k-1\ 0;\ k++\ )}$ do {
       $C_k$= apriori-gen($L_{k-1)}$ ; // new candidates
    for all transactions t in the dataset do {
      for all candidates c $C_k$ contained in t do
              c:count++
      }
      $L_k$ = { c $C_k$ l c:count >= min-support}
    }
    Answer := $_k$ $L_k$

    end
```

It makes multiple passes over the database. In the first pass, the algorithm simply counts item occurrences to determine the frequent 1-itemsets (itemsets with 1 item). A subsequent pass, say pass k, consists of two phases. First, the frequent itemsets L_{k-1} (the set of all frequent (k-1)-itemsets) found in the (k-1)th pass are used to generate the candidate itemsets C_k, using the apriori-gen() function. This function first joins L_{k-1} with L_{k-1}, the joining condition being that the lexicographically ordered first k-2 items are the same. Next, it deletes all those itemsets from the join result that have some (k-1)-subset that is not in L_{k-1} yielding C_k. The algorithm now scans the database. For each transaction, it determines which of the candidates in C_k are contained in the transaction using a hash-tree data structure and increments the count of those candidates [8], [14]. At the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding L_k. The algorithm terminates when L_k becomes empty.

3.2 Distributed/parallel algorithms

Databases or data warehouses may store a huge amount of data to be mined. Mining association rules in such databases may require substantial processing power [6]. A possible solution to this problem can be a distributed system.[5] . Moreover, many large databases are distributed in nature which may make it more feasible to use distributed algorithms.

Major cost of mining association rules is the computation of the set of large itemsets in the database. Distributed computing of large itemsets encounters some new problems. One may compute locally large itemsets easily, but a locally large itemset may not be globally large. Since it is very expensive to broadcast the whole data set to other sites, one option is to broadcast all the counts of all the itemsets, no matter locally large or small, to other sites. However, a database may contain enormous combinations of itemsets, and it will involve passing a huge number of messages.

A distributed data mining algorithm FDM (Fast Distributed Mining of association rules) has been proposed by [5], which has the following distinct features.

1. The generation of candidate sets is in the same spirit of Apriori. However, some relationships between locally large sets and globally large ones are explored to generate a smaller set of candidate sets at each iteration and thus reduce the number of messages to be passed.

2. After the candidate sets have been generated, two pruning techniques, local pruning and global pruning, are developed to prune away some candidate sets at each individual sites.

3. In order to determine whether a candidate set is large, this algorithm requires only O(n) messages for support count exchange, where n is the number of sites in the network. This is much less than a straight adaptation of Apriori, which requires $O(n^2)$ messages. [3]

Distributed data mining refers to the mining of distributed data sets. The data sets are stored in local databases hosted by local computers which are connected through a computer network [15], [16]. Data mining takes place at a local level and at a global level where local data mining results are combined to gain global findings. Distributed data mining is often mentioned with parallel data mining in literature. While both attempt to improve the performance of traditional data mining systems they assume different system architectures and take different approaches. In distributed data mining computers are distributed and communicate through message passing. In parallel data mining a parallel computer is assumed with processors sharing memory and or disk. Computers in a distributed data mining system may be viewed as processors sharing nothing. This difference in architecture greatly influences algorithm design, cost model, and performance measure in distributed and parallel data mining.

3.3 Distributed Algorithms [23]

- Distributed association rule learning
- Collective decision tree learning
- Collective PCA and PCA-based clustering
- Distributed hierarchical clustering
- Other distributed clustering algorithms
- Collective Bayesian network learning
- Collective multi-variate regression

3.4 Parallel Algorithms

The main challenges associated with parallel data mining include
- minimizing I/O
- minimizing synchronization and communication
- effective load balancing [12]
- effective data layout
- deciding on the best search procedure to use
- good data decomposition
- minimizing/avoiding duplication of work[2]

The Four Parallel Algorithms are

1. Count Distribution – parallelizing the task of measuring the frequency of a pattern inside a database
2. Candidate Distribution – parallelizing the task of generating longer patterns
3. Hybrid Count and Candidate Distribution – a hybrid algorithm that tries to combine the strengths of the above algorithms
4. Sampling with Hybrid Count and Candidate Distribution – an algorithm that tries to only use a sample of the database.

The speed and the efficiency of parallel formulations are discussed below.In a parallel data mining the main issues taken into account are

- Load balancing
- Minimizing communication
- Overlapping communication and computation

Speed Up

Serial fraction : f_s

Parallel fraction : $f_p = 1 - f_s$

Speedup : $S = \frac{T_s}{T_p} = \frac{1}{f_s + \frac{f_p}{p}}$

For a maximum speed up there are limits to the scalability of parallelism. For example, at fp = 0.50, 0.90 and 0.99, 50%, 90% and 99% of the code is parallelizable. However, certain problems demonstrate increased performance by increasing the problem size. Problems which increase the percentage of parallel time with their size are more "scalable" than problems with a fixed percentage of parallel time.

Efficiency

Given the parallel cost: $p \cdot T_p = T_s + T_o$

Efficiency E: $E = \frac{T_s}{p \cdot T_p} = \frac{1}{1 + \frac{T_o}{T_s}}$

In general, the total overhead is an increasing function of p, at least linearly when fs > 0:

- communication,
- extra computation,
- idle periods due to sequential components,
- idle periods due to load imbalance.

4. OPTIMIZED DISTRIBUTED ASSOCIATION RULE MINING ALGORITHM

The performance of Apriori ARM algorithms degrades for various reasons. It requires *n* number of database scans to generate a frequent *n*-itemset. Furthermore, it doesn't recognize transactions in the data set with identical itemsets if that data set is not loaded into the main memory. Therefore, it unnecessarily occupies resources for repeatedly generating itemsets from such identical transactions. For example, if a data set has 10 identical transactions, the Apriori algorithm not only enumerates the same candidate itemsets 10 times but also updates the support counts for those candidate itemsets 10 times for each iteration. Moreover, directly loading a raw data set into the main memory won't find a significant number of identical transactions because each transaction of a raw data set contains both frequent and infrequent items. To overcome these problems, we don't generate candidate support counts from the raw data set after the first pass. This technique not only reduces the average transaction length but also reduces the data set size significantly, so we can accumulate more transactions in the main memory. The number of items in the data set might be large, but only a few will satisfy the support threshold..

Consider the sample data set in Figure 1a. If we load the data set into the main memory, then we only find one identical transaction (ABCD), as Figure 1b shows. However, if we load the data set into the main memory after eliminating infrequent items from every transaction (that is, items that don't have 50 percent of support and thus don't occur once in every second transaction in this case, itemset E), we find more identical transactions (see Figure 1c). This technique not only reduces average transaction size but also finds more identical transactions. The following gives the pseudocode of ODAM algorithm [17].

```
NF={Non-frequent global 1-itemset}
for all transaction t £ D {
  for all 2-subsets s of t
        if (s £ $C_2$) s.sup++:
        $t^/$=delete_nonfrequent_items(t);
        Table.add($t^/$);
}
Send_to_receiver ($C_2$);
/* Global Frequent support counts from receiver*/
$F_2$=receive_from_receiver(Fα);
$C_3$=(Candidate itemset);
T=Table.getTransactions(); k=3;
While ($C_k$ ≠{}) {
 For all transaction t £ T
        For all k-subsets s of t
                If (s £ $C_k$) s.sup++;
k++;
send_to_receiver($C_k$);
/* Generating candidate itemset of k+1 pass*/
Ck+1={Candidate itemset);
}
```

Transactions	
No.	Items
1.	ABCD
2.	BC
3.	AB
4.	ABCDE
5.	ACD
6.	ABE
7.	CDE
8.	AD
9.	BCE
10.	ABCD

Figure 1.(a) An Example Dataset

Transactions	
No.	Items
1,10	ABCD
2.	BC
3.	AB
4.	ABCDE
5.	ACD
6.	ABE
7.	CDE
8.	AD
9.	BCE

Figure 1. (b) Identical transactions

Transactions	
No.	Items
1,4,10	ABCD
2.9	BC
3,6	AB
5	ACD
7	CDE
8	AD

Figure 1. (c) Transactions after pruning infrequent items.

ODAM eliminates all globally infrequent 1-itemsets from every transaction and inserts them into the main memory; it reduces the transaction size (the number of items) and finds more identical transactions. This is because the data set initially contains both frequent and infrequent items. However, total transactions could exceed the main memory limit.

ODAM removes infrequent items and inserts each transaction into the main memory. While inserting the transactions, it checks whether they are already in memory. If yes, it increases that transaction's counter by one. Otherwise, it inserts that transaction into the main memory with a count equal to one. Finally, it writes all main-memory entries for this partition into a temp file. This process continues for all other partitions.

5. PARALLEL AND DISTRIBUTED ASSOCIATION RULE WITH XML DATA

Parallelism is expected to relieve current ARM methods from sequential bottlenecks, providing the ability to scale to massive datasets and improving the response time [13]. The parallel design space spans 3 main components including the hardware platform, the kind of parallelism exploited and the load balancing strategy used.

Hardware Platform

Shared memory architecture has all the processors access common memory. Each processor has direct and equal access to all the memory in the system. Parallel programs are easy to implement on such a system.

Data Parallelism

The data warehouse is partitioned among P processors logically partitioned for SMP. Each processor works on its local partition of the database but performs the same computation of counting support.

Load Balancing

Dynamic load balancing seeks to address this issue by balancing the load and reassigning the loads to the lighter ones. The development of distributed rule mining is a challenging and critical task, since it requires knowledge of all the data stored at different locations and the ability to combine partial results from individual databases into a single result.

The association rule from XML data with a sample XML document is considered. We refer to the sample XML document, depicted in Figure 2 where information about the items purchased in each transaction are represented. For example, the set of transactions are identified by the tag <transactions> and each transaction in the transactions set is identified by the tag <transaction>. The set of items in each transaction Figure 2: Transaction document (transactions.xml) are identified by the tag <items> and an item is identified by the tag <item>. Consider the problem of mining all association rules among items that appear in the transactions document as shown in Figure 2. With the understanding of traditional association rule mining we expect to obtain the large itemsets document and association rules document from the source document.

Let the minimum support (minsup) = 30%

and minimum confidence (minconf) = 100%.

```
<transactions>
        <transaction id=1>
                <items>
                        <item> i1</item>
                        <item> i4</item>
                        <item> i7</item>
                </items>
        </transaction>


        <transaction id=2>
                <items>
                        <item> i2</item>
                        <item> i3</item>
                        <item> i5</item>
                </items>
        </transaction>
        <transaction id=3>
                <items>
                        <item> i1</item>
                        <item> i3</item>
                        <item> i7</item>
                </items>
        </transaction>
        <transaction id=4>
                <items>
                        <item> i2</item>
                        <item> i5</item>
                </items>
        </transaction>
        <transaction id=5>
                <items>
                        <item> i1</item>

                        <item> i5</item>

                </items>
        </transaction>
```

6. MINE GENERAL ASSOCIATION RULES FROM XML DATA

In XML data, multiple nesting is a problem that needs to be handled properly. Consider a file of sales receipts from a grocery chain. The grocery chain may want to group by the following information: Date, StoreId, Register, and Individual Sale. Any permutation of these attributes would be a logical construction of a file in XML [24]. With the individual sale as the attribute of most interest, consider the various nesting depths at which it may be located. At any node in an XML 'tree' the sub tree can be viewed as a record, relative to other records at that depth, or other records with similar record tags. This provides assurance that mining will be done on the correct nesting depth (along with other nesting depths also). However there is a potential for redundancy. It becomes more evident in highly nested files. In a highly nested XML file, the same set of leaf nodes may be involved in as many different records as there are nestings. The below Algorithm is used to derive General Association Rules from XML Data

```
Input: min-conf, min-sup, XML file(s)
Output: Association Rules
error check input
D <- new DOM from XML files
assign record IDs to non leaf node in D, by record type
L <- new linked list
construct set S, of all unique record types, with      elements as sets of size 1
L[1] <- S
i <- 1
do until BasketSetCollection generated from merge operation is empty
prune elements in L[i-1] not meeting min-sup
L[i] <- collection of all unique record types, with elements as sets of size i, constructed from L[i-1]
increment i
generate all possible rule combinations in L
O <- empty output set
for all rules R in L, if R meets min-conf add to O
output O
```

The basic idea is as follows. First the record ids are assigned per record type. A basketSet is constructed for each type of record encountered. An empty recordTypeList and an empty RIDList is taken first to start. These lists are parallel, in that the recordType at position n of the recordTypeList is associated with the RID at position n of the RIDList. A single path from root to leaf is considered. As the algorithm progresses along this path, it examines each node. If the node is not a leaf, it looks at the node type (recordType) and asks the basketSet associated with this recordType for a new RID. It then adds the recordType to the end of the recordType list and the RID to the end of the RIDList. If the node is a leaf (consider a leaf to be of the form <purchase>pen</purchase>) loop through the RIDList and recordType list to build Baskets.

7. PERFORMANCE EVALUATION

The number of messages that ODAM exchanges among various sites to generate the globally frequent itemsets in a distributed environment, we partition the original data set into five partitions. To reduce the dependency among different partitions, each one contains only 20 percent of the original data set's transactions. So, the number of identical transactions among different partitions is very low. ODAM provides an efficient method for generating association rules from different datasets, distributed among various sites.

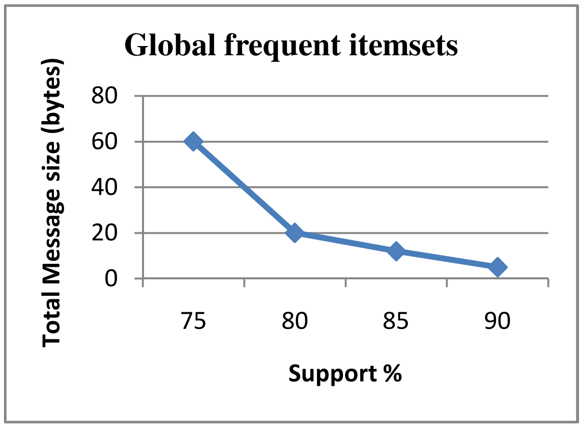

Figure 2: Total message size that ODAM transmits to generate the globally frequent itemsets

The datasets are generated randomly depending on the number of distinct items, the maximum number of items in each transaction and the number of transactions. The performance of the XQuery implementation is dependent on the number of large itemsets found and the size of the dataset.

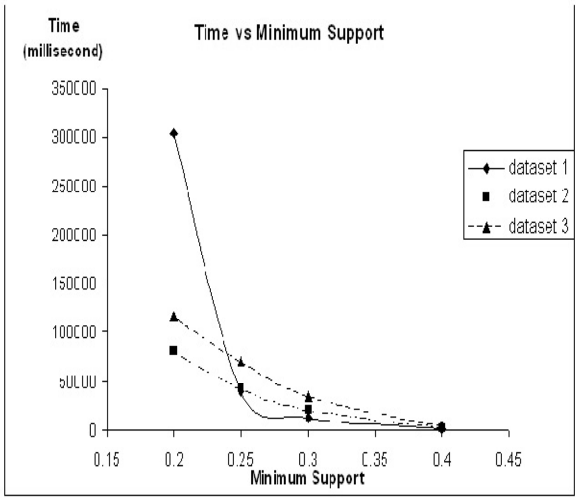

Figure 3. Time with Minimum support

The running time for dataset-1 with minimum support 20% is much higher than the running time of dataset-2 and dataset-3, since the number of large itemsets found for dataset-1 is about 2 times more than the other datasets.

The Response time of the parallel and distributed data mining task on XML data is carried out by the time taken for communication, computation cost involved. Communication time is largely dependent on the

DDM operational model and the architecture of the DDM systems. The computation time is the time to perform the mining process on the distributed data sets.

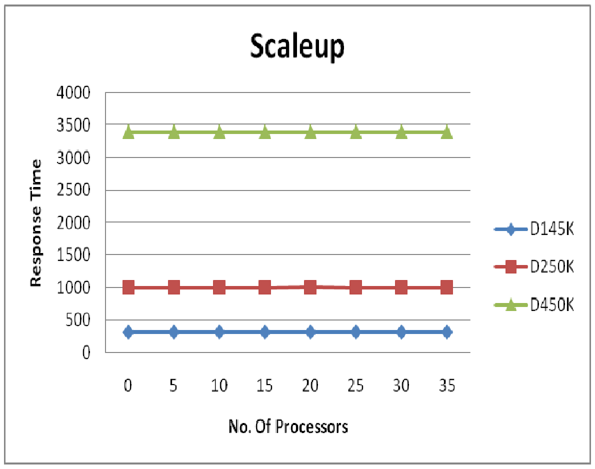

Figure 4. No. of Processors Vs Response time

A perfect scale up is found when the different types of datasets are considered. An improved response time is achieved for the taken XML data.

8. CONCLUSIONS

Association rule mining is an important problem of data mining. It's a new and challenging area to perform association rule mining on XML data due to the complexity of XML data. In our approach, multiple nesting problems in XML data is handled appropriately to assure the correctness of the result. The Optimized Distributed Association Mining Algorithm is used for the mining process in a parallel and distributed environment. The response time with the communication and computation factors are considered to achieve an improved response time. The performance analysis is done by increasing the number of processors in a distributed environment. As the mining process is done in parallel an optimal solution is obtained. The Future enhancement of this is to cluster the same XML dataset and find out the knowledge extracted out of that. A visual analysis can also be made for the same.

REFERENCES

[1] R. Agrawal and R. Srikant , "Fast Algorithms for Mining Association Rules in Large Database,"*Proc. 20th Int'l Conf. Very Large Databases* (VLDB 94), Morgan Kaufmann, 1994,pp. 407-419.

[2] R. Agrawal and J.C. Shafer , "Parallel Mining of Association Rules,"*IEEE Tran. Knowledge and* 16 IEEE Distributed Systems Online March 2004 *Data Eng. ,* vol. 8, no. 6, 1996,pp. 962-969;.

[3] D.W. Cheung , et al., "A Fast Distributed Algorithm for Mining Association Rules," *Proc. Parallel and Distributed Information Systems,* IEEE CS Press, 1996,pp. 31-42;

[4] A. Savasere , E. Omiecinski, and S.B. Navathe , "An Efficient Algorithm for Mining Association Rules in Large Databases,"*Proc. 21st Int'l Conf. Very Large Databases* (VLDB 94), Morgan Kaufmann, 1995, pp. 432-444.

[5] J. Han , J. Pei, and Y. Yin , "Mining Frequent Patterns without Candidate Generation,"*Proc. ACM SIGMOD Int'l. Conf. Management of Data ,* ACM Press, 2000,pp. 1-12.

[6] M.J. Zaki and Y. Pin , "Introduction: Recent Developments in Parallel and Distributed Data Mining,"*J. Distributed and Parallel Databases* , vol. 11, no. 2, 2002,pp. 123-127.

[7] M.J. Zaki , "Scalable Algorithms for Association Mining,"*IEEE Trans. Knowledge and Data Eng.,*vol.12 no. 2, 2000,pp. 372-390;

[8] J.S. Park , M. Chen, and P.S. Yu , "An Effective Hash Based Algorithm for Mining Association Rules,"*Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data* , ACM Press, 1995, pp. 175-186.

[9] M.J. Zaki , et al., *Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors* , tech. report TR 618, Computer Science Dept., Univ. of Rochester, 1996.

[10] D.W. Cheung , et al., "Efficient Mining of Association Rules in Distributed Databases,"*IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 6, 1996,pp.911-922;

[11] A. Schuster and R. Wolff , "Communication-Efficient Distributed Mining of Association Rules," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* ACM Press, 2001,pp. 473-484.

[12] M.J. Zaki , "Parallel and Distributed Association Mining: A Survey,"*IEEE Concurrency,* Oct.- Dec. 1999,pp. 14-25;

[13] C.L. Blake and C.J. Merz , UCI Repository of Machine Learning Databases, Dept. of Information and Computer Science, University of California, Irvine, 1998;

[14] T. Shintani and M. Kitsuregawa , "Hash-Based Parallel Algorithms for Mining Association Rules,"*Proc. Conf. Parallel and Distributed Information Systems,* IEEE CS Press, 1996, pp. 19-30;

[15] C.C. Aggarwal and P.S. Yu , "A New Approach to Online Generation of Association Rules,"*IEEE Trans. Knowledge and Data Eng.* , vol. 13, no. 4, 2001,pp.527-540;.

[16] G.W. Webb , "Efficient Search for Association Rules,"*Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD 00), ACM Press, 2000,pp. 99-107.

[17] Mafruz Zaman Ashrafi, David Taniar,Kate Smith, *Monash University* "ODAM: An Optimized Distributed Association Rule Mining Algorithm", IEEE distributed systems online 1541-4922 © 2004 published by the ieee computer society vol. 5, no. 3; march 2004

[18] T. Imielinski and A. Virmani. MSQL: A query language for database mining. 1999.

[19] R. Meo, G. Psaila, and S. Ceri. A new SQLlike operator for mining association rules. In The VLDB Journal, pages 122–133, 1996.

[20] A. Termier, M.-C. Rousset, and M. Sebag. Mining XML data with frequent trees. In DBFusion Workshop'02, pages 87–96.

[21] A. Prodromidis, P. Chan, and S. Stolfo. Chapter Meta-learning in distributed data mining systems: Issues and approaches. AAAI/MIT Press, 2000.

[22] Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, distributed data mining-an agent architecture. In Heckerman et al. [8], page 211.

[23] Albert Y. Zomaya, Tarek El-Ghazawi, Ophir Frieder, "Parallel and Distributed Computing for Data Mining", IEEE Concurrency, 1999.

[24] Qin Ding, Kevin Ricords, and Jeremy Lumpkin, “Deriving General Association Rules from XML Data”*Department of Computer Science Pennsylvania State University at Harrisburg Middletown, PA 17057, USA*

Sujni Paul obtained her Bachelors degree in Physics from Manonmanium Sundaranar University during 1997 and Masters Degree in Computer Applications from Bharathiar University during 2000. Completed her PhD in Data Mining in the Department of Computer Applications, Karunya University, Coimbatore, India. She is working in the area of parallel and distributed data mining. She is working in Karunya University as Associate Professor for the past 8 years to till date.