A COMPARATIVE STUDY OF DISTRIBUTED ALGORITHMS IN MINING ASSOCIATION RULES

Cornelia Győrödi*

**Department of Computer Science, Faculty of Electrotehnics and Informatics*
University of Oradea, Oradea, ROMANIA
E-mail: rgyorodi@rdsor.ro or cgyorodi@univ.uoradea.ro

Abstract. With the ever-growing database sizes, we have enormous quantities of data, but unfortunately we cannot use raw data in our day-to-day reasoning/decisions. We desperately need knowledge. This knowledge is in most cases in the gathered data, but the extraction of it is a very time and resources consuming operation. Association rule mining finds interesting association or correlation relationships among a large set of data items. The paper presents some considerations about distributed association rules mining together with a comparison between two representative distributed algorithms, namely CDA and FDM. The compared algorithms are presented together with some experimental data that leads to the final conclusions.

Keywords – Data Mining, Distributed Mining Algorithm, CDA, FDM.

1. INTRODUCTION

The continuing growth of database sizes and mainly the stringent need of pertinent information for decision support increased the interest in automatic knowledge discovery in databases.

The implicit information within databases, and mainly the interesting association relationships among sets of objects, that lead to association rules, may disclose useful patterns for decision support, financial forecast, marketing policies, even medical diagnosis and many other applications. This fact has attracted a lot of attention in recent data mining research (Fayyad *et al.* 1996). As shown by Agrawal and Shrikant (1994), mining association rules may require iterative scanning of large databases, which is costly in processing. Many authors focussed their work on efficient mining of association rules in databases [(Agrawal and Shrikant 1994), (Han *et al.* 2000), (Dunham 2003), (Fayyad *et al.* 1996), (Han and Kamber 2001)].

Association rule mining finds interesting association or correlation relationships among a large set of data items (Han and Kamber 2001). The association rules are considered interesting if they satisfy both a *minimum support* threshold and a *minimum confidence* threshold (Gyorodi C. and Gyorodi R. 2002a).

A very influential association rule mining algorithm, Apriori (Agrawal and Shrikant 1994), has been developed for rule mining in large transaction databases. Many other algorithms developed are derivative and/or extensions of this algorithm. A large step forward in improving the performances of these algorithms was made by introduction of a novel, compact data structure, called *frequent pattern tree*, or FP-tree (Han J. *et al.* 2000), and the associated mining algorithms, FP-growth (Han J. *et al.* 2000). Although these studies are on sequential data mining techniques, algorithms for parallel mining of association rules have been proposed recently [(Cheung *et al.* 1996), (Agrawal and Shafer 1996)].

The development of distributed algorithms for efficient mining of association rules has importance, based on the following reasoning. (1) Databases may store a huge amount of data. Mining association rules in such databases may require substantial processing power, and a distributed system is a possible solution. (2) Many large databases are distributed at different sites. Based on these observations, the study of efficient distributed algorithms for mining association rules is very important. Further more, a distributed mining algorithm can also be used to mine association rules in a single large database by partitioning the databases among a set of sites and processing the task in a distributed manner.

This study assumes that the database to be studied is a transaction databases. The databases consist of a huge number of transaction records, each with a transaction identifier and a set of data items, and it is "horizontally" partitioned and allocated to the sites in a distributed system which communicates by message passing.

This paper presents a comparative study of the most important distributed algorithms used in association rules mining CDA (Count Distributed Algorithm) (Agrawal and

Shafer 1996) and FDM (Fast Distributed Mining of association rules) (Cheung *et al.* 1996). It also includes a performance study that shows the advantages and disadvantage of the distributed algorithms CDA and FDM.

2. Problem Definition

The basic problem definition of association rules mining is given in (Agrawal and Shrikant 1994). Here are presented only the aspects that are specific to association rules mining in a distributed environment. Let D a transactional database with P transactions. Suppose that there are n sites $S_1, S_2, ..., S_n$ in a distributed system and that the database is correspondently partitioned on the n sites into $\{D_1, D_2, ..., D_n\}$.

Let P_i denote the dimension of partition D_i for $i = 1, ..., n$. Also let $X.sup$ and $X.sup_i$ denote the support factor of the itemset X from D and respectively D_i. $X.sup$ is known as the *global support counter*, and $X.sup_i$ as the *local support counter* for site i. For a given minimum support s, X is *globally frequent* if $X.sup \geq s \times P$, correspondently, X is *locally frequent* at site S_i, if $X.sup_i \geq s \times P_i$. In the following, L will correspond to globally frequent itemsets from D, and $L_{(k)}$ to globally frequent k-itemsets from L. The main goal of a distributed association rules mining algorithm is finding the globally frequent itemsets L.

3. Distributed Algorithms in Association Rules Mining

According to (Dunham 2003) most parallel or distributed association rule algorithms strive to parallelize either the data, known as *data parallelism*, or the candidates, referred to as *task parallelism*. With task parallelism, the candidates are partitioned and counted separately at each processor. Obviously, the partition algorithm would be easy to parallelize using the task parallelism approach. Other dimensions in differentiating the parallel association rule algorithms are the load-balancing approach used and the architecture. The data parallelism algorithms have reduced communication costs over the task, because only the initial candidates (the set of items) and the local counts must be distributed at each iteration. With task parallelism, not only the candidates but also the local set of transactions must be broadcast to all other sites. However, the data parallelism algorithms require that memory at each processor be large enough to store all candidates at each scan (otherwise the performance will degrade considerably because I/O is required for both the database and the candidate set). The task parallelism approaches can avoid this because only the subset of the candidates that are assigned to a processor during each scan must fit into memory. Since not all partitions of the candidates must be the same size, the task parallel algorithms can adapt to the amount of memory at each site. The only restriction is that the total size of all candidates be small enough to fit into the total size of memory in all processors combined. Performance studies have shown that the data parallelism tasks scale linearly with the number of processors and the database size. Because of the reduced memory requirements, however, the task parallelism may work where data parallelism may not work.

3.1. The CDA Algorithm

One data parallelism algorithm is the *count distribution algorithm (CDA)*. The database is divided into p partitions, one for each processor. Each processor counts the candidates for its data and then broadcasts its counts to all other processors. Each processor then determines the global counts. These then are used to determine the large itemsets and to generate the candidates for the next scan. The algorithm according to (Dunham 2003) is shown below.

Input:
I // *itemsets*
$p^1, p^2, ...p^p$ //processors
$D = D^1, D^2, ...D^p$ //database divided into partitions
s //support
Output:
L //large itemsets
Count distribution algorithm:
Perform in parallel at each processor p^1;
//count in parallel
$k = 0;$ //k is used as the scan number
$L = \varnothing;$
$C_1 = I;$ //initial candidates are set to be the items
repeat
  $k = k + 1;$
  $L_k = \varnothing;$
  for each $I_i \in C_k$ **do**
    $c^1_i = 0;$ //initial counts for each itemset are 0
    for each $t_j \in D^1$ **do**
      for each $I_i \in C_k$ **do**
        if $I_i \in t_i$ **then**
          $c^1_i = c^1_i + 1;$
  broadcast c^1_i *to all other processors;*
  for each $I_i \in C_k$ **do** //determine global counts
    $c_i = \sum^p_{l=1} = c^1_i;$
    for each $I_i \in C_k$ **do**
      if $c_i \geq (s \times | D^1 \cup D^2 \cup ... \cup D^p |)$ **then**
        $L_k = L_k \cup I_i;$
  $L = L \cup L_k;$
  $C_{k+1} = Apriori\text{-}Gen(L_k)$
until $C_{k+1} = \varnothing$.

3.2. The FDM Algorithm

The FDM (Fast Distributed Algorithm for Data Mining) algorithm, proposed in (Cheung *et al.* 1996) has the following distinguishing characteristiques:

1. Candidate set generation is Apriori-like. However, some interesting properties of locally and globally frequent itemsets are used to generate a reduced set of candidates at each iteration, this resulting in a reduction in the number of messages interchanged between sites.
2. After the candidate sets were generated, two types of reduction techniques are applied, namely a local reduction and a global reduction, to eliminate some candidate sets from each site.
3. To be able to determine if a candidate set is frequent, the algorithm needs only $O(n)$ messages for the exchange of support counts, where n is the number of sites from the distributed system. This number is much less than a disrect adaptation of Apriori, which would need $O(n^2)$ messages for calculating the support counts.

The algorithm according to (Cheung *et al.* 1996) is shown below.

Input:
DB_i //database partition at each site S_i
Output:
L //set of all globally large itemsets
Algorithm:
Iteratively execute the following program fragment (for the k-th iteration) distributively at each site S_i. The algorithm terminates when either $L_{(k)} = \varnothing$, or the set of candidate sets $CG_{(k)} = \varnothing$.

if $k = 1$ **then**
  $T_{i(1)} = get\_local\_count(DB_i, \varnothing, 1)$
else {
  $CG_{(k)} = \cup^n_{i=1} CG_{i(k)} = \cup^n_{i=1} Apriori\_gen(GL_{i(k-1)})$
  $T_{i(k)} = get\_local\_count(DB_i, CG_{(k)}, i)$ }
for each $X \in T_{i(k)}$ **do**
  if $X.sup_i \geq s \times D_i$ **then**
    for $j = 1$ *to* n **do**
      if $polling\_site(X) = S_j$ **then**
        insert $\langle X, X.sup_i \rangle$ *into* $LL_{i,j(k)}$
for $j = 1$ *to* n **do**
  send $LL_{i,j(k)}$ *to site* S_j
for $j = 1$ *to* n **do** {
  receive $LL_{j,i(k)}$
  for each $X \in LL_{j,i(k)}$ **do** {
    if $X \notin LPi(k)$ **then**
      insert X into $LP_{i(k)}$
    update X.large_sites } }
for each $X \in LP_{i(k)}$ **do**
  send_polling_request(X);
reply_polling_request($T_{i(k)}$)
for each $X \in LP_{i(k)}$ **do** {
  receive $X.sup_j$ *from sites* S_j
  where $S_j \notin X.large\_sites$
    $X.sup = \sum^n_{i=1} X.sup_i$
  if $X.sup \geq s \times D$ **then**
    insert X into $G_{i(k)}$ }
broadcast $G_{i(k)}$
receive $G_{j(k)}$ *from all other sites* S_j, $(j \neq i)$
$L_{(k)} = \cup^n_{i=1} G_{i(k)}$
divide $L_{(k)}$ *into* $GL_{i(k)}$, $(I = 1,\ldots,n)$
return $L_{(k)}$.

4. Comparative Study

The data set used for testing the performance of the two algorithms, CDA and FDM, was generated according to (Agrawal and Shrikant 1994), by setting the number of items N = 100, and the maximum number of frequent itemsets |L| = 3000, also the mean dimension of a transaction |T| = 10. To test the described algorithms 2 to 5 workstations with the following configuration: Pentium 4 at 1.5GHz, 512 MRAM, Windows 2000 Professional OS and 100Mb Ethernet network were used. The algorithms were implemented in Java 1.4. To study the algorithms the support factor was varied between 0.5% and 40%.

A first result, obtained by testing the two algorithms on data sets with 50000 to 520000 transactions and, as mentioned before, using between 2 and 5 workstations with a support factor of 5% is shown in Figure 1. This figure shows that the performance of the algorithm depends on the number of processors and the number of transactions. For a data set with 520000 transactions that was distributed on two workstations, the execution time for the CDA algorithm was 5583 seconds and for the FDM was 4055 seconds, while the same data set distributed on five workstations produced an execution time of just 1161 seconds for the CDA algorithm and 892 seconds for the FDM algorithm. So, in order to obtain fairly small execution times, we need to increase the number of processors if we increase the number of transactions of the data set.

For a relatively small data set distributed on a large number of workstations the execution time could be quite long because the local sets of candidates obtained for each site is large and the communication time between sites becomes considerable, as well. For example for a data set with 110000 transactions distributed on 2 sites the execution time for the CDA algorithm was 88 second and for the FDM algorithm 68 seconds, while the same data set distributed on 5 sites the execution time has risen to 120 seconds for the CDA algorithm and to 88 seconds for the FDM algorithm. Thus, when increasing the number of site (processors) the dimension of the data set must be taken into account. For a relatively small data set an increase in number of processors could lead to large sets of local candidates and a large number of messages transmitted between sites, thus, leading to an increase in execution time for the CDA and FDM algorithms.

Figure 1 shows that the performance of the algorithms increases with the number of processors, but the FDM algorithm has better performance than the CDA algorithm for the same number of processors and the same dimension of the data set, and a support factor of 5%.

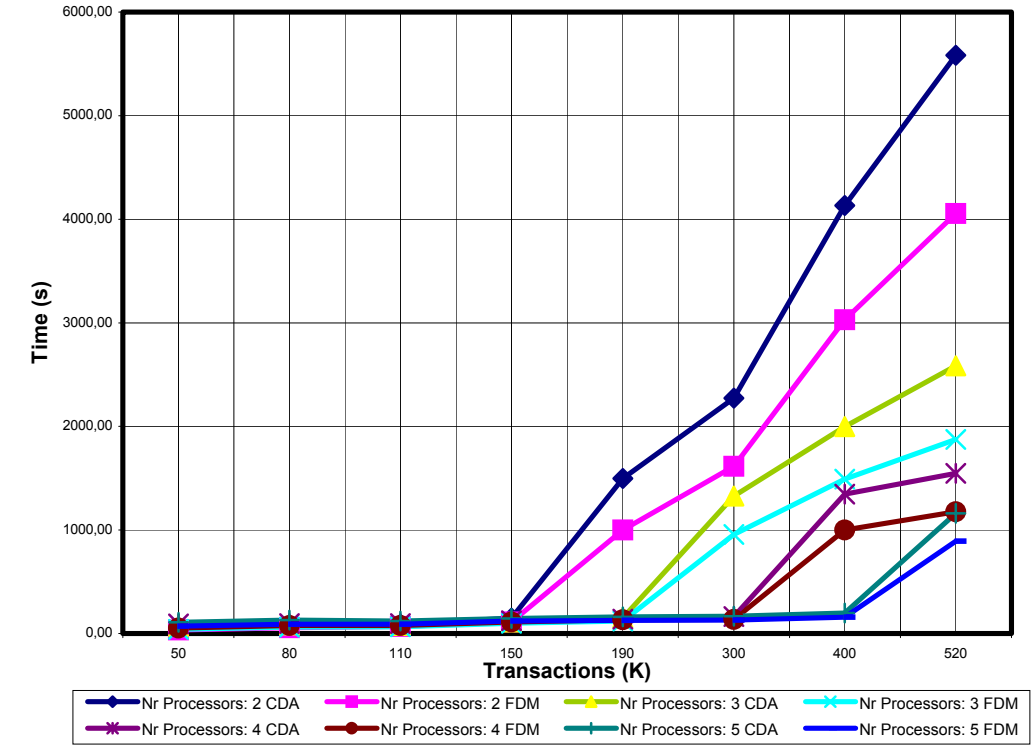

Figure 1. Scalability by transactions and number of processors (sites) (5% support)

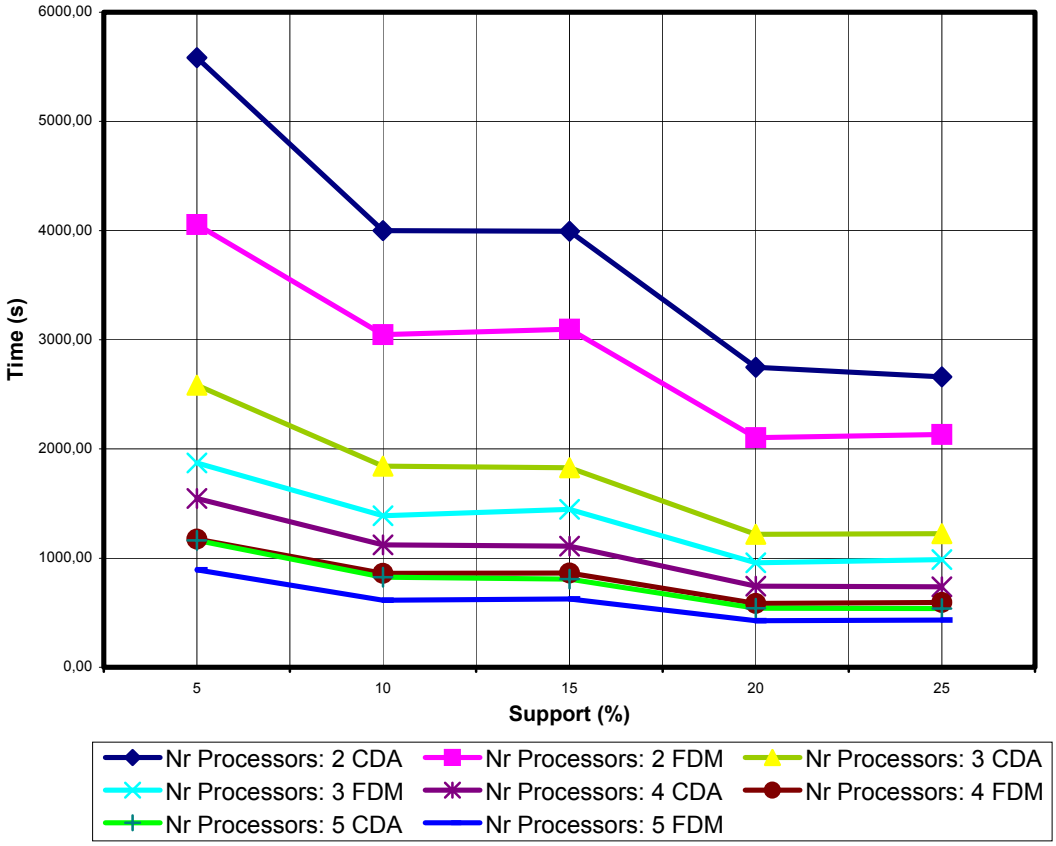

Figure 2. Scalability by support and processors (sites) (D1 520K)

Figure 2 shows that the CDA and FDM algorithms present a good scalability relative to the different support factors, for a large data set with 520000 transactions. The performance of the algorithms increases with the support factor. Also if the number of processors is increased, the performance is quite good for a small support factor of only 5%. So, for a data set with 520000 transactions distributed on 5 workstations and a support factor of 5% the execution time for the FDM algorithm is 892 seconds and for the CDA algorithm is 1161 seconds, meanwhile, for the same data set distributed on 2 workstations, the execution times increased five times, reaching 4055 for the FDM algorithm and 5583 for the CDA algorithm.

Thus, in order to increase the performance and shorten execution times of the two algorithms for large data sets with small support factors is necessary to increase the number of processors.

The comparison of the performance of the CDA and FDM algorithms for the same data set distributed on 4 sites and for different support factors is shown in Figure 3. It is noticeable that the performance of the algorithms increases with the support factor, but the FDM algorithm presents a better performance than the CDA algorithm.

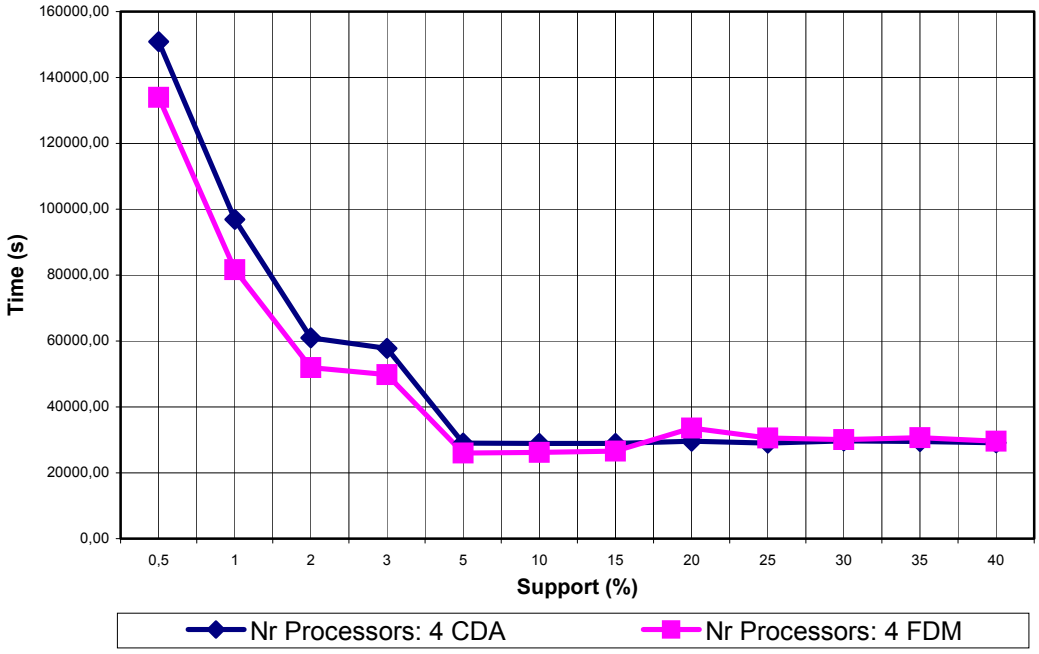

Figure 3. Scalability by support (D1 10M)

5. Conclusions

From the experiments made, resulted a good scalability for the CDA and FDM algorithms, relative to different support factors for a large data set. It is also noticeable that increasing the support factor also increases the performance of the algorithms. Also good performances were obtained when the support factor was low and the data set large, but the number of processors increased. These results show the fact that the increase in processor number should be done relative to the dimension of the data set. Thus, for a relatively small data set, the large increase in processor number can lead to large sets of local candidates and a large number of messages, thus increasing the execution time of CDA and FDM algorithms.

The CDA algorithm has a simple synchronization scheme, using only one set of messages for every step, while the FDM algorithm uses two synchronizations and the same scheme as CDA.

For the test the FDM algorithm generates a smaller candidate set and also uses a smaller number of messages than the CDA algorithm, thus leading to a smaller execution time for the FDM algorithm. The communication optimization for the FDM algorithm is done using polling-sites.

If the data is evenly distributed between sites, the CDA and FDM algorithms produce the same results, but if the data is not evenly distributed between sites, the performance of the FDM algorithm increases.

The distributed mining algorithms can be used on distributed databases, as well as for mining large databases by partitioning them between sites and processing them in a distributed manner. The high flexibility, the scalability, the small cost/performance ratio and the connectivity of a distributed system make them an ideal platform for data mining.

REFERENCES

Agrawal R. and Srikant R. (1994). "Fast algorithms for mining association rules in large databases". *Proc. of 20th Int'l conf. on VLDB*: 487-499.

Han J., Pei J., Yin Y. (2000). "Mining Frequent Patterns without Candidate Generation". *Proc. of ACM-SIGMOD*.

Gyorodi C. and Gyorodi R. (2002a). "Mining Association Rules in Large Databases". *Proc. of Oradea EMES'02*: 45-50, Oradea, Romania.

Dunham M. H. (2003). "Data Mining. Introductory and Advanced Topics". Prentice Hall, ISBN 0-13-088892-3.

Fayyad U.M., et al. (1996). "From Data Mining to Knowledge Discovery: An Overview", *Advances in Knowledge Discovery and Data Mining*:1-34, AAAI Press/ MIT Press, ISBN 0-262-56097-6.

Han J. and Kamber M. (2001), "Data Mining Concepts and Techniques", Morgan Kaufmann Publishers, San Francisco, USA, ISBN 1558604898.

Cheung D. W., Han J., Vincent T. Ng., and Ada W. Fu. (1996). "A fast distributed algorithm for mining association rules". In Proceedings of IEEE 4th International Conference on Parallel and Distributed Information Systems, pages 31-42, December.

Agrawal R. and Shafer J. C. (1996). "Parallel mining of association rules: Design, implementation and experience". In IBM Research Report.

BIOGRAPHY

Cornelia Győrödi was born on 13th April 1970, Oradea, Romania. She graduated in Computer Engineering the "Politehnica" University of Timisoara (Romania) in 1994 and received her PhD in *Computer Science* in 2003 from "Politehnica" University of Timisoara as well. Her current research interests include artificial intelligence, neural networks, database management systems, knowledge discovery in databases and data mining techniques in different domains such as databases. She participated in different TEMPUS financed projects. She is author/co-author of 3 books, and more than 35 published papers, in the aforementioned fields, many of them abroad. Mrs. Győrödi is a lecturer at University of Oradea, Romania.