Performance Analysis of PC Cluster Based Association Rule Mining Algorithms

Ferenc Kovács
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Goldmann György tér 3, H-1111 Budapest
Hungary
ferenc.kovacs@aut.bme.hu

István Vajk
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Goldmann György tér 3, H-1111 Budapest
Hungary
ferenc.kovacs@aut.bme.hu

Abstract – One of the most important problems in data mining is association rule mining. It requires very large computation and I/O traffic capacity. For that reason there are several parallel mining algorithms which can take advantage of the performance of the cluster systems. These algorithms are optimised and developed on supercomputer platforms, but nowadays the capacity of PC preserves the possibility to build cluster systems cheaper. Usage of PC cluster systems raises some issues about the optimisation of the distributed mining algorithms, especially the cost of the node to node communication and data distribution. In this paper a novel distributed association rule mining algorithm is introduced and investigated. The main parts of the execution time of the algorithm are identified by simulation results. The relationship among these parts can change in function of mining parameters. The approximations of these costs allow finding the appropriate cluster configuration.

I. INTRODUCTION

The association rule mining (ARM) is very important task within the area of data mining [1]. Many algorithms were developed to find association rules, but the Apriori is the best-known [2]. The one of the main disadvantage of the Apriori algorithm is its I/O cost. A paper [3] introduces an algorithm for I/O cost cutting, but it has higher memory requirements.

Because of the complexity of the ARM task several parallel algorithms have been developed. The main part of the distributed algorithms is based on the Apriori algorithm. The count distribution (CD) -like algorithms [4] generate the smallest network traffic, because they send only their own counters to the other nodes. The data distribution (DD)- based algorithms [4] generate higher network traffic, because the nodes send not only their local counters, but their own database as well. There are some distributed algorithms, which are not based on Apriori, for instance [5] contributes such an algorithm.

The paper [6] shows a detailed analysis of the Apriori algorithm and it concluded that the finding the 2 frequent itemsets is the significant part of the execution time. The paper [7] introduces two modifications to avoid the bottlenecks in the Apriori algorithm.

The distributed algorithms were developed and evaluated in supercomputer environment, but the PC cluster systems have several differences compared to traditional cluster systems. Therefore the contributed distributed association rule mining algorithms do not consider the data distribution cost and they begin the itemset counting after the data distribution phase. But the initial data distribution appears in case of the PC cluster systems as the database usually is stored separately from the mining cluster.

Just few distributed association rule mining algorithm were investigated in PC cluster environment. The paper [8] is concerned with the behaviour of HPA algorithm [9] in PC cluster environment. The node synchronization and possibility of different types of nodes can cause serious performance decrease in PC clusters. The PC cluster-based modification of CD and DD algorithms were discussed in [10]. The algorithm synchronization problem was examined in [11]. An asynchronous distributed algorithm was introduced in [12]

The execution time estimation is an important objective in many applications, and this is especially true for long running, resource intensive, costly data mining algorithms. Performance prediction not only allows estimating the execution time, but also helps to adjust the parameters the execution time is particularly sensitive to. It allows balancing the associated costs and the expected benefits of the execution. Good estimation of resource requirements is also important in distributed systems, where the balance of the running time and the amount of processing resources can be fine tuned.

This paper is organized as follows: first of all the widely spread sequential ARM algorithms are described. Afterwards the basic distributed ARM algorithms are summarized. After that a novel algorithm is described. Afterwards the detailed analysis of the novel algorithm is introduced. Finally the conclusion of the algorithm analysis is described.

II. PROBLEM STATEMENT

First we elaborate on some basic concepts of association rules using the formalism presented in [1]. Let $I=\{i_1,i_2,\ldots i_m\}$ be set of literals, called items. Let $D=\{t_1,t_2,\ldots t_n\}$ be set of transactions, where each transaction t is a set of items such that $t \subseteq I$. The itemset X has support s in the transaction set D if s% of transactions contains X, here we denote s= support(X). An association rule is an implication in the form of $X \rightarrow Y$, where $X, Y \subseteq I$, and $X \cap Y = \varnothing$. Each rule has two measures of value, support and confidence. The support of the rule $X \rightarrow Y$ is support($X \cup Y$). The confidence c of the rule $X \rightarrow Y$ in the transaction set D means c% of transactions in D that contain X also contain Y, which can be written in $c = S(X \cup Y)/S(X)$ form. The problem of mining association rules is to find all the rules that satisfy a user specified minimum support and minimum confidence. If support(X) is larger than a user defined minimum

support (denoted here min_sup) then the itemset X is called large itemset.

The association rule mining can be decomposed into two subproblems:

- Finding all of the large itemsets
- Generating rules from these large itemsets

The second subproblem is much easier than the first one that is the reason why the ARM algorithms are different from each other only in the method handling the first subproblem.

III. BASIC ALGORITHMS

In this section some basic association rule mining algorithms are described. First the Apriori algorithm is introduced because it is the base algorithm of the investigated distributed algorithms. Then three fundamental and referential algorithms are described: count distribution [4], data distribution [4] and hash-based partition algorithm [14]. Table 1 contains the notations that are used in detailed descriptions of the sequential algorithms.

A. Apriori Algorithm

The Apriori algorithm [2] use the following theorem to reduce the search space: if an itemset is large then all of its subsets are large as well. This means it is possible to generate the potentially large i+1 itemset using large i itemset. Each subsets of candidate i+1 itemset must be large itemset. Hereby it is possible to find all large itemset using database scan repeatedly. During the i^{th} database scan it counts the occurrence of the i itemset and by the end of the pass i, it generates the candidates, which contain i+1 item. Figure 1 shows the pseudo code of the Apriori algorithm. The main disadvantage of this algorithm is the multiple database scan. There are many solutions to reduce the number of database scans [2][3][5][13].

B. Count Distribution Algorithm

The count distribution algorithm [4] is a fundamental distributed association rule algorithm. The basic idea of this algorithm is that each of the nodes keeps large itemsets and counters of candidates locally, which are related to the whole database. These counters are maintained in accordance with the local dataset and incoming counter values. The nodes locally execute the Apriori algorithm and after reading through the local dataset they broadcast own counters to the other nodes. Hence each of the nodes can generate new candidates on the basis of the global counter values. Figure 2 shows the pseudo code of the CD algorithm and figure 3 gives an overview of the CD algorithm.

Table 1.
Notations in the sequential algorithms

k itemset	An itemset having k items
L	Set of the frequent itemset
L_i	Set of frequent i itemset
C_i	Set of candidate i itemset (potentially frequent itemset)
|A|	Number of elements in set A

$L_1 \leftarrow \{1 \text{ frequent itemsets}\}$
$C_2 \leftarrow GenerateCandidate(L_1)$
$i \leftarrow 2$
$while(L_{i-1} \neq \varnothing)$
{
$foreach(t \in D)$
{
$IncrementCounter(C_i, t)$
}
$L_i \leftarrow \left\{ c \in C_i \middle| \frac{c.counter}{|D|} \geq \min\_supp \right\}$
$i \leftarrow i+1$
$C_i \leftarrow GenerateCandidate(L_i)$
}
$L \leftarrow \bigcup_i L_i$

Fig. 1. Pseudo Code of the Apriori Algorithm

$C_1^j \leftarrow \{items\}$
$i \leftarrow 1$
do
{
$foreach(t \in D^j)$
{
$IncrementCounter(C_i^j, t)$
}
$Broadcast(C_i^j)$
$for(k = 1 \text{ to } N, k \neq j)$
{
$Receive(C_i^k)$
$ComputeCounters(C_i^j, C_i^k)$
}
$L_i \leftarrow \left\{ c \in C_i^j \middle| \frac{c.counter}{|D|} \geq \min\_supp \right\}$
$i \leftarrow i+1$
$C_i^j \leftarrow GenerateCandidate(L_{i-1})$
} $while(L_{i-1} \neq \varnothing)$
$L \leftarrow \bigcup_i L_i$

Fig. 2. Pseudo Code of the Count Distribution algorithm

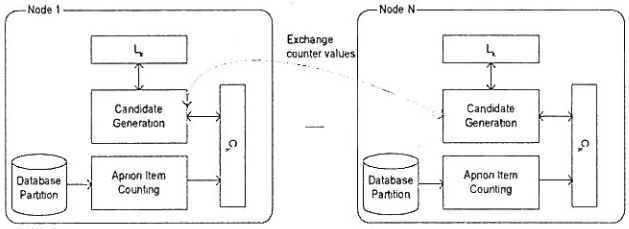

Fig. 3. The overview of the count distribution algorithm

This algorithm keeps the possibility of low data transmission, but each of the nodes is synchronized after each of the database scans. The other disadvantage of previously mentioned algorithm is that if there are too many candidates then it will not take the opportunity that there could be enough memory in whole cluster to keep all the candidates in the memory.

C. Data Distribution Algorithm

Data distribution algorithm [4] gives a solution for a situation, when one of the nodes does not have enough memory for all of the candidates. In this case each of the nodes is responsible for only a part of the candidates. Each of the nodes counts the occurrence of its own candidates in the whole dataset. But all of the nodes have to broadcast their own local database to the other nodes. Figure 4 gives an overview of the DD algorithm and Figure 5 shows the pseudo code of the algorithm.

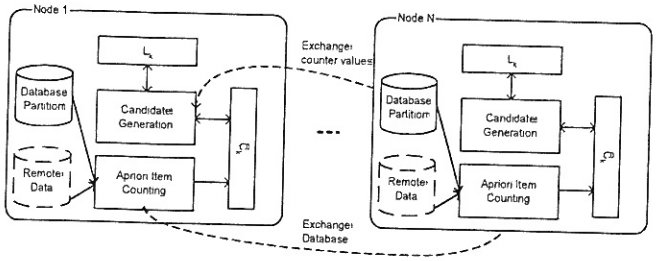

Fig. 4. Overview of the Data Distribution algorithm

$C_i^j \leftarrow \{items\}$
$i \leftarrow 1$
do
{
  foreach $(t \in D^j)$
  {
    $IncrementCounter(C_i^j, t)$
  }
  $Broadcast(D^j)$
  for $(k = 1$ *to* $N, k \neq j)$
  {
    $Receive(D^k)$
    foreach $(t \in D^k)$
    {
      $IncrementCounter(C_i^j, t)$
    }
  }
  $Broadcast(C_i^j)$
  $C_i \leftarrow C_i^j$
  for $(k = 1$ *to* $N, k \neq j)$
  {
    $Receive(C_i^k)$
    $C_i \leftarrow C_i \cup C_i^k$
  }
  $L_i \leftarrow \left\{ c \in C_i \middle| \frac{c.counter}{|D|} \geq \text{min\_supp} \right\}$
  $i \leftarrow i + 1$
  $C_i^j \leftarrow GenerateCandidate(L_{i-1})$
} *while* $(L_{i-1} \neq \emptyset)$
$L \leftarrow \bigcup_i L_i$

Fig. 5. Pseudo Code of the Data Distribution algorithm

The disadvantage of the algorithm is that it generates very large network traffic, because it does not use any kind of optimisation to reduce the network traffic. It can be noticed that the huge number of the candidates are decreasing during the later iteration step. Hence the local database broadcast is not necessary when the number of counter is decreased.

D. HPA algorithm

The HPA algorithm[14] is an improved version of the data distribution algorithm; it uses a hash function to determine the owner node of the candidate. With the help of this hash function it is possible to decide where to send each of the read itemsets. In this way the network traffic can be reduced.

V. COUNTING WHILE DISTRIBUTING ALGORITHM

The Counting While Distributing (CWD) [15] algorithm is based on count distribution algorithm but it uses a triangular matrix to find the 2 frequent itemset [7]. The other modification is that the traditional CD algorithm uses all to all broadcast mechanism to share own local itemset counter when the nodes finished a database scan but in PC cluster environment the all to all broadcast can be very expensive therefore in this algorithm there is a coordinator node. This node collects the local counter value of the candidates and generates new candidates [10][12].

The traditional distributed association rule mining algorithms do not consider the initial database distribution cost hence they start the itemset counting after the data distribution. Due to the increased searching capability of triangular matrix representation of the 2 itemset it is possible to count the 2 itemsets during the data distribution phase. Therefore after the initial data distribution the coordinator can immediately collect the counters of the 2 itemsets. This solution allows overlapping the initial data distribution and a significant part of the itemsets counting.

The asynchronous communication model can improve the overall efficiency of the cluster. The asynchronous communication model facilitates to overlap the communication and the data processing. During the initial data distribution the coordinator node creates small data fragments from the database and these are sent to the worker nodes. This fragmentation keeps the possibility to increase the efficiency of the data processing. Due to the asynchronous communication the worker nodes can process a data fragment while a new one is arriving through the network communication channel. In the current implementation the size of the fragment is 128 kilobytes.

VI. PERFORMANCE EVALUATION OF CWD ALGORITHM

The CWD algorithm is implemented in standard C++ using the pyramid [16] asynchronous message passing library on Windows XP platform. The simulation took place in the PC laboratory of the department using 11 uniform PCs having an Intel Pentium IV processor of 2.26 GHz, 256 MB RAM, and an Intel 82801DB PRO/100 VE network adapter of 100 Mbits. The nodes were connected through a switching hub of type 3Com SuperStack 4226T.

The datasets used by the algorithm are generated by IBM synthetic data generator [2]. The main parameters of the processed dataset are the number of the transactions (D), the average size of the transactions (T), and the average length of the maximal frequent itemsets (I).The parameter T varied from 10 to 25 by 5, the parameter I was in the range of 6 to 10. Earlier experiences showed that there is a liner relationship between the response time and number of transaction [17] hence this parameter was fixed to 500.000. The maximal number of the items that can occur in a transaction was 1000. The introduced figures are based on dataset T20I8D500K but the behaviour of the identified parameters is similar to the introduced ones independently of the dataset.

A. The response time of the algorithm

Figure 6 and Figure 7 show the response time of the CWD algorithm in the function of working nodes and the function of minimum support threshold.

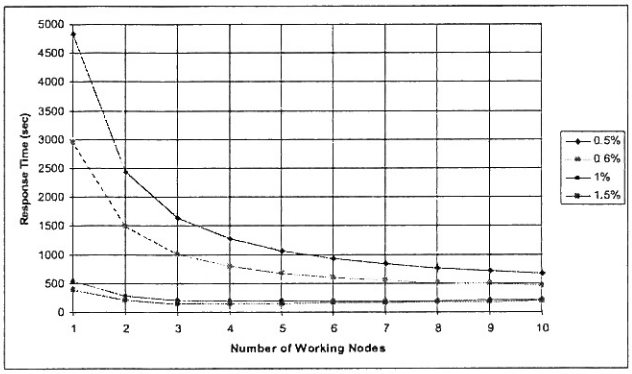

Fig. 6. Response time of the CWD algorithm in function of working nodes

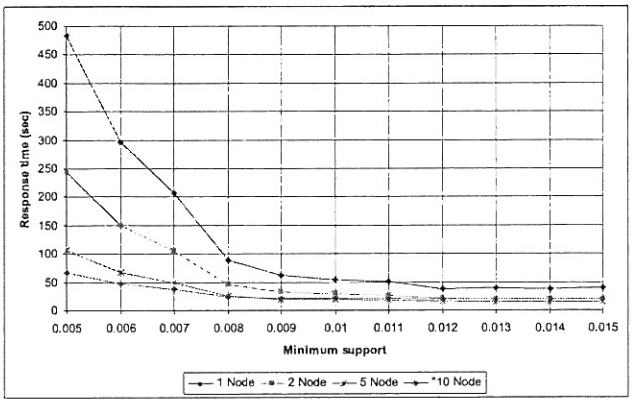

Fig. 7. Response time of the CWD algorithm in function of minimum support threshold

It is possible to identify four independent components of the response time of the CWD algorithm:

1. Costs of the initial data distribution
2. Processing time of the coordinator node (merging counters and generating new candidates)
3. Processing time of the worker nodes (itemset counting)
4. Communication costs

Figure 8 and 9 introduce the distribution of these costs in the function of the minimum support threshold. It is possible to realise that the role of these costs can change widely. .

B. The costs of the initial data distribution

Basically the initial data distribution is independent of the minimum support threshold and it depends on the number of working nodes. Figure 10 describes the initial data distribution costs.

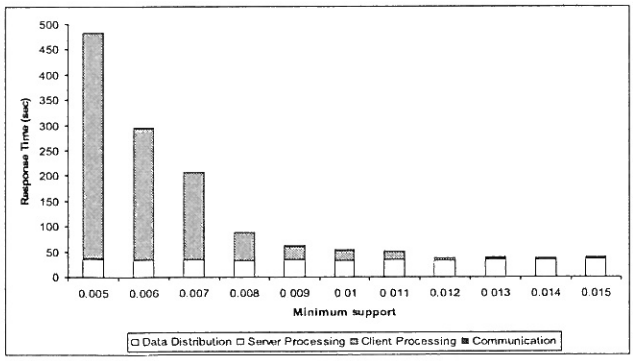

Fig. 8. Components of the response time in case of 1 worker node

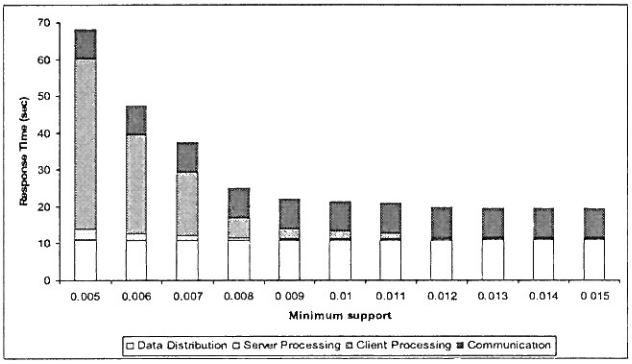

Fig. 9. Components of the response time in case of 10 worker nodes

This Figure shows that this cost can be approximated by the following form:

$$\frac{C_0}{p - C_1}, \tag{1}$$

where p is the number of nodes, C_0 and C_1 are constant values that depend on the communication channel. The scale up capability of these costs can be explained by that the worker nodes make data processing during data distribution phase.

C. Processing time of the coordinator node

The coordinator node processing costs are introduced in Figure 11 and in Figure 12. These figures show that processing time of the coordinator node can be predicted in the following form:

$$(C_0 \cdot p + C_1) \cdot \frac{1}{s^3 - C_2}, \tag{2}$$

where p is the number of nodes, s is the minimum support and the constant parameters depend on the data source. Although usually these costs are not significant part of the response time.

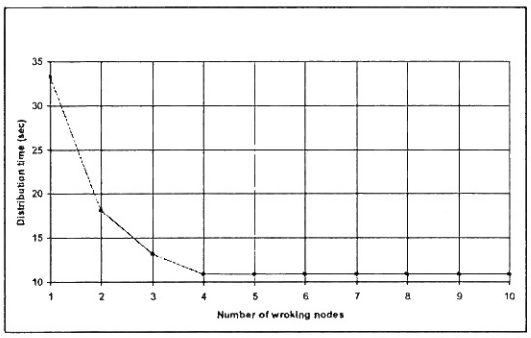

Fig. 10. Initial data distribution costs in function of working nodes

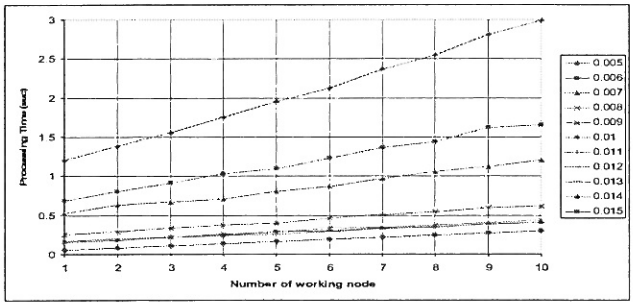

Fig. 11. Coordinator node processing time in function of worker nodes

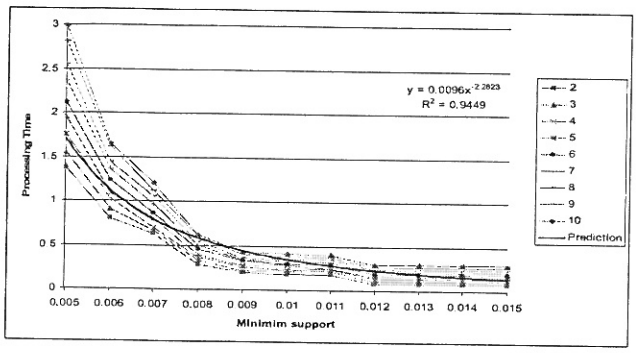

Fig. 12. Coordinator node processing time in function of minimum support threshold

D. Processing time of the worker nodes

Basically substantial part of CWD algorithm response time is the processing time of the worker nodes. But this cost can be pushed into the background in case of higher number of nodes and in case of higher minimum support values as it is described on Figure 9. Therefore it is important to investigate the scale up capability of this component of the response time. Figure 13 and 14 describe the worker nodes processing time as a function of worker nodes and as a function of minimum support. These diagrams show that the processing time can be predicted by the following way:

$$C_0 \frac{1}{s^5 - C_1} \frac{1}{p - C_2}, \qquad (3)$$

where s is the minimum support threshold, p is the number of worker nodes and the constant values depend on the dataset. Equation 3 shows that this part of the response time decreases very fast in function of minimum support. For that reason the response time of the algorithm is become nearly constant in function of working node in case of higher support values.

E. Communication costs

The additional communication costs of the CWD algorithm can become significant part of the response time in case of higher number of nodes and in case of higher minimum support as described on Figure 9. Figure 15 and Figure 16 show the communication costs of the CWD algorithm. These figures show that the communication costs are independent of the minimum support threshold and they can be approximated by a simple linear function of the number of working nodes, where the const parameters depend on the dataset.

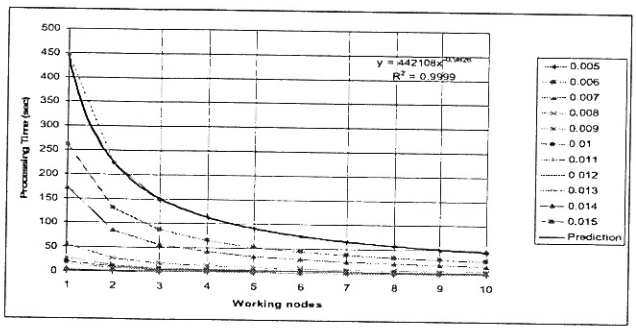

Fig. 13. Worker node processing time in function of worker nodes

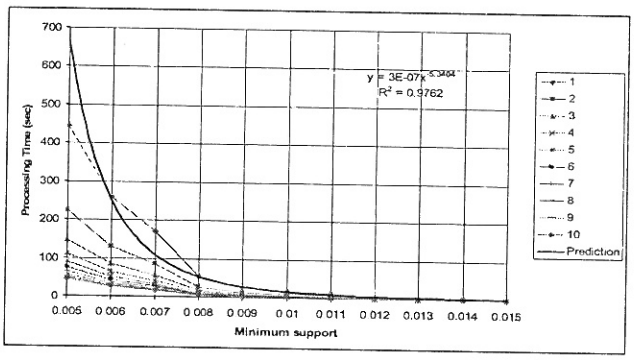

Fig. 14. Worker node processing time in function of minimum support threshold

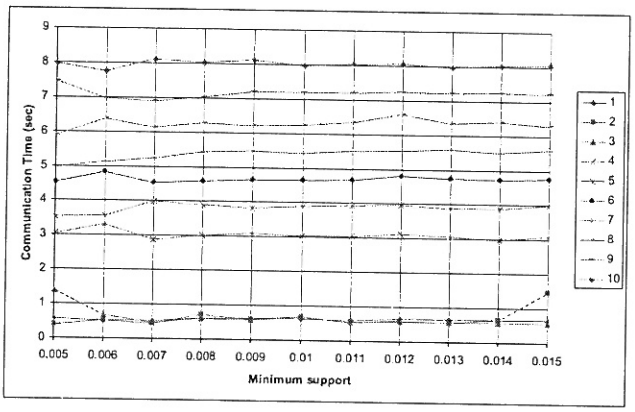

Fig. 15. Additional communication costs in function of minimum support threshold

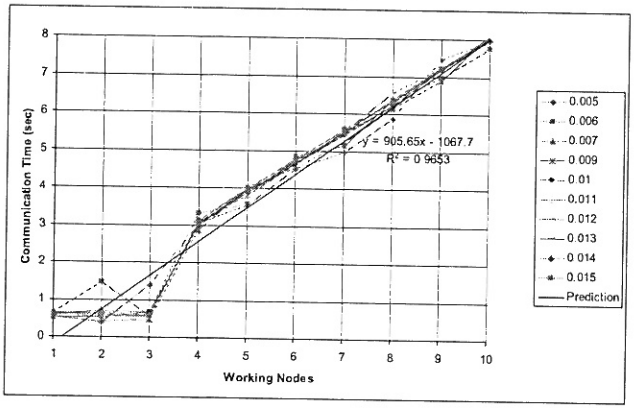

Fig. 16. Additional communication costs in function of worker nodes

VII CONCLUSION

In this paper a detailed analysis of the CWD algorithm is introduced. During the analysis the main components of the algorithm execution time were identified:

1. Initial data distribution
2. Costs of the processing coordination
3. Itemset counting costs
4. Additional communication costs

The significant parts of the response time and the scale up capability depend on the minimum support threshold and on the number of worker nodes. In case of low support values the determinant part of the execution time is the processing time of the worker nodes. Hence the scale up capability of algorithm is higher. But in case of higher support threshold (approximately 1%) the communication

costs and initial data distribution are the main part of the execution time. Therefore in this situation CWD algorithm has worse scale up possibility, the response time is become nearly constant very fast.

VIII. ACKNOWLEDGMENTS

This work has been supported by IBM Hungary and by the fund of the Hungarian Academy of Sciences for control research and the Hungarian National Research Fund (grant number: T042741).

IX. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", *in Proc. of ACM-SIGMOD Conference*, 1993

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", *in Proc. of 20th Very Large Databases Conference, 1994*

[3] J. Han J. Pei and Y. Yin, "Mining Frequent Patterns without Candidate Generation", *in Proc. of ACM SIGMOD International. Conference on Management of Data*, 2000

[4] R. Agrwal and J.C. Schafer, "Parallel mining of association rules", *in IEEE Trans. Knowledge and Data Engineering*, vol. 8, no 6, 1996

[5] O. R. Zaïne, M. El-Hajj and P. Lu, "Fast Parallel Association Rule Mining Without Candidacy Generation", *in Proc. of IEEE International Conference on Data Mining*, 2001

[6] Renáta Iváncsy, Ferenc Kovács and István Vajk, "An Analysis of Association Rule Mining Algorithms", *in Proc. of International Conference of ICSC EIS*, 2004

[7] Ferenc Kovács, Renáta Iváncsy and István Vajk, "Evaluation of the Serial Association Rule Mining Algorithms", *in Proc. of International Conference of IASTED DBA*, 2004

[8] M. Tamura and M. Kitsuregawa, "Dynamic load balancing for parallel association rule mining on heterogeneous PC cluster system", *in Proc. of 25th Very Large Databases Conference*, 1999

[9] T. Shintani and M. Kitsuregawa, "Hash based parallel algorithms for mining association rules", *in Proc. of Parallel and Distributed Information Systems Conference*, 1996

[10] T. Shimomura and S. Shibusawa, "Performance Evaluation of Distributed Algorithms for Mining Association Rules on Workstation Cluster", *in Proc. of IEEE International Workshops on Parallel Processing (ICPP'00- Workshops)*, 2000

[11] J. Zhang, H. Shi and L. Zheng, "A Method and Algorithm of Distributed Mining Association Rules in Synchronism", in *Proc. of IEEE International Conference on Machine Learning and Cybernetics*, 2002

[12] Ferenc Kovács, Renáta Iváncsy and István Vajk, "Dynamic Itemset Counting in PC Cluster Based Association Rule Mining", *in Proc. of International Conference of ICSC EIS*, 2004

[13] S.Brin, R. Motawani, J.D. Ullman and S. Tsur, "Dynamic Item set counting and implication rules for market basket data", *in Proc. of ACM-SIGMOD Conference*, 1997

[14] T. Shintani and M. Kitsuregawa: "Hash based parallel algorithms for mining association rules", *in Proc. of Parallel and Distributed Information Systems Conference*, 1996

[15] S. Juhász F.Kovács, "A New PC Cluster Based Distributed Association Rule Mining", *in Proc. of International Conference on Technical Informatics (CONTI 2004)*, 2004

[16] S. Juhász et al., *"The Pyramid Project"*, Budapest University of Technology and Economics, Department of Automation and Applied Informatics, 2002. http://avalon.aut.bme.hu/~sanyo/piramis

[17] S. Juhász, R. Iváncsy and I. Vajk, "Performance Modelling of the Apriori Association Rule Mining Algorithm", *in Proc. of microCAD International Conference*, 2004