Performance Evaluation of Distributed Algorithms for Mining Association Rules on Workstation Cluster

Tomohiro Shimomura and Susumu Shibusawa
Department of Computer and Information Sciences
Faculty of Engineering, Ibaraki University
st96038@satsuki.cis.ibaraki.ac.jp, sibusawa@cis.ibaraki.ac.jp

Abstract

The mining of association rules is one of the database mining techniques used to extract useful information from large quantities of data. Finding association rules, however, requires that the transaction database be scanned repeatedly, and we need to handle very large amounts of transaction data. This requires an incredibly large amount of computation time. There have therefore been many attempts to speed-up database mining by using parallel computers. Recent improvements in the performance of PCs and workstations (WSs), and the recent dissemination of network technology have made parallel processing distributed within computer clusters an attractive alternative to parallel computers.

In this paper we describe two new algorithms effective for parallel processing distributed in a WS cluster environment. One is an algorithm in which the size of data transmitted between WSs is smaller than that of the former algorithm. The other is an algorithm that reduces the number of scan processings at each node by dividing data and that uses shift operations for data communication. We have implemented these algorithms on a WS cluster and have evaluated their performance.

1. Introduction

The amount of data accumulated by enterprises and individuals has recently become enormous. This is a result of the development of such computer technologies as microprocessors, storage devices and computer networks. Database mining is one of the methods used to search large databases for useful information, such as rules or previously unknown relations between data items. One of the most important fields in database mining is the discovery of association rules inferred from transaction databases. An association rule implies certain association relationships among sets of items, and several algorithms for the mining of association rules have been proposed[1, 2, 3]. Some of them are parallel algorithms used to process large quantities of data efficiently by distributing the processing among the nodes of parallel computers [4, 5, 6].

On the other hand, the performance of the processors in PCs and WSs has advanced rapidly and the prices of PCs and WSs has fallen. Furthermore, the dissemination of network technologies has also reduced the cost of using high-performance networks. As a result parallel processing distributed among PC and WS clusters has become an attractive alternative to parallel computers.

This paper describes new distributed algorithms effective on WS clusters and evaluate their performance. One is the Count Communication (CC) algorithm that transfers small amounts of data between nodes and uses broadcast operations. The other is the Shift algorithm, which reduces the number of scan processings needed at each node by dividing data and which uses shift operations for data communication. We also implemented these algorithms on an SR2201 distributed-memory parallel computer and compared their communication cost and search cost in that environment with the corresponding costs in the cluster environment.

The rest of the paper is organized as follows. Section 2 gives an overview of the mining of association rules and of the Apriori algorithm on which our parallel algorithms are based. Section 3 describes an earlier nonpartitioned Apriori algorithm and also describes the CC and Shift algorithms. Section 4 describes the experimental environment, and Section 5 presents the results of the performance measurements. Section 6 discusses the experimental results, and Section 7 concludes by very briefly summarizing its results and mentioning some future work we intend to do ourselves.

0-7695-0771-9/00 $10.00 © 2000 IEEE

2. The mining of association rules

Association rules show relationships between items or itemsets. For example, consider the following rule: "90% of customers who purchase bread and butter also purchase milk." This association rule may be used for actually digging up the combination pattern of purchase items through an analysis of POS (point-of-sales) systems at retail stores. The results of such an analysis can be used when planning advertisements, the layout of items at various stores, and so on. Since association rules cannot be found without scanning the transaction database repeatedly, the mining of association rules incurs a very high processing load and a large execution time.

2.1. Association rules

We first review some basic concepts of association rules here. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and let $D = \{t_1, t_2, \ldots, t_n\}$ be a set of transactions, each of which consists of a set of items such that $T \subseteq I$. We say each transaction T contains a set of items X if $X \subseteq I$. The itemset X has support s ($s = support(X)$) in the transaction set D if $s\%$ of transactions in D contain X. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I, Y \subset I$ and $X \cap Y = \phi$. Each rule has two measures of value, support and confidence. The support of 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. That is,

$$confidence(X \Rightarrow Y) \equiv \frac{support(X \cup Y)}{support(X)} \times 100. \quad (1)$$

When mining association rules, we need to find all the rules that satisfy the user-specified minimum support and minimum confidence. This can be decomposed into two subproblems:

1. Find all sets of items (itemsets) whose support is greater than the user-specified minimum support. Itemsets which satisfy the minimum support are called large itemsets.
2. Generate from these itemsets the association rules whose confidence is greater than the user-specified minimum confidence.

2.2. Apriori algorithm

One of the algorithms most widely in the mining of association rules is the Apriori algorithm for finding all large itemsets [1]. The first pass of the algorithm simply counts the number of item occurrences to determine the large itemsets having one item (called large 1-itemsets).

A subsequent pass, say pass k, consists of two phases. First the large itemsets found in the $(k-1)$th pass are used to generate candidate itemsets produced by combining of the large $(k-1)$-itemsets. Then the database is scanned to determine the support of these candidate itemsets. For all passes $k > 1$, the algorithm works as follows:

1. Generate the candidate k-itemsets using the large $(k-1)$-itemsets created at the end of pass $k-1$.
2. Scan the transaction database and count the number of occurrences of the candidate k-itemsets, and compute support.
3. Let the large k-itemsets be candidate k-itemsets which satisfy the minimum support.
4. The procedure terminates if the large itemset is empty. Otherwise $k := k+1$ and go to step 1.

3. Parallel algorithms

3.1. Former parallel algorithms

Several researchers have proposed parallel algorithms which execute the Apriori algorithm shown in Section 2.2 [4, 5, 6]. Assuming that transactions are evenly partitioned into each node in advance, these parallel algorithms use the following four steps to find the large itemsets.

1. Each node generates the candidate k-itemsets by using the large $(k-1)$-itemsets created at the end of pass $k-1$ and copied into each node.
2. Each node counts the number of occurrences of the candidate k-itemsets using its local transactions.
3. All node's local counts are gathered, merged, and checked to determine whether or not the minimum support condition is satisfied.
4. If large k-itemset is empty, the algorithm terminates. Otherwise $k := k+1$, the large itemsets are copied into each node, and go to step "1."

Next, we introduce the earlier non-partitioned Apriori (NPA) algorithm which uses the notation in Table 1.

In this algorithm the candidate itemsets are copied over all the nodes. Each node counts the number of

Table 1. Notation.

k-itemset	An itemset having k items.
L_k	Set of all large k-itemsets.
C_k	Set of all candidate k-itemsets.
C_k^j	The local candidate k-itemsets of the j-th node.
D^j	The local transaction datasets of the j-th node.
N_k^j	The number of occurrences of the candidate k-itemsets at the j-th node.

occurrences of itemsets using its local transactions, and one node gets global counts and the large itemsets by merging local counts. The processing of pass k is as follows:

1. Each node generates the candidate k-itemsets C_k by using the large $(k-1)$-itemsets $L_{(k-1)}$ created at the end of pass $k-1$ and copied into each node.
2. Each node counts the number N_k of occurrences of the candidate k-itemsets by scanning the local transactions.
3. All node's local counts are gathered, merged, and checked to determine whether or not the minimum support condition is satisfied. Those for which the condition is satisfied are large k-itemsets L_k.
4. If large k-itemset L_k is empty, the algorithm terminates. Otherwise $k := k+1$, the large itemsets are copied into each node, and go to step "1."

A problem with the algorithm is that the number of scans increases when there are many candidate itemsets.

3.2. Algorithms for cluster-based parallel distributed systems

Since algorithms mining association rules process the mining of large itemsets from candidate itemsets, they need to refer to large itemsets of the previous step frequently. Therefore the search cost becomes very large. And when this processing is executed in parallel, the communication cost becomes very large. The former NPA parallel algorithm reduces the communication cost by copying all of the large itemsets of the previous step into each node. Then each node executes the same operation individually and generates candidate itemsets locally.

These algorithms are applicable to parallel computers whose communication between processors is fast. Communication delay becomes a problem when they are executed on cluster-based parallel distributed systems where nodes are PCs or WSs connected via Ethernet. We developed two algorithms to reduce the communication cost and evaluated them on a cluster-based environment.

3.2.1. Count communication algorithm. The Count Communication(CC) algorithm does not broadcast the large itemsets to all the nodes but instead broadcasts the global counts because data size is smaller. Each node has the large itemsets which satisfy user-specified minimum support. The processing of pass k in CC algorithm is as follows.

1. Each node generates the candidate k-itemsets C_k by using the large $(k-1)$-itemsets $L_{(k-1)}$ created at the end of pass $k-1$ and copied into each node.
2. Each node counts the number N_k^j of occurrences of the candidate k-itemsets using its local transactions.
3. All node's local counts are gathered and merged at one node, and the global count N_k is broadcasted to all the other nodes.
4. Each node checks to determine whether or not the candidate k-itemsets in C_k satisfy the user-specified minimum support.
5. If large k-itemset L_k is empty, the algorithm terminates. Otherwise $k := k+1$ and go to step "1."

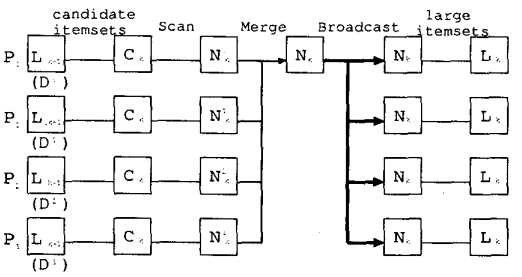

Figure 1. CC Algorithm.

3.2.2. Shift algorithm. When transmitting transaction databases, Shift algorithm does not execute broadcast operations but instead does shift operations. The processing of pass k in Shift is as follows:

1. Each node generates the candidate k-itemsets C_k^j by using the large $(k-1)$-itemsets $L_{(k-1)}$ created at the end of pass $k-1$, and the candidate k-itemsets C_k are partitioned into each node.

2. Each node counts the number N_k^j of occurrences of the candidate k-itemsets using its local transactions, and shifts the transaction database D^j to the next node. The transactions transmitted from other nodes are executed similarly.

3. After scanning all the transaction data, each node determine individually whether or not the candidate k-itemsets C_k^j satisfy the user-specified minimum support.

4. If large k-itemset L_k is empty, the algorithm terminates. Otherwise $k := k+1$ and the large itemsets are copied into each node and go to step "1."

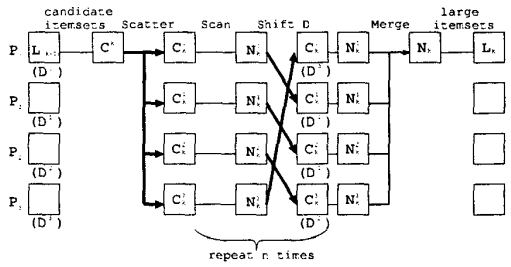

Figure 2. Shift Algorithm.

3.3. Cost analysis

This section analyzes the communication cost and the search cost for the NPA, CC, and Shift algorithms.

3.3.1. Communication cost analysis. The communication cost is the total amount of data which are transmitted among nodes in one pass of the mining of association rules.

NPA Algorithm:

This algorithm incurs communication costs when counts are merged and when the large itemsets are broadcasted. The amount of communication M_k^{NPA} in pass k is as follows:

$$M_k^{NPA} = LAR_k \times (P-1) + n_k \qquad (2)$$

This equation and the following equations in this subsection use the variables listed in Table 2.

Table 2. Variables.

variable	interpretation
LAR	Size of large itemsets
CAN	Size of candidate itemsets
n	Size of counts
D	Number of transactions
P	Number of nodes

CC Algorithm:

This algorithm broadcasts only the counts. Therefore, the amount of communication M_k^{CC} in pass k is as follows:

$$\begin{aligned} M_k^{CC} &= n_k \times (P-1) + n_k \\ &= n_k \times P \end{aligned} \qquad (3)$$

Shift Algorithm:

This algorithm transmits data when shifting all transaction data and merging counts. The amount of communication M_k^{Shift} in pass k is as follows:

$$M_k^{Shift} = D \times (P-1) + n_k \qquad (4)$$

Since the count size is less than the size of large itemsets or the size of all transactions, the amount of communication M_k^{CC} is smaller than M_k^{NPA} and M_k^{Shift}.

3.3.2.Search cost analysis. The search cost is the total amount of data which are searched at all nodes in one pass of the mining of association rules. Parallel algorithms generate candidate itemsets using large itemsets, search the candidate itemsets in transaction databases, and count the number of candidate itemsets.

NPA Algorithm:

In this algorithm each node computes the all candidate itemsets, and one node generates the large itemsets. The search cost S_k^{NPA} in pass k is as follows:

$$S_k^{NPA} = CAN_k \times P + LAR_k \qquad (5)$$

CC Algorithm:

In this algorithm each node computes the all candidate itemsets, and all nodes generate the large itemsets. The search cost S_k^{CC} in pass k is as follows:

$$\begin{aligned} S_k^{CC} &= CAN_k \times P + LAR_k \times P \\ &= (CAN_k + LAR_k) \times P \end{aligned} \qquad (6)$$

Shift Algorithm:

This algorithm computes the candidate itemsets partitioned into each node, and one node generates large itemsets. The search cost S_k^{Shift} in pass k is as follows:

$$S_k^{Shift} = \frac{CAN_k}{P} \times P + LAR_k = CAN_k + LAR_k \quad (7)$$

The search cost S_k^{Shift} is smaller than S_k^{NPA} and S_k^{CC}. And because each node computes the all candidate itemsets and all nodes generate the large itemsets in CC algorithm, S_k^{CC} is greater than either S_k^{NPA} or S_k^{Shift}. Since each node executes the search processing at almost the same time, the search cost does not show the computation time of each algorithm.

4. Experiments

4.1. Experiment environments

We implemented the NPA, CC, and Shift algorithms on a WS cluster and on the parallel computer SR2201.

WS cluster:

Our WS cluster comprised thirty-two 500MHz Alpha 21164 WSs connected via a 100Mbps Ethernet LAN.

Parallel computer SR2201:

The SR2201 is a distributed-memory parallel computer which nodes are connected via high-speed three-dimensional crossbar switch networks. We used a SR2201 with 16 nodes.

4.2. Programming environments

We implemented the algorithms by using C-based MPI (Message Passing Interface) programming libraries. The MPI was designed for efficient and reliable communication and it can be implemented on many platforms.

4.3. Execution of parallel algorithms

We used transaction datasets comprising 10000 transactions, the average length of the transactions was 5 and the number of items was 50. With the similarity of POS ID, we used natural numbers as items of transactions. The datasets were partitioned into each node.

In the experiments reported in this paper, we varied the number of transactions, the minimum support, and the number of nodes. The experiment was executed 10 times for a pair of fixed values of the number of transactions, the minimum support and the number of nodes, and the minimum execution time was reported.

5. Performance evaluation

5.1. Relation between execution time and minimum support

In Figure 3 the execution time of the three algorithms is plotted against the minimum support. These results were obtained on the WS cluster when the number of nodes was 8 and the number of transactions was 5000. It shows that when the minimum support is

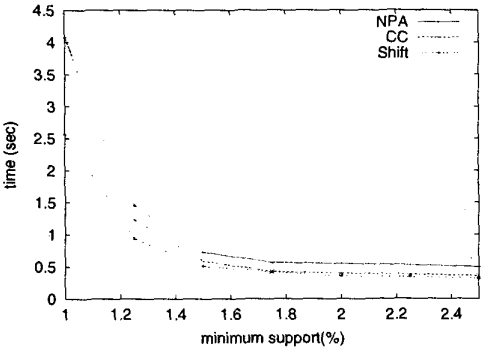

Figure 3. The execution time vs. minimum support.

small, Shift algorithm is superior to NPA and CC algorithms. It also shows that when the minimum support is small, the execution times for NPA and CC rapidly increases. On the other hand, the execution time for Shift does not increase as rapidly as do the execution times for NPA and CC.

5.2. Relation between execution time and the number of transactions

The execution time and the communication time for three algorithms are plotted against the number of transactions in Figure 4. These results were obtained on the WS cluster when the number of nodes were 8 and the minimum support was 1%. This figure also contains the exection time of a sequential algorithm. The execution time for each parallel algorithm is plotted in Figure 5 as a percentage of the execution time for the corresponding sequential algorithm. Figure 4

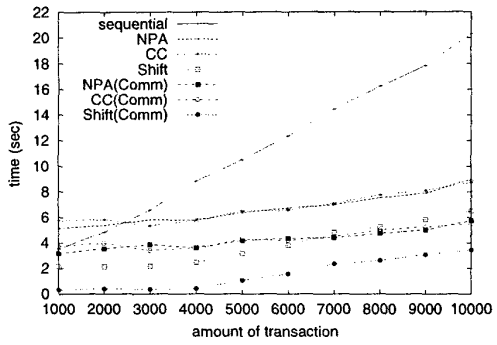

Figure 4. The execution time and the communication time vs. the number of transactions.

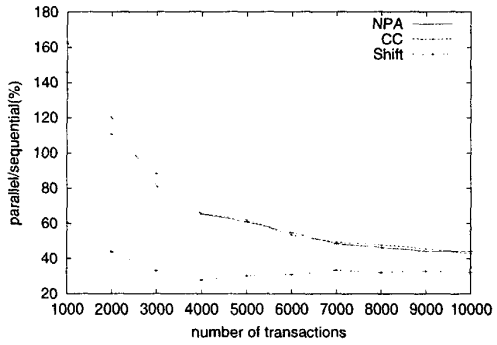

Figure 5. The ratio of the execution time for each parallel algorithm to the execution time for corresponding sequential algorithm.

shows that the execution time and the communication time for all algorithms increase in proportion to the number of transactions. The execution time for the Shift is shorter than those for the other algorithms. The advantage of parallelization is evident when the number of transactions is more than 2500. Figure 5 shows that the ratio of the execution time of a parallel algorithm to that of a sequential algorithm decreases as the number of transactions increase.

5.3. Relation between execution time and the number of nodes

The execution time and the communication time of three parallel algorithms on the WS cluster when the number of transactions was 5000 and the minimum support was 1% are plotted in Figure 6 against the number of nodes. For all algorithms the execution time

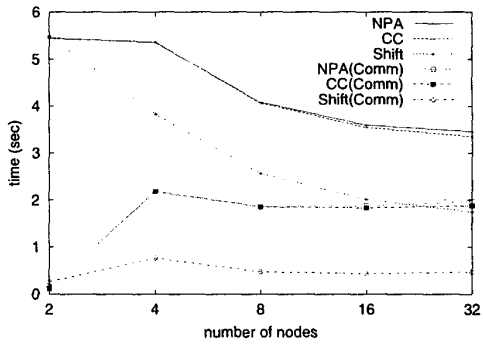

Figure 6. The execution time and the communication time vs. the number of nodes.

decreases when the number of nodes increases, whereas the communication time increases when the number of nodes increases. The performance of Shift algorithm is the best, and there is little difference between the performances of the NPA and CC algorithms.

5.4. Results on the parallel computer

The execution time and the communication time of the three algorithms running on the parallel computer when the number of transactions was 5000 and the minimum support was 1% is plotted in Figure 7 against the number of nodes. The communication times for three algorithms running on the WS cluster and on the parallel computer are listed, along with the numbers of nodes, in Table 3.

Table 3. Communication time. (sec)

	WS cluster		
Number of nodes	NPA	CC	Shift
2	0.148	0.117	0.258
4	2.180	2.180	0.766
8	1.855	1.855	0.469
	SR2201		
Number of nodes	NPA	CC	Shift
2	0.369	0.356	0.889
4	0.385	0.385	1.114
8	0.378	0.378	0.842

Table 3 shows that the communication times for NPA and CC algorithms are shorter on the parallel computer than they are in the WS cluster environment.

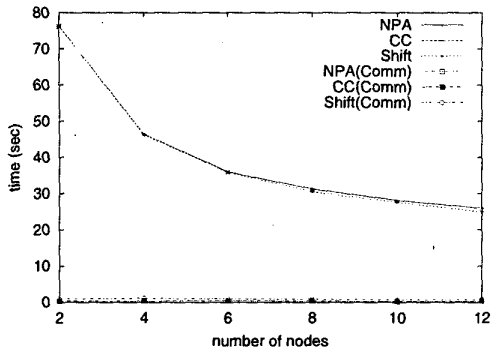

Figure 7. The execution time and the communication time on the SR2201.

On the other hand, the communication time for Shift algorithms is shorter on the WS cluster than that on the SR2201.

6. Discussions

6.1. Execution on a workstation cluster

The execution times for NPA and CC algorithms increase rapidly with a decrease in the minimum support. This is because the number of candidate itemsets increases when the minimum support decreases, and these algorithms scan the all candidate itemsets in the transaction database at each node. But because the Shift algorithm partitions the candidate itemsets into each node, the execution time of scanning is shorter than that for the other algorithms, and does not increase as rapidly when the minimum support decreases.

The execution times for all algorithms increase with an increase in the number of transactions, and the ratio of the execution time of each parallel algorithm to the execution time of the corresponding sequential algorithm decreases with an increase in the number of transactions. This ratio for the Shift algorithm reaches a minimum value when the number of transactions is 4000, and the algorithm converges more rapidly than the other algorithms.

The execution times for all algorithms decreases with an increase in the number of nodes. The reduction of the communication time in Shift operations seems to be due to the scanning processing and communication processing being executed asynchronously. Because data are searched in parallel and all amounts of searched data are almost the same for all these algorithms, CPU processing time which is the difference of the communication time from the execution time is almost the same for all algorithms.

Our cost analysis showed that the amount of communication was smallest for the CC algorithms, but the execution times for NPA and CC algorithms were almost the same in our experiments. This seems to be because that the size of data used in our experiment is not so large.

6.2. Execution on the parallel computer

The execution times for all algorithms were much longer on the parallel computer than that were in the WS cluster environment. We think this is because amount of CPU memory available on the parallel computer was insufficient. On the other hand, the communication time was more stable on the parallel computer, and the time for communication between nodes is shorter on the parallel computer. In other words, the ratio of the communication time to the execution time is large in a WS cluster environment, and the communication time has a great influence on the execution time.

In the WS cluster environment, the communication time for the Shift algorithm, which uses shift operations, is less than that for the CC algorithm, which uses broadcast operations. On the other hand, the communication time for the Shift algorithm on the parallel computer is larger than that for the CC algorithm. The Shift algorithm is therefore effective in a WS cluster environment that can execute shift operations rapidly.

7. Conclusion

The distributed algorithms proposed in this paper are effective when parallel processing distributed throughout clustered computers is used to mine databases for association rules. When we implemented these algorithms on a WS cluster and on a parallel computer so that we could evaluate their performance, we found that the Shift algorithm was the most effective when there was a large number of candidate itemsets and processor nodes in the WS cluster environment. This is because the ratio of communication time to execution time is large in a WS cluster environment, and the communication time therefore has a great influence on the execution time.

We intend to perform more analysis about the communication cost between nodes. We also intend to perform further experiments with large size data which are used at companies and institutes. We also intend to develop a new algorithm to share the loads among nodes because real data are often distributed unevenly.

References

[1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. of Int'l. Conf. on Very Large Data Bases*, pp.487-499, 1994.

[2] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proc. of Int'l. Conf. on Very Large Data Bases*, pp.407-419, 1995.

[3] N Megiddo and R. Srikant, "Discovering Predictive Association Rules," *Proc. of the 4th Int'l. Conf. on Knowledge Discovery in Databeses and Data Mining*, 1998.

[4] E.H. Han, G. Karypis and V. Kumar, "Scalable parallel data mining for association rules," *Proc. of ACM SIGMOD Int'l. Conf.*, pp.277-288, 1997.

[5] T. Shintani and M. Kitsuregawa, "Implementation of Parallel Mining Association Rules and their Evaluation," *JSPP'96*, pp.97-104, June 1996.

[6] L. Harada, N. Akaboshi, K. Ogihara and R. Take, "Parallel Algorithm with Load Balancing for Mining Association Rules," *IEICE Trans. on Info. and Syst.*, Vol.J82-D-1, No.1, pp.70-81, January 1999.