December 2005

Parallel Association Rule Mining by Data De-Clustering to Support Grid Computing

Frank Tseng
National Kaohsiung First University of Science and Technology

Pey-Yen Chen
National Kaohsiung First University of Science and Technology

Parallel Association Rule Mining by Data De-Clustering to Support Grid Computing*

Frank S.C. Tseng† and Pey-Yen Chen

Dept. of Information Management
National Kaohsiung First University of Science and Technology
1, University Road, YenChao, Kaohsiung County, Taiwan, 824 R.O.C.
imfrank@ccms.nkfust.edu.tw

Abstract

Most of the association rule mining algorithms suffer from the time-consuming elaboration on finding all candidates that fit the subjective conditions. We believe the most effective way is to develop parallel algorithms to promote the performance. However, prior parallel architectures and algorithms suffer from overhead in inter-site communications or requiring large number of space to maintain the local support counts of a large number of candidate sets. In this paper, we propose a parallel approach, which absolutely eliminates the inter-site communication cost for the most influential Apriori algorithm or its variations. The merit makes our approach to be easily deployed in a grid computing environment. Our work is based on the idea of data de-clustering, such that the transaction database is de-clustered into partitions for all participating sites. That guarantees all subgroups are not only quite similar to each other, but also quite similar to the original group. To balance the workload of the most time-consuming subtasks (i.e., the candidate itemsets generation process) of all participating sites, elements in the frequent 1-itemset are dispatched in row-prime order to each processor to execute in parallel. We have conducted experiments to show that the result obtained by our approach is almost the same as that obtained by running the Apriori algorithm on a single site. However, if there are m processors executed in our parallel approach, then the total speed up can be promoted up to m^2, which makes our work a very efficient and effective approach.

Keywords: Association Rule Mining, De-Clustering, Minimum Spanning Tree, Shortest Spanning Path.

1. Introduction

Thanks to the progress of the Internet technology, most of the enterprise databases can collect huge amounts of data easily from daily transactions. The volume is expected to grow considerably in the future. Hence, to effectively utilize these data, an attractive task is to derive valuable patterns or collect profitable insights from the data to guide enterprises' marketing strategies and management policies. That inspires data mining technologies being prolifically adopted in contemporary applications to explore versatile and previously untapped knowledge for supporting decision makers.

Among the various data mining approaches, the association rule generation from vast

* This Research was supported by the National Science Council, Taiwan, ROC, under contract no. NSC 93-2416-H-327-007.

† To whom all correspondence should be sent. Tel: +886-7-6011000 Ext. 4113, Fax: +886-7-7659541

amounts of business transactional data has been extensively studied in this decade (Aggarwal and Yu 2001; 2002; 2003) and gained considerable prominence in the database community (Aggarwal *et al*. 2002; Agrawal and Srikant 1994; Brin *et al*. 1997; Han and Fu 1995; Rastogi and Shim 2002). However, owing to the huge target data volume, most of the algorithms suffer from the elaboration on finding all candidates that fit the subjective conditions, which is a process prone to time-consuming. Therefore, we believe the most effective way is to develop parallel algorithms to promote the performance.

Although some parallel architectures and algorithms have been proposed, we found there are still some overhead exists in these approaches, which motivates our work. In this paper, we intend to develop a parallel algorithm that de-clusters the problem into subtasks and dispatches them to all participating sites. The purpose is to absolutely eliminate inter-site communications after distributing all subtasks. That is, communications occur only at the very beginning and ending of the whole process. As a result, there is no communication cost during the itemset generation process, which makes our framework a very efficient and effective approach.

The idea of our approach is based on the concept of data de-clustering adopted from (Fang *et al.* 1986). Based on this approach, all the transaction data will be de-clustered into subgroups, such that all subgroups are not only quite similar to each others, but also quite similar to the original group. Then, the subgroups of itemset generation will be assigned to the participating sites by initially assigning elements in the frequent 1-itemset in row-prime order (Samet 1989). By such task assignment, the load of each site can be highly balanced and the frequent itemsets of each site can be quickly obtained independently. Finally, the complete frequent itemsets can be obtained by directly collecting all the locally obtained frequent itemsets. Although it seems that our approach may lose some of the frequent itemsets, we have conducted an empirical study on our approach and found that the obtained result is almost exactly the same as that obtained from *Apriori*. However, if there are m sites participating in the system, then the total task can be achieved in nearly $1/m^2$ of the original time complexity.

This paper is organized as follows. Section 2 surveys some related works. Section 3 addresses our approach in details. Section 4 presents an empirical study of our work. Finally, Section 5 concludes our work and proposes some future directions.

2. Related Works

2.1 Association Rule Mining

The basic problem of finding association rules as introduced by Argawal *et al.* (1993) is as follows. Let $\tau = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$ be a set of literals, called *items*. Let D be a set of

transactions, where each transaction T is an itemset such that $T \subseteq \tau$. We say that a transaction T contains X, a set of some items in τ, if $X \subseteq T$. An *association rule* is an implication of the form $X \Rightarrow Y$, where $X \subset \tau$, $Y \subset \tau$, and $X \cap Y = \varnothing$. The rule $X \Rightarrow Y$ has *support s* in the transaction set D if s% of transactions in D contains $X \cup Y$. This is taken to be the probability, $P(X \cup Y)$. The rule $X \Rightarrow Y$ holds in the transaction set D with *confidence c* if c% of transactions in D that contain X also contain Y. This is taken to be the conditional probability, $P(Y \mid X)$.

Such problem can be decomposed into the following sub-problems:

1. First, all sets of items that are contained in a number of transactions above the minimum support requirement are iteratively identified and referred to as *frequent itemsets*.
2. Then, the desired association rules can be generated in a straightforward manner based on the finally obtained frequent itemsets.

Much of the research has been focused on the first sub-problem, which can be reduced to finding *frequent itemsets* with respect to a given *support threshold* (Agrawal *et al.* 1993; Cheung and Xiao 1999). Our work also focuses on finding all frequent itemsets in parallel.

We follow the notations used in Agrawal and Shafer (1996) to illustrate our approach, where we use superscripts to indicate processor id and subscripts to indicate the pass number (and the size of the itemset) as illustrated in Table 2.1.

Table 2.1: Notations Used in the *Apriori* Algorithm.

Notation	Description
L_k	Set of frequent k-itemsets (those with minimum support threshold). Each member of this set has two fields: (a) itemset and (b) support count.
C_k	Set of candidate k-itemsets (potentially frequent itemsets). Each member of this set has two fields: (a) itemsets and (b) support count
s^i	The local support threshold of processor P^i.
P^i	Processor with id i.
D^i	The dataset local to the processor P^i.
L^i_k	Set of frequent k-itemsets (those with local minimum support threshold) in P^i. Each member of this set has two fields: (a) itemset and (b) support count.
C^i_k	The candidate set maintained with the Processor P^i during the k-th pass (there are k items in each candidate).

In our approach, a transaction T will be represented as a bit string p, called *pattern*, which is defined as follows.

Definition 2.1: For a transaction $T \subseteq \tau = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$, the *pattern* of T is defined as a bit string $p = a_{n-1}\, a_{n-2} \ldots a_0$, where $a_i = 0$ if $A_i \notin T$, and $a_i = 1$ if $A_i \in T$.

For a set of transactions $D = \{T_1, T_2, \ldots, T_n\}$, it can be represented as a set of patterns $G = \{p_1, p_2, \ldots, p_n\}$ by transforming all transactions into their corresponding patterns.

2.2 Parallel Association Rule Mining

The influential algorithm *Apriori* developed by Agrawal *et al.* (Agrawal *et al.* 1993) is simple and easy to implement. However, since the first step may need repeatedly scanning the database to generate exponential candidates for counting the support, there are so many disk I/O operations involved in the frequent itemset generation process, which becomes a severe bottleneck.

Owning to the costs of computer peripherals are usually getting cheaper, but the speed performs higher, some parallel algorithms have been proposed. For instance, Park *et al.* (1995) generalized their DHP algorithm into a parallel version called PDM (Parallel Data Mining), Cheung *et al.* (1996) proposed FDM (Fast Distributed Mining) method, and Agrawal and Shafer (1996) presents three parallel algorithms, namely Count Distribution, Data Distribution, and Candidate Distribution methods. Cheung *et al*. (2002) have pointed out that most of the research activities fall into one of these paradigms. For an earlier comprehensive survey, readers are referred to (Zaki 1999).

These approaches usually suffer from the inter-site communication overhead during the itemset generation process. When the data is skew, the performance may downgrade drastically. Therefore, the main objective of our work is to establish a parallel architecture for the *Apriori* method or its variations, and get rid of all the inter-site communications to maximize the speed of the frequent itemset generation process in parallel. Such merit supports a high-performance parallel and distributed grid computation over a wide-area network (Fox 2003).

2.3 Data De-Clustering

The concept of de-clustering is just the opposite effect of clustering analysis. By the definition of (Fang *et al.* 1986), for a set of data *S*, a *de-clustering* analysis divides *S* into *n* groups $G_1, G_2, \ldots, G_n$, such that

(1) All groups $G_1, G_2, \ldots, G_n$ will be quite similar to the original set *S*.

(2) $G_1, G_2, \ldots, G_n$ should be quite similar to each others.

In (Fang *et al.* 1986), two approaches have been proposed to de-cluster a set of data, namely the minimal spanning tree approach and the shortest spanning path approach. To explain these approaches easily, the original group will be divided into two subgroups, which are similar to each other. We briefly describe them as follows.

(1) Minimal Spanning Tree (MST) approach: For a set of transactions, construct a minimal spanning tree (Graham and Hell 1985; Kruskal 1956; Yao 1975) by regarding each transaction as a node, and finally alternatively separate the nodes in odd levels and even levels into two similar subgroups.

(2) Shortest Spanning Path (SSP) approach: For a set of transactions, a shortest spanning path (Lee 1981) is constructed with a unique ordering of nodes. Then, let the nodes be labeled according to their positions in the path. Finally we may put nodes with odd numbers into one subgroup and nodes with even numbers into another subgroup.

From the research result shown in (Fang *et al.* 1986), the MST approach is superior to the SSP approach in regarding the similarity between the original group and the obtained subgroups. However, the SSP approach divides the node number in the original group more evenly. The SSP approach guarantees that the difference between the numbers of nodes in subgroups is only at most one. Although finding an SSP is generally NP-hard (Papadimitriou and Steiglitz 1976), in this paper, we employ a polynomial approximate algorithm to construct a spanning path for a set of transaction $D = \{T_1, T_2, \ldots, T_n\}$ with the summation of the weight of the affinity of all the consecutive neighbors being nearly minimum. In Section 4, we will conduct experiments to show that the performance of de-clustering by our approximate SSP algorithm is superior to that by the MST approach.

3. Our Approach

The architecture of our approach is depicted in Figure 3.1. The architecture assumes a shared-nothing parallel processing environment, where each participant has a private memory and a private disk. The participants are connected by a communication network and can communicate only by message passing. The whole process can be explained by the following three stages:

(1) *Data Transformation*: In the first stage, the set of transactions $D = \{T_1, T_2, \ldots, T_n\}$ will be transformed and represented as a set of patterns $G = \{p_1, p_2, \ldots, p_n\}$, which will be used to generate the first global frequent 1-itemset L_1.

(2) *Data De-clustering*: Then, in the second stage, suppose there are m participants, then we may employ MST or SSP to de-cluster the set of transaction patterns $G = \{p_1, p_2, \ldots, p_n\}$ into m groups $G_1, G_2, \ldots, G_m$., which will be respectively copied from the original database to their corresponding participants.

(3) *Task Distribution*: Finally, without loss of generality, suppose $L_1 = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$, we then use row-prime order to assign these elements to the participants, which will be regarded as L^i_1, $i = 1, .., m$. For a participant P^i assigned with L^i_1, it will be supposed to be responsible for generating those itemsets with leading items belong to L^i_1. This is

illustrated by the dotted lines in Figure 3.1.

Such task assignment has a very important merit that the loading of all clients will be balanced and the communication among all clients is dispensable. That means the communication cost among all participants is zero, and all participants could be fully utilized to maximize the parallel processing performance. Therefore, the total data hold by each participant is only 1/*m* of the original data, if there are *m* participants executed in parallel. Besides, the workload of each participant is only 1/*m* of the total process. Theoretically, since there are *m* participants, the total speed up can be promoted up to m^2.

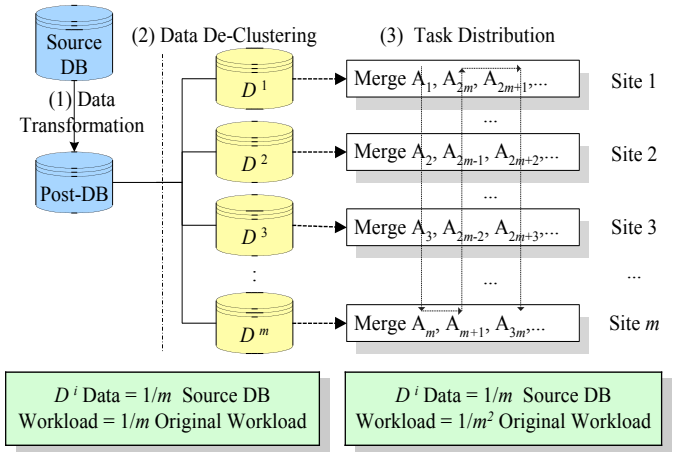

Fig. 3.1: The System Architecture

We further clarify these stages in the following subsections.

3.1 Data Transformation Process

To reduce the data size, all transactions will be transformed into their corresponding patterns, which are all represented as bit strings. For example, let $\tau = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$ be a set of items, suppose there are *k* transactions in $D = \{T_1, T_2, \ldots, T_k\}$, where $T_1 = \{A_1, A_2\}$, $T_2 = \{A_2, A_3, A_4, \ldots, A_n\}$, …, and $T_k = \{A_1, A_4, A_n\}$. The data transformation process will transform them into a binary sparse matrix as shown in Figure 3.2. Based on the matrix and a given support threshold *s*, we can obtain the first frequent 1-itemset L_1.

T_{id}	Items
T_1	$\{A_1, A_2\}$
T_2	$\{A_2, A_3, A_4, \ldots, A_n\}$
…	…
T_k	$\{A_1, A_4, A_n\}$

Original Transactions

⇒

T_{id}	A_1	A_2	A_3	A_4	…	A_n
T_1	1	1	0	0	…	0
T_2	0	1	1	1	…	1
…	…	…	…	…	…	…
T_k	1	0	0	1	…	1

Binary Sparse Matrix

Fig. 3.2: An Example to Illustrate Data Transformation Process

3.2 Data De-Clustering Process

To de-cluster data records, according to Liu *et al.* (1997), nominal attribute values

should be ordered and assigned with different weights to differentiate their differences. Therefore, we define the weight of an item and a transaction in Definitions 3.1 and 3.2, respectively.

Definition 3.1: For any item $A_i \subseteq \tau = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$, $0 \le i \le n-1$, the *weight* of A_i is defined as 2^i.

Definition 3.2: For a transaction T with pattern $p = a_{n-1}\ a_{n-2}\ \ldots\ a_0$, $T \subseteq \tau$, the *weight* of T, denoted $w(T)$, is defined as $\sum_{i=0}^{n-1} a_i \cdot 2^i$.

To compute the difference between two transactions, we define the affinity between two transactions and its corresponding weight in Definitions 3.3 and 3.4, respectively.

Definition 3.3: For two transactions T_i and $T_j \subseteq \tau$, the *affinity* between T_i and T_j, denoted $aff(T_i, T_j)$, is defined as a bit string $p_i \oplus p_j = b_{n-1}\ b_{n-2}\ \ldots\ b_0$, where $\oplus$ represents the bit-wise exclusive-or operation.

Definition 3.4: For the affinity between T_i and T_j, $aff(T_i, T_j) = b_{n-1}\ b_{n-2}\ \ldots\ b_0$, the *weight* of $aff(T_i, T_j)$, denoted $w(aff(T_i, T_j))$, is defined as $\sum_{i=0}^{n-1} b_i \cdot 2^i$.

Based on the work presented in (Liu and Setiono 1997) and the definitions presented above, the following situations can be assured:

1. *Uniqueness of any transaction weight*: For any two transactions T_i and T_j containing different items, it is guaranteed that $w(T_i) \neq w(T_j)$.

2. *Uniqueness of any affinity*: For any two transactions T_i and T_j containing different items, the weight of the affinity between T_i and T_j, i.e., $w(aff(T_i, T_j))$, will be unique.

In (Fang *et al.* 1986), there are two approaches to de-cluster a set of records into disjoint sets. We rewrite it for our approach, and summarize them as follows.

1. *Minimal Spanning Tree approach* (*MST*): For a set of transactions, derive the weight of the affinity between any two transactions, $w(aff(T_i, T_j))$. Then, based on the obtained result, construct a minimal spanning tree by regarding these $w(aff(T_i, T_j))$ as the edge weights, and finally separate the nodes in each level circularly into the desired number of similar subgroups.

2. *Shortest Spanning Path approach* (*SSP*): For a set of transactions, derive the weight of the affinity between any two transactions, $w(aff(T_i, T_j))$. Then, based on the obtained result, construct an approximate shortest spanning path by regarding these $w(aff(T_i, T_j))$ as the edge weights, and finally separate the nodes in each level

circularly into the desired number of similar subgroups.

3.3 Task Distribution Process

After the original set of transaction patterns G is de-clustered into m subgroups, say G_1, G_2, …, G_i, …, G_m, they will be respectively copied from the original database to the databases of the corresponding participants, say D^1, D^2, …, D^i, …, D^m. Besides, without loss of generality, by assuming $L_1 = \{A_{n-1}, A_{n-2}, ..., A_0\}$, we then use row-prime order to assign these elements to the participants. For a participant P^i assigned with elements A_j's, it will be supposed to be responsible for generating only those itemsets leaded by A_j's for C_k^i, for all $k > 1$, by scanning their local database D^i. Besides, if the *minimum support count* for the original problem is s, then processor P^i will use $s^i = s/m$ as the local *minimum support count*, for all $1 \le i \le m$, where m is the number of processors.

This assignment has a very important merit that the communication cost among all participants is zero, and all participants could be fully utilized to maximize the parallel processing performance. The merit is obtained by an important observation explained as follows.

Let $\tau = \{A_{n-1}, A_{n-2}, ..., A_0\}$ be a set of items. By the original *Apriori* algorithm, if $L_1 = \{A_{n-1}, A_{n-2}, ..., A_0\}$, and suppose the elements in L_1 are scanned from left to right to derive the next candidate itemset C_2, then $A_{n-1}, A_{n-2}, ..., A_0$ will be accessed *n-1, n-2, ..., 0* times respectively. Repeatedly, if $L_{k-1} = \{A_{l-1}, A_{l-2}, ..., A_0\}$, then $A_{l-1}, A_{l-2}, ..., A_0$ will be accessed *l-1, l-2, ..., 0* times respectively, to derive the next candidate itemset C_k. That means the load is quite unbalanced as Figure 3.3 illustrates. Therefore, if we dispatch these elements serially to all participants in a parallel processing environment, then the load of all clients is severely unbalanced and the computation cost will be time-consuming.

L_{k-1}	C_k
A_{l-1}	$A_{l-1} A_{l-2}$, $A_{l-1} A_{l-3}$, $A_{l-1} A_{l-4}$, ..., $A_1 A_0$
A_{l-2}	$A_{l-2} A_{l-3}$, $A_{l-2} A_{l-4}$, ..., $A_{l-2} A_0$
A_{l-3}	$A_{l-3} A_{l-4}$, ..., $A_{l-3} A_0$
A_{l-4}	..., $A_{l-4} A_0$
…	…
A_0	∅

Fig. 3.3: Unbalanced scanning of L_{k-1} to derive the next candidate itemset C_k.

By our approach, we assign the task to generate a next candidate itemset C_k by row-prime order to balance the load. If there are m participants executed in parallel, then the total data hold by each participant is only about $1/m$ of the original data and their loads are balanced as Figure 3.4 depicts. Besides, the workload of each participant is only $1/m$ of the total process. Theoretically, since there are m participants, the total speed up can be up to m^2.

Participants	Number of Task Assignments	Total Number of

	of L_1 Elements			Task Assignments
P^1	$n-1$	$n-2m$	…	$2n-2m-1$
P^2	$n-2$	$n-2m+1$	…	$2n-2m-1$
P^3	$n-3$	$n-2m+2$	…	$2n-2m-1$
…	…	…	…	$2n-2m-1$
P^m	$n-m$	$n-m-1$	…	$2n-2m-1$

Fig. 3.4: Load Balancing of Each Participant.

3.4 The Revised Algorithm

Based on the above discussions, we revise *Apriori* into a parallel version, such that it is executed in any participant P^i by scanning its local database D^i using the minimum support threshold $s^i = s/m$, where s is the original minimum support threshold, and m is the number of participants. All participants will execute independently. Finally, the complete frequent itemsets can be obtained directly by collecting all of the result derived in all participants.

As the elements in L_1 are partitioned and distributed to different processors (in row-prime order), the *Apriori property*—all nonempty subsets of a frequent itemset must also be frequent—used in each participant should be relaxed to avoid losing frequent itemsets when reducing the search space. We call the relaxed property as *Anti-Apriori property*—all nonempty subsets of a local frequent itemset is supposed to be frequent globally, but it may not be supposed to be frequent locally. This relaxed property is based on the following observation. During calculating a candidate itemset C^i_k for some $k > 2$ in processor P^i, by joining L^i_{k-1} with itself, an itemset l with $l \in L^i_{k-1}$, $l[1] \notin L^i_1$, and is not satisfying the local minimum support threshold s^i does not mean l is not frequent globally. This is because there may exist another processor P^j, which is responsible for generating l as a frequent itemset and $l[1] \in L^j_1$. Therefore, if an item A is added to the itemset l, then the resulting itemset (i.e., $l \cup A$) may be also as frequent as l from global point of view. Therefore, $l \cup A$ should not be discarded in C^i_k. Since D^i and D^j are two databases with similar characteristics (i.e., they have the same probability containing l), if l is not frequent globally (i.e., when scanning the database D by support threshold s), then it will not be expected frequent in D^i or D^j and will be discarded eventually.

Our parallel algorithm mainly revises the *join* and *prune* process of *Apriori* to find the result, which will be explained as follows. Note that we also assume items within a transaction or itemset are sorted into lexicographic order.

1. *The join step*: It is easily to realize that the result of the *join step* in the original *Apriori* algorithm in a single processor, which joins L_{k-1} with itself (i.e. $L_{k-1} \bowtie L_{k-1}$ for $k > 2$), should be preserved when these tasks are accomplished by multiple processors. If there are m processors, then it is straightforward to realize that $L_{k-1} \bowtie L_{k-1}$ can be decomposed into $(\cup_{1 \le i \le m} L^i_{k-1}) \bowtie L_{k-1} = (L^1_{k-1} \bowtie L_{k-1}) \cup (L^2_{k-1} \bowtie L_{k-1}) \cup \ldots \cup (L^m_{k-1} \bowtie$

L_{k-1}). To guarantee any possible frequent itemsets being properly handled, the join step will be distinguished into the following two cases.

(a) To find L^i_2, according to the decomposition formula, we perform $L^i_1 \square L_1$ to produce the candidate set C^i_2 in P^i, for all $1 \le i \le m$. This makes items in $L_1 - L^i_1$ be taken into account for generating a candidate set in any processor P^i. If they are discarded in the first place, then according to the *Apriori property*, none of its supersets will be included in the forthcoming candidate sets, which implies some frequent itemsets will be lost in later iterations. Let l_1 be an itemset in L^i_1 and l_2 be an itemset in L_1 (i.e., l_1 and l_2 both contain only one element), then they are joined if ($l_1[1] < l_2[1]$). The result 2-itemsets formed by joining l_1 and l_2 is $l_1[1]$ $l_2[1]$.

(b) To find L^i_k in P^i, for $k > 2$, a set of candidate k-itemsets, denoted C^i_k, is generated by $L^i_{k-1} \square L^i_{k-1}$. The process is absolutely analogous to the join step in the *Apriori* algorithm. Let l_1 and l_2 be itemsets in L^i_{k-1}. When the first (k-2) items in members l_1 and l_2 are in common, then they are joined if ($l_1[1] = l_2[1]$) $\wedge$ ($l_1[2] = l_2[2]$) $\wedge \ldots \wedge$ ($l_1[k\text{-}2] = l_2[k\text{-}2]$) $\wedge$ ($l_1[k\text{-}1] < l_2[k\text{-}1]$). The last condition $l_1[k\text{-}1] < l_2[k\text{-}1]$ simply ensures that no duplicates are generated. The result k-itemsets formed by joining l_1 and l_2 is $l_1[1]$ $l_1[2] \ldots$ $l_1[k\text{-}1]$ $l_2[k\text{-}1]$.

2. *The prune step*: All candidates in C^i_k having a count no less than s^i are locally frequent and therefore belong to L^i_k. For any (k-1)-subset of a candidate k-itemset not in L^i_{k-1}, the candidate cannot be frequent locally. However, according to the *Anti-Apriori property* discussed above, those itemsets l with $l \in L^i_{k-1}$, but $l[1] \notin L^i_1$ may be frequent globally and should be retained in C^i_k to avoid losing frequent itemsets.

4. Experimental Results

To verify and assess the effectiveness of our approach, we have implemented the algorithms and conducted some experiments on different sample databases. We use four IBM-compatible personal computers to run on Windows 2000 Server with Microsoft SQL Server 2000 as our database engine. The test data are generated from a program provided by IBM Data Mining Research Group (Intelligent Information Systems Research department) in IBM Almaden Research Center (http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html), which has been widely adopted by many data mining research works (Agrawal and Shafer 1996; Agrawal *et al.* 1993; Park *et al*. 1995; Savasere *et al*. 1995). The program simulates customer-purchasing behaviors and generates transactions data according to the Poisson distribution. The parameters we have used to generate the test databases are listed in Table 4.1. Based on these parameters, we have generated five different databases as Table 4.2 illustrates.

Table 4.1: The Parameters Used for Generating the Test Databases

Parameters	Description
| D |	Total number of transactions in the database
| T |	Average number of items per transaction (i.e., transaction length)
| I |	Average number of attributes in the candidate itemsets
| L |	The largest number of attributes in the candidate itemsets
N	Number of attributes in the database

Table 4.2: The Test Databases

Database Id	| D |	N	| T |	| I |
N30.T7.I5.D01K	100	30	7	5
N30.T7.I5.D05K	500	30	7	5
N30.T7.I5.D1K	1000	30	7	5
N30.T7.I5.D2K	2000	30	7	5
N30.T7.I5.D3K	3000	30	7	5

We have tested these databases to run the *Apriori* algorithm on a single site and obtained the result as shown in Figure 4.1. The times needed in testing different databases are raised exponentially when the sizes of the databases are increased.

Besides, the costs needed for constructing a minimal spanning tree (MST) and the shortest spanning path (SSP) for each test database are illustrated in Figure 4.2. The cost for constructing an MST using Kruskal's algorithm is dominated by the edge sorting process, which takes $O(m\log m)$ time complexity, where m is the number of edges in the graph. In the worst case, $m = n^2$, where n is the number of nodes in the graph. That is, it takes $O(n^2\log n)$ time complexity to generate an MST. In contrasts, the time complexity of our divide-and-conquer method takes only $O(n\log n)$ time.

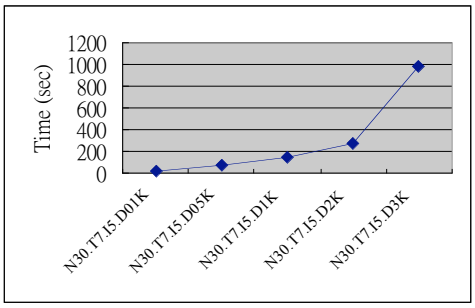

Fig. 4.1: The Single Site Test Result

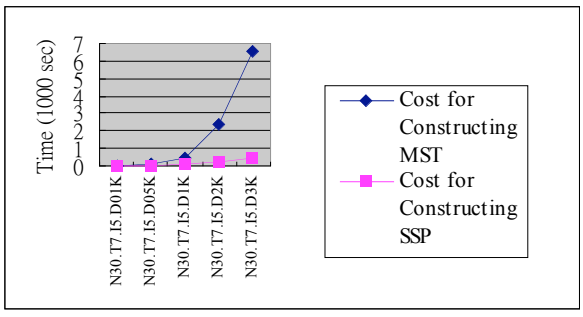

Fig. 4.2: The Costs for Constructing MST and SSP

Finally, after conducting the performance test of generating L_k in parallel, we obtain the result as Figure 4.3 depicts. Obviously, the performance can be greatly improved by multi-sites execution in parallel and there are only slightly differences among the cases using two, three, and four sites to accomplish the tasks. Based such derivations, we obtain the precision and recall rates are all greater than 0.9, which support that our parallel method can be effective and feasible to be applied to a very large database for mining association rules.

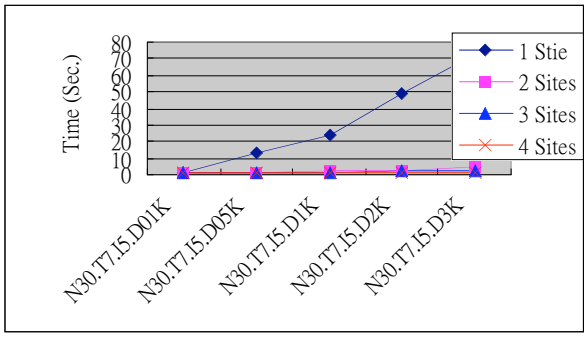

Fig. 4.3: Conducting the Performance Test of Generating L_k in Parallel.

5. Conclusions and Future Work

The association rule mining is useful in selective marketing, decision analysis, and business intelligence. However, most of the algorithms suffer from its time-consuming process. In this paper, we have presented a parallel approach that de-clusters the transaction database into partitions and dispatches the task to all participating sites. The purpose is to absolutely eliminate inter-site communications after distributing all subtasks. Such merit can be employed to support a high-performance parallel and distributed grid computation over a wide-area network.

Based on this approach, all the transaction data will be de-clustered into partitions, such that all partitions are not only quite similar to each others, but also quite similar to the original group. Then, the subtasks of itemset generation will be assigned to the participating sites by initially assigning elements in L_1 in row-prime order to highly balance the workload of each site.

We have conducted an empirical study on our approach and found that the obtained result is almost exactly the same as that obtained from *Apriori* when the dataset is large enough. Besides, we have also shown the *Apriori property* employed in the original *Apriori* algorithm should be relaxed to retain some locally infrequent but may be globally frequent subsets of a candidate itemset to avoid information loss.

The inaccurate results produced by our approach are caused from unbalanced partitions produced by the de-clustering process and the round-off error on computing the local support count $s^i = s/m$, where s and m are the original support count and the number of processors, respectively.

Based on different criteria, association rules can be further classified into several categories such as the following categories summarized in (Han and Kamber 2001):

1. Based on the types of values handled in the rule, associations can be classified into *Boolean* versus *quantitative*.

2. Based on the dimensions of data involved in the rules, associations can be classified into *single-dimensional* versus *multi-dimensional*.
3. Based on the levels of abstractions involved in the rule, associations can be classified into *single-level* versus *multi-level*.
4. Based on various extensions to association mining, association mining can be extended to correlation analysis, and the mining of maximal frequent patterns and frequent closed itemsets.

All of the categories have their corresponding algorithms been proposed. How to parallelize these algorithms by employing the concept of data de-clustering and the row-prime order task distribution will be further investigated in the future.

References

Aggarwal, C.C. and P.S. Yu, "A New Approach to Online Generation of Association Rules," *IEEE Trans. Knowledge & Data Eng.* (13:4), 2001, pp. 527-540.

Aggarwal, C.C. and P.S. Yu, "Mining Associations with the Collective Strength Approach," *IEEE Trans. Knowledge & Data Eng.* (13:6), 2001, pp. 863-873.

Aggarwal, C.C. and P.S. Yu, "Redefining Clustering for High-Dimensional Applications," *IEEE Trans. Knowledge & Data Eng.* (14:2), 2002, pp. 210-225.

Aggarwal, C.C., C. Procopiuc, and P.S. Yu, "Finding Localized Associations in Market Basket Data," *IEEE Trans. Knowledge & Data Eng.* (14:1), 2002, pp. 51-62.

Agrawal, R., and J.C. Shafer, "Parallel Mining of Association Rules," *IEEE Trans. Knowledge & Data Eng.* (8:6), 1996, pp. 962-969.

Agrawal, R., and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Proc. the 20th International Conf. Very Large Data Bases－VLDB'*94, Sep. 1994, pp. 487-499.

Agrawal, R., T. Imielinski, and A. Swami, "Mining Association Rule between Sets of Items in Large Databases," *Proc. the ACM SIGMOD International Conf. Management of Data－SIGMOD'*93, May 1993, pp. 207-216.

Brin, S., R. Motwani, J.D. Ullman, and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *Proc. the ACM SIGMOD International Conference on Management of Data－SIGMOD'*97, 1997, pp. 255-264.

Cheung, D., and Y. Xiao, "Effect of Data Skewness in Parallel Mining of Association Rule," *Proc. 12th Pacific-Asia Conf. Knowledge Discovery & Data Mining*, 1999, pp. 48-60.

Cheung, D.W., J. Han, V.T. Ng, A.W. Fu, and Y. Fu, "A Fast Distributed Algorithm for Mining Association Rules", *Proc. 4th International Conf. Parallel & Distributed Information Systems*, 1996, pp. 31-42.

Cheung, D.W., S.D. Lee, and Y. Xiao, "Effect of Data Skewness and Workload Balance in Parallel Data Mining," *IEEE Trans. Knowledge and Data Eng.* (14:3), 2002, pp. 498-514.

Fang, M.T., R.C.T. Lee, and C.C. Chang, "The Idea of De-Clustering and its Applications", *Proc. 12ih International Conf. Very Large Data Bases－VLDB'*86, Kyoto, Japan, Aug. 1986, pp. 181-188.

Fox, Geoffrey, Education and the enterprise with the Grid, *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and T. Hey eds. John Wiley & Sons, 2003.

Graham, R. and P. Hell, "On the History of the Minimum Spanning Tree Problem," *Annals of the History of Computing* (7:1), 1985, pp. 43-57.

Han, E., G. Karypis, and V. Kumar, "Scalable Parallel Data Mining for Association Rules," *Proc. ACM-SIGMOD Int'l Conf. Management of Data*, 1997.

Han, J. and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," *Proc. 21st Int'l Conf. Very Large Data Bases*, 1995, pp. 420-431.

Han, J. and M. Kamber, §6.2.1 in *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

Kruskal, J. B., Jr., "On the Shortest Spanning Subtree of a Graph and Traveling Salesman Problem," *Proc. the American Mathematical Society*, V. 7, 1956, pp. 48-50.

Lee, R.C.T., "Clustering Analysis and Its Applications," *Advances in Information System Science*, Ed. By J.T. Tou, Plenum Press, New York, Vol. 8, 1981, pp. 169-292.

Liu, H. and R. Setiono, "Feature Selection via Discretization," *IEEE Trans. Knowledge & Data Eng.* (9:4), July/Aug. 1997, pp. 642-645.

Park, J.S., M.S. Chen, and P.S. Yu, "Efficient Parallel Data Mining for Association Rules," *Proc. the 1995 Int'l Conf. Infor. & Knowledge Management*, 1995, pp. 31-36.

Rastogi R. and K. Shim, "Mining Optimized Association Rules with Categorical and Numerical Attributes," *IEEE Trans. Knowledge & Data Eng.* (14:1), 2002, pp. 29-50.

Samet, H., *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1989, pp.13-15.

Savasere A., E. Oiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," *Proc. the 20th Int'l Conf. Very Large Data Bases－VLDB'*95, 1995, pp. 432-444.

Yao, A.C., "An O(|E| log log |V|) Algorithm for Finding Minimum Spanning Trees," *Information Processing Letters* (4:1), 1975, pp. 21-23.

Zaki, M.J., "Parallel and Distributed Association Mining: A Survey," *IEEE Concurrency—special issue on Parallel Mechanisms for Data Mining* (7:4), December, 1999, pp14-25.