Expert Systems with Applications 37 (2010) 2459–2464

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A load-balanced distributed parallel mining algorithm

Kun-Ming Yu [a], Jiayi Zhou [b,*], Tzung-Pei Hong [c], Jia-Ling Zhou [d]

[a] Department of Computer Science and Information Engineering, Chung Hua University, 707, Sec. 2, WuFu Rd., HsinChu 300, Taiwan, ROC
[b] Institute of Engineering and Science, Chung Hua University, 707, Sec. 2, WuFu Rd., HsinChu 300, Taiwan, ROC
[c] Department of Computer Science and Information Engineering, National University of Kaohsiung, 700, Kaohsiung University Rd, Kaohsiung 811, Taiwan, ROC
[d] Department of Information Management, Chung Hua University, 707, Sec. 2, WuFu Rd., HsinChu 300, Taiwan, ROC

ARTICLE INFO

Keywords:
Parallel and distributed processing
Cluster computing
Frequent patterns
Association rules
Data mining

ABSTRACT

Due to the exponential growth in worldwide information, companies have to deal with an ever growing amount of digital information. One of the most important challenges for data mining is quickly and correctly finding the relationship among data. The Apriori algorithm has been the most popular technique in finding frequent patterns. However, when applying this method, a database has to be scanned many times to calculate the counts of a huge number of candidate itemsets. Parallel and distributed computing is an effective strategy for accelerating the mining process. In this paper, the Distributed Parallel Apriori (DPA) algorithm is proposed as a solution to this problem. In the proposed method, metadata are stored in the form of Transaction Identifiers (TIDs), such that only a single scan to the database is needed. The approach also takes the factor of itemset counts into consideration, thus generating a balanced workload among processors and reducing processor idle time. Experiments on a PC cluster with 16 computing nodes are also made to show the performance of the proposed approach and compare it with some other parallel mining algorithms. The experimental results show that the proposed approach outperforms the others, especially while the minimum supports are low.

1. Introduction

With the rapid development of information technology, companies have been working on digitizing all areas of business to improve efficiency and thus competitiveness. Tremendous amounts of data are thus generated due to the full digitization. It is important to extract meaningful information from scattered data. Data mining techniques have recently been developed for this purpose. They can be classified into different models like classification, regression, time series, clustering, association, sequence, and among others. Especially, association rules are commonly used in many applications.

The most important step in mining association rules is to discover frequent patterns. This step needs to count the times of the patterns appearing in a database. According to the ways of generating candidate patterns, researches can be classified into generate-and-test (*Apriori*-like) (Agawal, Imilinski, & Swami, 1993) and pattern-growth approach (*FP-growth*) (Han, Pei, Yin, & Mao, 2004). The former uses a bottom-up approach, which extends frequent subsets by one item at a time. If an itemset with length k is frequent, then its any subset with length less than k is also frequent. Although many Apriori-like methods have been proposed, it takes long time to find the frequent patterns when the database contains a large number of transactions. Some researches thus apply parallel and distributed techniques to effectively speed-up the mining process (Agrawal & Shafer, 1996; Cheung, Han, Ng, Fu, & Fu, 1996; Cheung, Lee, & Xiao, 2002; Cheung, Ng, & Fu, 1996; Ye & Chiang, 2006). In a distributed environment, irregular and imbalanced computation loads may cause the overall performance to be greatly degraded. Load balance among processors in the mining process is thus very important to parallel and distributed mining.

In this paper, the Distributed Parallel Apriori (DPA) algorithm is proposed as a solution to this problem. Its goal is to reduce the frequency of database scans and to balance the computation loads among participated computing nodes. In the proposed method, a database has only to be scanned once because metadata are stored in the form of Transaction Identifiers (TIDs). The approach also takes itemset counts into consideration to improve load balancing as well as to reduce idle time of processors.

The experimental results also show that the running time of the proposed approach is significantly less than that of some previous methods. The results also depict that DPA can successfully reduce the number of scan iterations and can evenly distribute workloads among processors.

The paper is organized as follows. Association rules and parallel-distributed algorithms are reviewed in Section 2. The DPA algorithm is proposed in Section 3. An example to illustrate

* Corresponding author. Tel.: +886 3 5186360; fax: +886 3 5186416.
E-mail addresses: yu@chu.edu.tw (K.-M. Yu), jyzhou@pdlab.csie.chu.edu.tw (J. Zhou), tphong@nuk.edu.tw (T.-P. Hong), jlzhou@pdlab.csie.chu.edu.tw (J.-L. Zhou).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.07.074

the proposed algorithm is given in Section 4. The experimental results are shown in Section 5. Finally, the conclusion is stated in Section 6.

2. Related work

The frequent-pattern mining problem is defined as follows. Let $DB = \{T_1, T_2, \ldots, T_m\}$ be a transactional database, in which each transaction T_i consists of a set I of items $\{i_1, i_2, \ldots, i_m\}$. Associated with each transaction is a unique identifier, TID (Apte & Weiss, 1997). The support of an itemset x in a database DB, denoted $sup_{DP}(x)$, is the number of transactions in DB that contain x. Formally, $sup_{DP}(x) = |\{t|\ t\ DB \text{ and } x\ t\}|$. The problem of frequent-pattern mining is to find all itemsets x's with $sup_{DP}(x) \geqslant s$ for a given threshold s ($|DB| \geqslant s \geqslant 1$).

The Apriori algorithm was proposed by Agrawal and Srikant (1994) and is one of the most representative algorithms in mining frequent patterns. Its main idea is based on the observation that subsets of frequent itemsets must be frequent as well. The Apriori algorithm extends frequent itemsets by one item at a time and tests the candidates against the data. The algorithm terminates when no further successful extension is possible. Even though the Apriori algorithm can efficiently find frequent patterns, the execution time gets longer when the database sizes get larger because of each candidate itemset should be tested against the database. Since each candidate itemset with the same length may be tested independently, a good design of data structure will make the algorithm to be parallelized easily. Thus, many distributed parallel methods based on the Apriori algorithm were proposed (Einakian & Ghanbari, 2006; Parthasarathy, Zaki, Ogihara, & Li, 2001; Zaki, Ogihara, Parthasarathy, & Li, 1996; Zaki, Parthasarathy, Ogihara, & Li, 1997).

Agrawal and Shafer (1996) then proposed parallel algorithms based on *Count Distribution* (*CD*) and *Data Distribution* (*DD*) to solve the frequent-pattern mining problem. The former (*CD*) partitions the database into blocks and sends them to processors to compute frequent itemsets. In the approach, $(k+1)$-itemsets are also generated from k-itemsets. The advantage is that each processor only needs to process the data it owns. The other one, *DD,* then further improves the memory usage of *CD*. The amount of communication, however, increases with processors increased. Cheung et al. then proposed the Fast Distributed Mining (FDM) approach for finding association rules (Cheung, Han et al., 1996; Cheung, Ng et al., 1996). FDM reduces the candidate set by both local pruning and global pruning. Cheung et al. also improved the above approach and proposed the Fast Parallel Mining (FPM) algorithm (Cheung et al., 2002) in 2002 for parallel and distributed mining. FPM needs less communication than FDM. Its mining performance can thus be raised.

Ye and Chiang (2006) also proposed a parallel-distributed algorithm based on the Trie tree (Bodon, 2003). Their algorithm distributes workloads according to the Trie tree to balance and speed-up the computation. However, the items are distributed to the nodes only based on the first level of the Trie tree. This may cause the sizes of candidate itemsets (workloads) among processors significantly varying. Moreover, this method also requires a database to be scanned many times.

Recently, Wu and Li proposed an efficient frequent-pattern mining algorithm, called *EDMA*, based on the Apriori algorithm (Wu & Li, 2008). *EDMA* uses the *CMatrix* data structure to store the transactions for mining. This can get rid of database re-scanning. *EDMA* can minimize the number of candidate sets and reduce the exchange messages by local and global pruning. Since it may decrease the average size of transactions and datasets, the execution time for verifying frequency can also be reduced. Moreover, it can decrease the communication time among computing nodes. The execution time, however, gets longer when the database size is larger, since *EDMA* will access *CMatrix* a lot of times when calculating candidate itemsets.

Therefore, in this paper, a Distributed Parallel Apriori (DPA) algorithm is proposed to speed-up the process of frequent-pattern mining. By storing the TIDs of itemsets and precisely calculating and distributing computation workloads, DPA is able to effectively accelerate the computation of itemsets and reduce the required scan iterations to a database.

3. The proposed Distributed Parallel Apriori (DPA) algorithm

The execution time of different processors in Ye and Chiang's algorithm may vary in a wide range because distributing the items according to the Trie tree in upper levels may lead to imbalanced workload. In order to observe the execution time of each processor in Ye and Chiang's algorithm, we implement their algorithm on the dataset T10I4D12KN100 K with the minimum support set at 0.2% using the MPI library on a PC cluster. Table 1 shows the execution time of each processor in Ye and Chiang's algorithm. Moreover, it can also be observed that the CPU time and the communication time occupied 97% and 3% of the total execution time, respectively. Since their algorithm scans a database many times to verify whether the candidate patterns are frequent or not, we may also improve the performance by reducing database scan.

To avoid the problems of load imbalance and multiple scans, the DPA algorithm is proposed in this paper, so that a database needs to be scanned only once while maintaining load balance among processors. In the proposed algorithm, each transaction has a unique Transaction Identifier, called TID. By using hash functions to store TIDs in a table structure, the number of itemsets can be quickly calculated without the need of re-scanning the database.

For achieving a good load balance, the proposed approach adopts a heuristic based on the weights of frequent itemsets. The workload measure for finding frequent $(k+1)$-itemsets is estimated from frequent k-itemsets. The frequent k-itemsets are first sorted according to their counts in descending order. Let $len(freq_k)$ denote the total number of frequent k-itemsets. The weight of the ith frequent k-itemset (I_i) is then set as follows:

$$weight(I_i) = len(freq_k) - i - 1. \tag{1}$$

The concept can be represented by Fig. 1.

The total weight of all the frequent k-itemsets can then be found as follows:

$$TotalWeight = \sum_{i=0}^{len(freq_k)-1} weight(I_i). \tag{2}$$

Assume there are p processors available. Each professor can then process the subset of k-itemsets with the sum of their weights close to $TotalWeight/p$. For simplicity, the frequent k-itemsets are put one

Table 1
Execution time of each processor in Ye and Chiang's algorithm.

Processor ID	Execution time	CPU time	Communication time	
			Send	Receive
0	129.4010358	127.838176	0.367843151	1.195016623
1	89.78008485	88.00373292	0.33731699	1.439034939
2	77.47582674	75.11535358	0.364768028	1.995705128
3	89.60780978	87.40449095	0.320035219	1.883283615
4	75.14281654	73.05057096	0.312260389	1.779985189
5	70.93045855	68.58841801	0.307432413	2.034608126
6	71.56252313	68.99555421	0.301764011	2.265204906
7	72.59657574	70.05979991	0.319941998	2.21683383