Abstract

Association rules are the main technique for data Mining. Apriori algorithm is a conventional algorithm of association rule mining. There are many algorithms for mining association rules and their variations are proposed on basis of Apriori algorithm, but those algorithms are not efficient. With the rapid development of networks and information technology, the vast information has paid more and more attention by people. While getting the information with high speed, the analysis and mining of the information and rules hidden deep in the data are also getting more attention. Data mining technology is to organize and analyze the data, which can extract and discover knowledge from the huge and unstructured data, so how to apply the data mining techniques in transactional databases is the focus of this topic studied. In this paper, combining with parallel programming techniques, Apriori algorithm in association rule mining algorithm is described in detail, the algorithm implementation process is illustrated, and the optimized methods of the algorithm are discussed.
1. Introduction

Now a day’s modern computer technology and database technology has been developing with rapid speed and could support to store and quickly retrieve the grand scale databases or data warehouses, but these techniques was only to gather these "huge" data, and not to efficiently organize and use the knowledge hidden in them [7], which in turn led to structured data but data with less knowledge. The provision of data mining technology met people needs. The technology involved in artificial intelligence, machine learning, statistical analysis and other technologies, and it makes decision analysis into a new stage [5]. In this paper the association rule mining algorithm - Apriori algorithm which is commonly used in data mining is mainly discussed. Frequent item sets play key role in many data mining tasks that try to find interesting patterns from databases [1]. The original inspiration of searching frequent pattern came from the need to analyze supermarket transaction data to examine customer behavior in terms of the purchased products. Frequent Pattern shows how often items are purchased together [9]. The frequent item set and association rule mining problems are came to focus. Many research papers have been published since past decades. Many algorithms are searched and developed for existing algorithms to solve these mining problems more efficiently [13]. In this chapter, we are going to optimize the apriori algorithm in serial and parallel approach.

2. Related Work done

Agrawal R, Srikan R present two new algorithms for solving this problem that is fundamentally different from the known algorithms. Empirical evaluation shows that these algorithms outperform the known algorithms by factors ranging from three for small problems to more than an order of magnitude for large problems [5]. Chao Yang, Tzu Chang and Chih Chang presents the comparison of some tools that are specifically designed to extract the most of data parallelism on multi-core system using OpenMP [3]. Ying Liu, Fuxiang Gao’s experiments show that the parallel implementation of the algorithm using results in a speed-up about 200% compared with sequential implementation on a Dual-core processor, while a speed-up about 400% on a Quad-core processor. [4]. Ketan shah and Sunita Mahajan present the performance of parallel Apriori algorithm on heterogeneous nodes with different datasets and n processors on a commodity cluster of machines [6]. Anuradha.T and Satya Prasad.R presents an evaluation of the performance of Apriori on a hyper threaded dual core processor compared to the performance on a non-hyper threaded dual core processor using fread() and mmap() functions [7]. Zhang Zheng, Jaiwen Li, Xuhoo Chen, Li Shen, and Zhiying Wang present a performance model for OpenMP parallelized loops to address the critical factors which influences the performance [8].Kyung Min Lee, Tae Houn Song, Seung Hyun Yoon, Key Ho Kwon and Jae Wook Jeon presents a parallel programming model, OpenMP, and parallel programs that can be benchmarked to multi-core processors of embedded boards using OpenMP and Executed parallel programs on a dual-core embedded system, analyzing the performance of sequential programs and parallel programs by SERPOP analysis [9].Anuradha.T, Dr.Saty Prasad.R, Dr.Tirumala Rao.S.N, evaluates the performance of Apriori using Linux mmap() function compared to fread() function in both the serial and parallel environments [10].

3. Apriori Algorithm

Notations:
K=pass number, Fk=set of frequent K-itemset(with those min support) , ck=set of candidate K-itemset.
1. $K=1$
2. $F_k=\{i/i \in I, \delta(i) \geq \text{min. sup}\}$ \{Find all frequent 1-itemset\}
3. Repeat
4. $K=K+1$
5. $C_k=\text{apriori-gen}(F_k-1)$ \{generate candidate itemset\}
6. For each transaction $t \in T$ do
7. $C_t=\text{subset}(C_k,t)$ \{identify all candidate that belongs to $t$\}
8. For each candidate itemset $C \in C_t$ do
9. $\delta(c)=\delta(c)+1$ \{Increment support count\}
10. End for
11. End for
12. $F_k=\{C/C \in C_k, \delta(c) \geq \text{minsup}\}$ \{exact the frequent itemset\}
13. Until $F_k=\Phi$
14. Result=$U_fk$

4. Related Example
Following is an example based on the transaction database $D$ of Figure 1. Consisting of total nine transactions in this database, that is, $T=9$. We use the improved Apriori algorithm for finding frequent item sets in $D$.

![Transaction Database Example](image)

Figure 4.1: Generation of candidate item sets and frequent item sets.
i. Scan database D for counting each candidate. In its first pass, each item is a member of the set of candidate 1-itemsets, C1. The algorithm scans all of the transactions to count the number of occurrences of each item.

ii. Compare candidate support count with minsup. Suppose that the minimum transaction support count required is 2. The set of frequent 1-itemsets, L1, can then be determined. It consists of the candidate1-itemsets satisfying minimum support.

iii. Generate C2 candidates from L1 and scan D for count of each candidate. To discover the set of frequent 2-itemsets, L2, the algorithm generates a candidate set of 2-itemsets, C2. And then the transactions in D are scanned and the support count of each candidate item set in C2 is accumulated, as shown in the table of C2 in Figure 1.

iv. Compare candidate support count with minsup. The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support. Then D2 was determined from L2.

v. Generate C3 candidates from L2 and scan D2 for count of each candidate. First let C3=L2×L2={{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}. Based on the Apriori property that all subsets of a frequent item set must also be frequent, it can determine that the four latter candidates cannot possibly be frequent, therefore remove them from C3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D2 to determine L3.

vi. Compare candidate support count with minsup. The transactions in D2 are scanned in order to determine L3, consisting of those candidate 3-itemsets in C3 having minimum support.

vii. The algorithm uses L3×L3 to generate a candidate set of 4-itemsets, C4. Although the join results in {{I1, I2, I3, I5}}, this item set is pruned since its subset {{I2, I3, I5}} is not frequent. Thus, C4=Φ, and the algorithm terminates, having found all of the frequent item sets.

5. Analysis And Evaluation Of The Algorithm
The Serial/Parallel Apriori Algorithm has been implemented by Java 8 on a personal computer with 2.10 GHz Intel Core2Duo CPU, 2 GB in memory and Windows 7 operating system. The experimental transactional database sources are created by Generator and not taken from market. In the Fig.2, the horizontal axis represents the number of support in database and the vertical axis represents mining time (second) of the same database with the algorithm Apriori. The two curves denote different time cost of the algorithm Apriori and RAAT with different minsup.

6. Conclusion
In this paper, we have implemented Apriori algorithm on quad core processor in a serial and parallel manner with four standard datasets using different support counts varying from 30% to 70%. From our work we can say that parallel results are better than serial once. Our aim of measuring the serial
and parallel performance of Apriori algorithm on a quad core processor with respect to time and comparison between them get satisfied

References

[1] Jiawei Han and Micheline Kamber, Data Mining concepts and Techniques 2nd edition Morgan Kaufmann Publishers, San Francisco 2006.


