SQL Based Association Rule Mining using Commercial RDBMS (IBM DB2 UDB EEE)

Takeshi Yoshizawa☆★, Iko Pramudiono★, Masaru Kitsuregawa★

★Institute of Industrial Science, The University of Tokyo 7-22-1 Roppongi, Minato-ku, Tokyo 106, Japan
☆IBM Japan Co.,Ltd. 1-1, Nakase, Mihama-ku, Chiba-shi, Chiba 261-8522, Japan

{yoshi,iko,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. Data mining is becoming increasingly important since the size of databases grows even larger and the need to explore hidden rules from the databases becomes widely recognized. Currently database systems are dominated by relational database and the ability to perform data mining using standard SQL queries will definitely ease implementation of data mining. However the performance of SQL based data mining is known to fall behind specialized implementation and expensive mining tools being on sale. In this paper we present an evaluation of SQL based data mining on commercial RDBMS (IBM DB2 UDB EEE). We examine some techniques to reduce I/O cost by using View and Subquery. Those queries can be more than 6 times faster than SETM SQL query reported previously. In addition, we have made performance evaluation on parallel database environment and compared the performance result with commercial data mining tool (IBM Intelligent Miner). We prove that SQL based data mining can achieve sufficient performance by the utilization of SQL query customization and database tuning. Keywords: data mining, parallel SQL, query optimization, commercial RDBMS

1 Introduction

Extracting valuable rules from a large set of data has attracted lots of attention from both researcher and business community. This is particularly driven by explosion of the information amount stored in databases such as Data Warehouses during recent years. In business world, many organizations begin to apply data mining techniques directly to raw transaction data、and some results such as unidentified buying patterns and credit card fraud indications are widely recognized.

One method of data mining is finding association rule [1]. Basket data analysis is typical of this method. There are some approaches proposed to mine association rules, [1,2,6,9] some of them are based on relational database standard SQL [3,7,8]. But this kind of mining is known as CPU power demanding application and it has to handle very large amounts of transaction data. Unfortunately SQL approach is reported to

have drawback in performance although it has many advantages such as seamless integration with existing system and high portability.[7]

On the other hand recently most major commercial database systems have included capabilities to support parallelization although no report available about how the parallelization affects the performance of complex query required by association rule mining. This fact motivated us to examine how efficiently SQL based association rule mining can be parallelized and speeded up using commercial parallel database system (IBM DB2 UDB EEE). We propose two techniques to enhance association rule mining query based on SETM [3]. And we have also compared the performance with commercial mining tool (IBM Intelligent Miner). Our performance evaluation shows that we can achieve comparable performance with commercial mining tool using only 4 nodes.

Some considerable works on effective SQL queries to mine association rule such as [7] and [8] didn't examine the effect of parallelization. [4] and [5] have reported a performance evaluation on PC cluster as parallel platform. Comparison with natively coded programs is also reported. However we use currently available commercial products for the evaluation.

2 Association Rule Mining Based on SQL

An example of association rule mining is finding "if a customer buys A and B then 90% of them buy also C" in transaction databases of large retail organizations. This 90% value is called confidence of the rule. Another important parameter is support of an itemset, such as support ({A,B,C}), which is defined as the percentage of the itemset contained in the entire transactions. For above example, confidence can also be measured as support ({A,B,C}) divided by support ({A,B}).

In our experiments we employed three type of SQL query. First of all the standard SQL query using SETM algorithm [3]. Second is the enhanced SQL query using view materialization technique. Third is another enhanced SQL query using subquery technique.

2.1 SQL query using SETM algorithm

Transaction data is transformed into the first normal form (transaction ID, item). In the first pass we simply gather the count of each item. Items that satisfy the minimum support are inserted into large itemsets table C_1 that takes form (item, item count). SETM employs temporary tables to reuse item combination in next pass. In first pass, transactions that match large itemsets are preserved in temporary table R_1.

In other passes for example pass k, we first generate all lexicographically ordered candidate itemsets of length k into another temporary table RTMP_k by self-joining table R_(k-1) that contains k-1 length transaction data. Then we generate the count for those itemsets, itemsets that meet minimum support are included into large itemset table C_k. Finally transaction data R_k of length k generated by matching items in candidate itemset table RTMP_k with items in large itemsets. In order to avoid excessive I/O, we disable the log during the execution.

2.2 Enhanced SETM query using view materialize technique

SETM has to materialize its temporary tables namely R_k and RTMP_k. Those temporary tables are only required in the next pass and they are not needed for generating the rules. In fact, those tables can be deleted after execution of its subsequent pass. Based on this observation we could avoid materialization cost of those temporary tables by replacing the table creation with view.

2.3 Enhanced SETM query using subquery technique

We expect significant performance improvement with utilization of view, however view still requires time to access the system catalog and are holding locks to the system catalog table during creating views so we further use subquery instead of temporary tables. Therefore we embed the generation of item combinations into the query to generate large itemsets.

3 Performance Evaluation

3.1 Parallel Execution Environment

In our experiment we employed commercial Parallel RDBMS: IBM DB2 UDB EEE version 6.1 on IBM UNIX Parallel Server System: IBM RS/6000 SP. 12 nodes make this system and using shared nothing architecture. Each node has POWER2 77Mhz CPU, 4.4GB SCSI hard disk, 256MB RAM and connected by High Performance Switch HPS with 100MB/s network speed.

We also used commercial data mining tool IBM Intelligent Miner on single node of RS/6000 SP for performance comparison with the SQL based data mining.

3.2 Dataset

We use synthetic transaction data generated with program described in Apriori algorithm paper [2] for experiment. The parameters used are : number of transactions 200000, average transaction length 10 and number of items 2000. Transaction data is partitioned uniformly by hashing algorithm corresponds to transaction ID among processing nodes' local hard disks.

3.3 Performance comparison

Figure.1 shows the execution time on each degree of parallelization. On average, we can derive that View and Subquery SQL is about 6 times faster than SETM SQL regardless of the number of nodes. The result is also compared with the execution time of Intelligent Miner on single processing node. It is true that Intelligent Miner on single node with transaction data stored in flat file is much faster than the SQL queries. However, the View and Subquery SQL are 50% faster than Intelligent Miner

on single node if the transaction data have to be read from RDBMS. We exemplified that we can achieve comparable performance of Intelligent Miner on single node with flat file by activating only 4 nodes when we used View and Subquery SQL. The result gives evidence for the effectiveness of parallelization of SQL query to mine association rule.

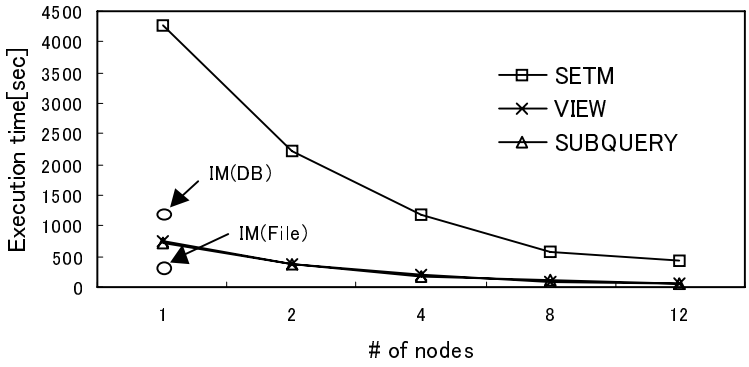

Fig. 1. Execution time on parallel database environment

The speedup ratio is shown in figure 2. This is also reasonably good, especially View and Subquery SQL are not being saturated as the number of processing nodes increased. That means they can be parallelized well. The execution is 11 times faster with 12 nodes. In parallel environment, network potentially becomes botleneck which degrades the speed-up ratio. However our experiments suggest that association rule mining using variants of SETM is mostly CPU bound and network I/O is negligible.

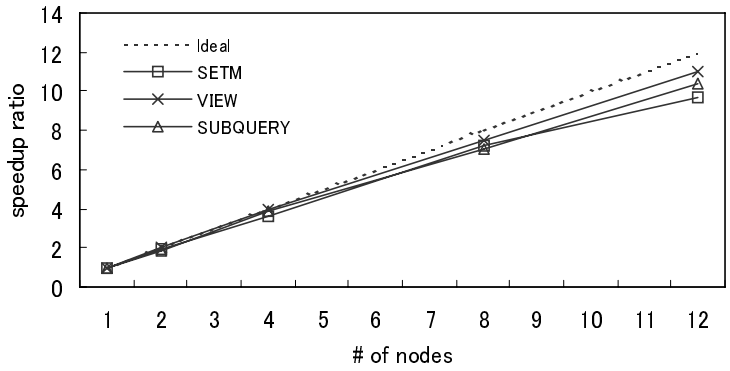

Fig. 2. Speedup ratio in parallel database environment

3.4 Analysis of execution trace

In this section, we compared the three variations of SQL query described before. The performance evaluation is done on 12 nodes.

The mining is two passes long. It is well known that in most cases the second pass generates huge amount of candidate itemsets thus it is the most time consuming phase[4][5]. Our results are very much alike. Almost over 80% of execution time belongs to PASS2 in all three SQL queries. Obviously View and Subquery SQL complete their first and second passes faster than SETM SQL query.

We have recorded the execution traces of the three SQL in each PASS. The decomposition of execution time is analysed as shown Figure 3 (PASS2) respectively. Comparing the elapsed time with the cpu time at Fig 3, we find that both are close for View SQL and Subquery SQL. This means these SQL's are cpu bound, while SETM SQL is not cpu bound. Most of execution time of SETM query is dominated by disk write time for creating temporary table such as R_k and RTMP_k. We can also see that sort time is almost equal for all three SQL's, which represents the cost of group by aggregation.

In PASS2, SETM reuses item combinations in temporary table R1 on the secondary storage that is generated in PASS1. We replace it with view or subquery. Then data is transferred directly through memory from PASS1 to PASS2. Figure 3 indicates that PASS2 of those modified SQL queries only read data from buffer pool. Thus the disk write time of View SQL and Subquery SQL is almost negligible, although it is dominait for SETM SQL. This analysis clarifies the problem of SETM and how to cost can be reduced for View and Subquery SQLs, which is the key to the performance improvement.

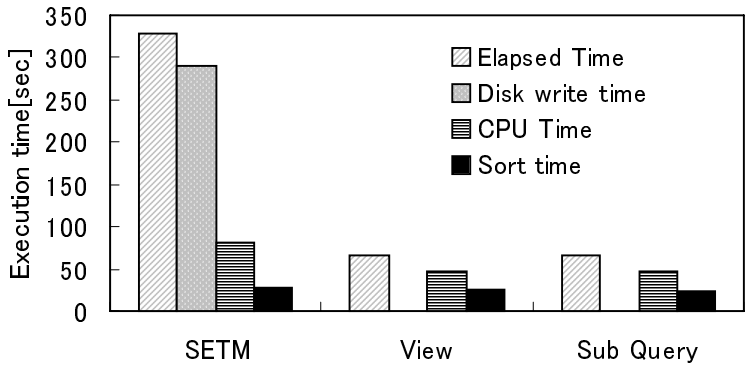

Fig. 3. Decomposition of execution time of PASS2 for three types of SQL queries

4 Summary and Conclusion

The ability to perform data mining using standard SQL queries will benefit data warehouses with the better integration with commercial RDBMS. It also allows easier porting codes among different systems.

In this paper, we reported the parallelization of SQL query to mine association rule on commercial RDBMS (IBM DB2 UDB EEE). We showed that good speedup ratio can be achieved, that means it is parallelized well. We also examined two variations of SETM SQL queries to improve performance, which reduce I/O cost by using View materialize or Subquery technique, and can achieve performance more than 6 times faster than SETM SQL query with two passes. We expect still more improvement with more passes. We'd like to report it next time.

We have compared the parallel implementation of SQL based association rule mining with commercial data mining tool (IBM Intelligent Miner). Through real implementation, we have showed our improved SETM query using View or Subquery can beat the performance of specialized tool with only 4 nodes while original SETM query needs more than 24 processing nodes to achieve the same performance. We don't have to buy expensive data mining tool, since parallel execution of SQL comes at no extra cost. It is also extremely easy to implement and flexible.

There remain lots of further investigations. We'd like to check the performance with more passes situations. In addition we plan to do using more large transaction data. And we'd like to investigate the effect of intra parallelism under SMP environment.

References

1. R. Agrawal, T. Imielinski, A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proc. of the ACM SIGMOD Conference on Management of Data, 1993.
2. R. Agrawal, R. Strikant. Fast Algorithms for Mining Association Rules. In Proc. of the Very Large Database (VLDB) Conference, 1994.
3. M. Houtsma, A. Swami. Set-oriented Mining of Association Rules. In Proc. of International Conference on Data Engineering (ICDE), 1995.
4. Iko Pramudiono, Takahiko Shintani, Takayuki Tamura, Masaru Kitsuregawa. Parallel SQL Based Association Rule Mining on Large Scale PC Cluster: Performance Comparison with Directly Coded C Implementation. In Proc. of Third Pacific-Asia Conference on Knowledge Discovery a nd Data Mining (PAKDD99), 1999.
5. Iko Pramudiono, Takahiko Shintani, Takayuki Tamura, Masaru Kitsuregawa. Mining Generalized Association Rule using Parallel RDB Engine on PC Cluster. In Proc. of First International Conference on Data Warehousing and Knowledege Discovery (DAWAK99), 1999.
6. Jong Soo Park, Ming-Syan Chen, Philip S. Yu. An Effective Hash-Based Algorithm for Mining Association Rules. In Proc. of the ACM SIGMOD Conference on Management of Data, 1995.
7. Sunita Sarawagi, Shiby Thomas, Rakesh Agrawal. Integrating Association Rule Mining with Relational Database Systems : Alternatives and Implications. In Proc. of the ACM SIGMOD Conference on Management of Data, 1998.
8. Shiby Thomas, Sharma Chakravarthy. Performance Evaluation and Optimization of Join Queries for Association Rule Mining. In Proc. of First International Conference on Data Warehousing and Knowledege Discovery (DAWAK99), 1999.
9. T. Shintani, M. Kitsuregawa. Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy. In Proc. of the ACM SIGMOD Conference on Management of Data, 1998.