e-ISSN: 2349-9745
p-ISSN: 2393-8161

Scientific Journal Impact Factor (SJIF): 1.711

International Journal of Modern Trends in Engineering and Research

www.ijmter.com

A Survey on Improve Efficiency And Scability vertical mining using Agriculter large data base

Annu Kumari Mishra[1], Anju Singh[2], Divakar Singh[3]
[1] Computer Science & engineering,BU-UIT
[2] Computer Science & Information Technology, UTD-BU
[3] Computer Science & engineering,BU-UIT

Abstract— Basic idea is that the search tree could be divided into sub process of equivalence classes. And since generating item sets in sub process of equivalence classes is independent from each other, we could do frequent item set mining in sub trees of equivalence classes in parallel. So the straightforward approach to parallelize Éclat is to consider each equivalence class as a data (agriculture). We can distribute data to different nodes and nodes could work on data without any synchronization. Even though the sorting helps to produce different sets in smaller sizes, there is a cost for sorting. Our Research to analysis is that the size of equivalence class is relatively small (always less than the size of the item base) and this size also reduces quickly as the search goes deeper in the recursion process. Base on time using more than using agriculture data we can handle large amount of data so first we develop éclat algorithm then develop parallel éclat algorithm then compare with using same data with respect time .with the help of support and confidence.

Keywords - Association Rule, Apriori algorithm, Éclat algorithm, AI, Parallel

I. INTRODUCTION

To count supports of candidates, we need to go through transactions in the transaction database and check if transactions contain candidates. Since the transaction database is usually very large, it is not always possible to store them into main memory. Furthermore, to check if a transaction containing an item set is also a non-trivial task. So an important consideration in frequent item set mining algorithms is the representation of the transaction database to facilitate the process of counting support. There are two layouts that algorithms usually employ to represent transaction databases: Horizontal and vertical layout.

In the horizontal layout, each transaction T_i is represented as T_{id} *:* (t_{id} *, I)* where T_{id} is the transaction identifier and I is an item set containing items occurring in the transaction. The initial transaction consists of all transactions T_i.

As the size of item sets increases, the size of their tid sets will decrease, using the vertical layout, counting support is usually faster and using less memory than counting support when using the horizontal layout.

II. PROBLEM DOMAIN

The Apriori heuristic achieves good performance gained by (possibly significantly) reducing the size of candidate sets. However, in situations with a large number of frequent patterns, long patterns, or quite low minimum support thresholds, this problem to be overcome in using éclat. but now *éclat* algorithm may suffer from the nontrivial - It is costly to handle a huge number of candidate sets. Divide the database evenly into horizontal partitions among all processes; each process scans its local database partition to collect the local count of each item; but do not handle properly large amount of data. All processes exchange and sum up the local counts to get the global counts of all items and find frequent 1-itemsets.

III. SOLUTION DOMAIN

To better utilize the aggregate computing resources of parallel machines, a localized algorithm based on parallelization of Éclat was proposed and exhibited excellent scalability. It makes use of a vertical data layout by transforming the horizontal database transactions into vertical tid-lists of item sets. Task parallelism is our data by dividing the mining tasks for different classes of item sets among the available processes. The equivalence classes of all frequent 2-itemsets are assigned to processes and the associated. Each process then mines frequent item sets generated from its assigned equivalence classes independently, by scanning and intersecting the local tid-lists. The steps for the parallel Éclat algorithm are presented below for distributed-memory multiprocessors. Divide the database evenly into horizontal partitions among all processes; each process scans its local database partition to collect the counts for all 1-itemsets and 2-itemsets; all processes exchange and sum up the local counts to get the global counts of all 1-itemsets and 2-itemsets, and find frequent ones among them; Each process transforms its local database partition into vertical tid-lists for all frequent 2-itemsets, we also introduce a new parallel approach for Éclat algorithm, which can address the problem of load unbalancing and better exploit power of association with many nodes.

IV. SYSTEM DOMAIN

All the experiments are performed on a 3-GHz Pentium PC machine with 512 megabytes main memory, running on Microsoft Windows/NT. All the programs are written in Microsoft Visual studio .net(C# 7.0). Notice that we do not directly compare our absolute number of runtime with those in some published reports running on the RISC workstations; because different machine architectures may differ greatly on the absolute runtime for the same algorithms. Instead, we implement their algorithms to the best of our knowledge based on the published reports on the same machine and compare in the same running environment. Please also note that *run time* used here means the total execution time, that is, the period between input and output, instead of *CPU time* measured in the experiments in some literature. We feel that run time is a more comprehensive measure since it takes the total running time consumed as the measure of cost, whereas CPU time considers only the cost of the CPU resource. This method has limitation that we have to compress the dataset and after mining frequent item set we have to again decompress the data set.

- This application not having its own storage management. It depends on SQL SERVER- data base package.
- The application has no window based GUI.
- The application will work only for VB net (7.0) higher version
- The application is based on Boolean association rules.

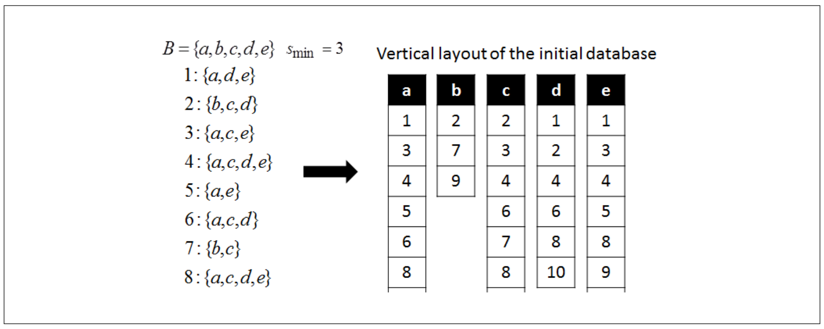
V. APPLICATION DOMAIN

(1) It constructs a highly compact parallel Éclat which is usually substantially smaller than the original database, and thus saves the costly database scans in the subsequent mining processes.
(2) By using parallel technique into the process of construction which hugely shortens the time of construction. And the performance is much more scalable than the éclat method.
(3) Which avoids costly candidate generation and test by successively concatenating frequent 1-itemset in most Apriori-like algorithms?

(4) It applies a parallel-based divide-and-conquer method, which dramatically reduces the size of the subsequent conditional pattern bases and conditional P-Éclat.

A. Authors and Affiliations

a) Dr. S. Vijayarani, "An Efficient Algorithm for Mining Frequent Items in Data Streams".
b) S. Vijayaranis "Mining Frequent Item Sets over Data Streams using Éclat Algorithm".
c) Mingjun Song, and Sanguthevar Rajasekaran"A Transaction Mapping Algorithm for Frequent Item sets Mining".



$B = \{a,b,c,d,e\}$ $s_{\min} = 3$
1: $\{a,d,e\}$
2: $\{b,c,d\}$
3: $\{a,c,e\}$
4: $\{a,c,d,e\}$
5: $\{a,e\}$
6: $\{a,c,d\}$
7: $\{b,c\}$
8: $\{a,c,d,e\}$

Vertical layout of the initial database

a	b	c	d	e
1	2	2	1	1
3	7	3	2	3
4	9	4	4	4
5		6	6	5
6		7	8	8
8		8	10	9

Fig.1 Horizontal and Vertical Layout

VI. EXPECTED OUTCOME

Parallel éclat is faster than Apriori and Éclat. P-Eclate is faster and more scalable and perform incremental than Apriori. The parallel Data base is more incremental then partition Apriori and based éclat but parallel eclat save memory space since the dataset is sparse, as the support threshold is high, the frequent item sets are short and the set of such item sets is not large, the advantages of parallel eclat over Apriori are not so impressive. As the support threshold goes down, the gap becomes wider. Eclat can finish the computation for support threshold 2% within the time for Apriori over 3%. Parallel eclat is also scalable but is slower than FP-growth and apriori the advantages of parallel eclat over Apriori become obvious when the dataset contains an abundant number of mixtures of short and long frequent patterns. The experimental results to improve scalability and efficiency of éclat algorithm eclat can mine with support threshold as low, with which Apriori cannot work out within reasonable time. Parallel eclat is also scalable and faster than Apriori and éclat. Scalability with threshold over dataset with abundant mixtures of short and long frequent patterns other methods and candidate set and frequent item generations. Our experiments analysis showed that the benefit of sorting outweighs the cost of sorting and the sorting significantly reduces the running time and memory usage.

ACKNOWLEDGEMENT

I would like to express my special thanks to Dr. Divakar Singh, Head of the Department, for his valuable and constructive suggestions during the planning and development of this work.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A.N. Swami, "Mining association rules between sets of items in large databases," in ACM SIGMOD International Conference on Management of Data, Washington, 1993.

[2] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," in 20th International Conference on Very Large Data Bases, Washington, 1994.

[3] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in ACM SIGMOD International Conference on Management of Data, Texas, 2000.

[4] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," in Third International Conference on Knowledge Discovery and Data Mining, 1997.

[5] Paul W. Purdom, Dirk Van Gucht, and Dennis P. Groth, "Average case performance of the apriori algorithm," vol. 33, p. 1223–1260, 2004.

[6] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri, "Adaptive and resource-aware mining of frequent sets," in Proceedings of the 2002 IEEE International Conference on Data Mining, 2002.

[7] Yo unghee Kim, Won Young Kim and Ungmo Kim "Mining frequent item sets with normalized weight in continuous data streams". Journal ofinformation processing systems. 2010.