An improved approach of FP-Growth tree for Frequent Itemset Mining using Partition Projection and Parallel Projection Techniques

Rana Krupali
Parul Institute of Engineering and technology,
Parul University, Limda, Vadodara, Gujarat
Email: krupalirana97@gmail.com

Dweepna Garg
Parul Institute of Engineering and technology,
Parul University, Limda, Vadodara, Gujarat
Email: dweeps1989@gmail.com

Ketan Kotecha
Parul Institute of Engineering and technology,
Parul University, Limda, Vadodara, Gujarat

Abstract**—** In Data mining, it is about analyzing data; about extracting information out of data. It is a very actual as well as interesting issue having more and more data stored in database. The most important usage: customer behavior in market purchasing, shopping cart processed information provide, management of campaign , customer relationship management, mining about web usage called web mining, mining of text. In the current age of science we developed such technology by using it each type of data related to anything such like person, place, shop, or any organization can be stored. By analysis it is found that FP-growth is efficient in terms of tree construction as compared to Apriori and Tree Projection. Tree Projection is faster and more scalable than Apriori. The parallel projection technique is proved to be more scalable than partition projection as partition projection saves memory space as it works well for the dataset which is dispersed, if the FP-growth tree algorithm and Tree Projection are compared on the basis of benefits it holds on, Apriori does not result to be convenient enough. The pros of FP-growth as compared to Apriori concludes to be transparent as the datasets which it contains has an enormous number of combinations of short-narrative frequent patterns. FP-growth tree implemented along with projection techniques i.e. Partition projection technique constructed to reduce execution time for constructing FP-Growth tree has to be carried out.

Keywords*- FP-Growth tree, Apriori algorithm, Association Rule, Projection techniques, frequent sequential patterns, database projection algorithms, parallel processing.*

*****

I. Introduction

FP-Tree, frequent pattern mining, in data mining breaks the Apriori bottlenecks problem.[1][3][7] In the construction procedure, without generation candidate item-set, the set of frequents occurring item-sets can be generated with the number of passes: 2 over the whole database. As compared to Apriori algorithm, FP-Growth Tree Algorithm performs a way better as support threshold is kept low, but in case of Apriori algorithm the number and length of frequently generated item-sets increases dramatically.[9][21] If the construction of FP-Growth Tree has to be carried out successfully, first task to be done would be importing a solid data-structure that can genuinely justify the requirements of the FP-Growth Tree. Still it cannot be ensured that construction of such a pattern tree will be most efficient one as there may arise problem in making combinations of the candidate generation.[10]

FP GROWTH TREE ALGORITHM[17][20]

Input:1.A transaction database DB 2.minimum support threshold ξ.
Output: FP-growth tree
Procedure:
Step-1: Scan the transactional database and find support count for each item.
Step-2: If item_id < support, discard the item.
Step-3: Construct a header table called I-list to store the sorted of frequent item-sets in declining order based on its support and node link.
Step-4: Initially, construct FP-Growth tree .in the first step, it creates the root of an FP-Growth tree and labels it as “null”. And Read the item in each transaction and created branch for each transaction. If the each node has shared a common prefix so increment by 1 otherwise create new node.
Step-5: In header table, each item points to its corresponding occurrences in the tree through a single link list it is represent by dotted lines.
Step-6: Construct the mine FP tree is call FP growth tree.

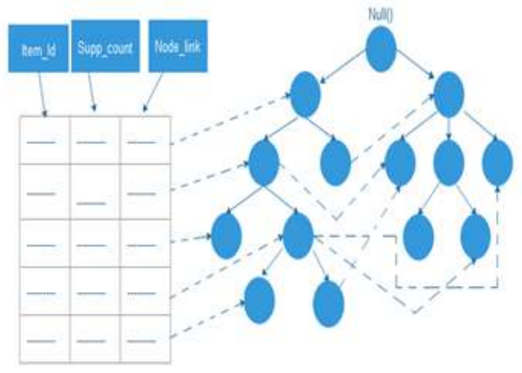

Figure: 1 Architectural view of FP-Growth Tree

	Horizontal layout based algorithms			**Vertical layout based algorithm**	**Projected layout based algorithm**	
Algorithm/ Parameter	Apriori algorithm	DHP algorithm	Partition algorithm	Eclat algorithm	FP-Tree algorithm	H- mine algorithm
Storage structure	Array based	Array based	Array based	Array based	Tree based	Tree based
Technique	Uses apriori property , join and prune method	Uses hashing technique for finding frequent itemsets	Partition the database to find the local frequent item first	Uses interaction of transaction ids for generating candidate itemset	It constructs conditional frequent pattern tree and conditional frequent pattern base	It uses hyperlink pointers to store partition projected database in the main m/m.
Memory Utilization	Large memory	Less space at initial pass and more later on	Each partition can be easily occupied in the main memory	Less memory space as compared to apriori algorithm	Less memory space due to compact structure	Memory utilized as per required for partitions of projected database
Database	Sparse as well as dense datasets	Medium database	Large database	Medium database and dense database	Large and medium datasets	Sparse as well as dense datasets

***Table: 1 Comparison between various types of algorithms with respect to its layouts*[5][11][14]**

There has been various algorithms used for frequent item-set mining.[2][19]But FP-Growth Tree algorithm works on divide and conquer strategy.[8][12] The above comparison table shows comparison between various layout based algorithms such as horizontal layout based algorithms, vertical layout based algorithms and projection layout based algorithms.

II. PARALLEL PROJECTION TECHNIQUE

- Scans the database for projection once.
- If there exist more than one program it would be executed at a time as all the projected datasets would be stored in the same memory location from where it may be retrieved easily, this procedure is termed as parallel projection.
- In the end of the scanning process, parallel projection provides parallel processing as a result of all the projected databases would be available for mining.[13][15]
- Databases which are projected will be accessible and can be easily mined in parallel but it uses a vast memory space.

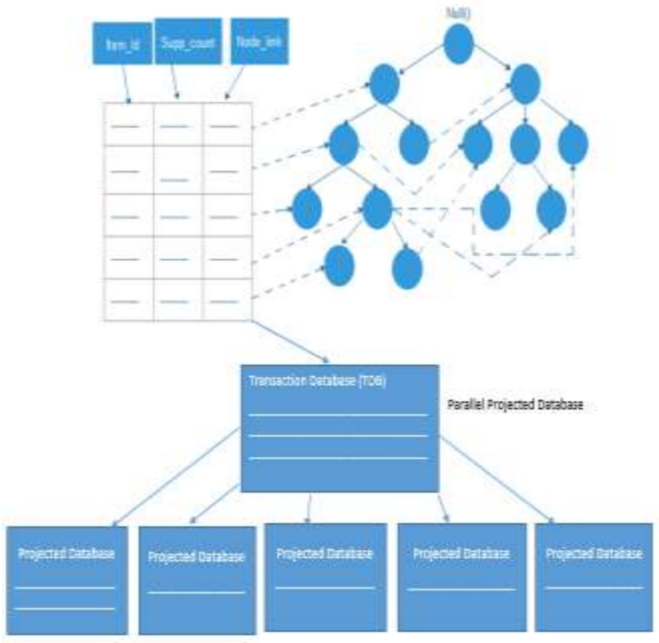

Figure: 2 Architectural view of parallel projection Database

III. PARTITION PROJECTION TECHNIQUE

- Scans the original or α-projected for carrying out projection.
- Since an operation has to be individually projected for only a single projected database scan, once scanning process of the whole database would be divided logically w.r.t the projection scheme in the formation of a set of projected segments & each segment have to be processed individually with its own local memory, this king of projection can be termed as partition projection.[16][18][21]

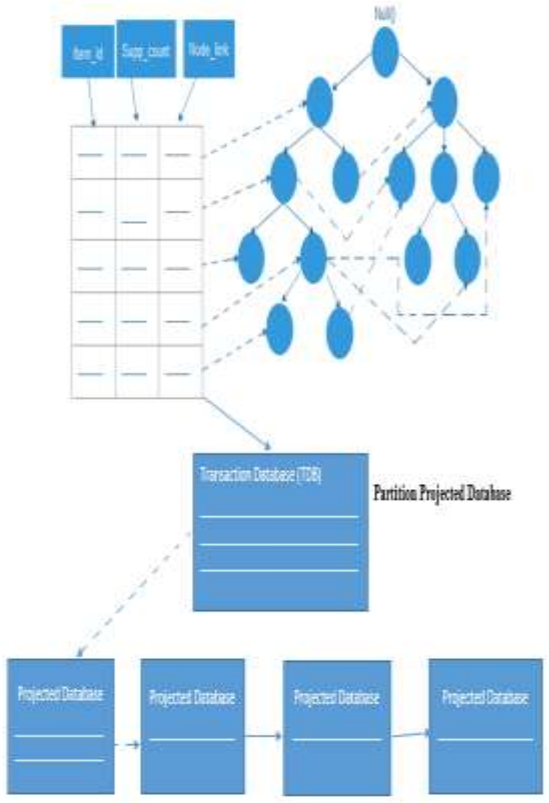

Figure: 3 Architectural view of partition projection Database

IV. IMPLEMENTATION RESULTS

For projecting the database, a projection technique is introduced to deal in the condition when the Frequent Pattern tree cannot fit in main memory. Extensive experimental results have been reported.

Experimental result shows that at some point where the size of Frequent Pattern Tree on the projection of data to generates Frequent Pattern Tree. The portion of the frequent pattern which consist shared parts would be brought together by applying single prefix structure until count registration is done. On the basis of order of frequent items two or more record accord a common prefix.

Minimum support count FP-GROWTH Tree	Execution Time (In Milliseconds)
2	130
3	124
4	88
5	73

Table 2: Minimum support Count& execution time for FP-Growth Tree Algorithm

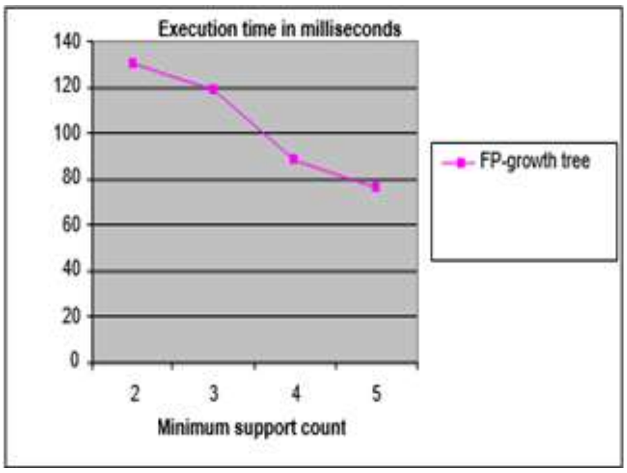

Figure 4: Graph representing execution time vs minimum support count for basic FP-Growth Tree Algorithm

Number of records	Execution Time (In millisecond)
200	87
300	98
400	144
500	199

Table 3: Represents no. of records &execution time for FP-Growth Tree Algorithm

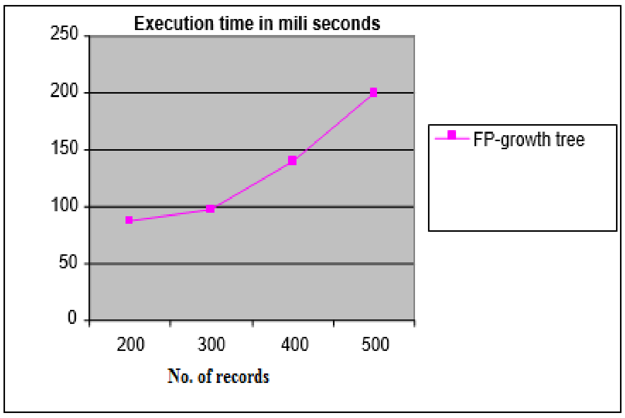

Figure 5: Graph representing execution time (in milliseconds) VS number of records for basic FP-Growth Tree Algorithm

Minimum support	Execution Time (In millisecond) **FP-Growth Tree**	Execution Time (In millisecond) **FP- Growth Tree-partition projection**
2	130	113
3	124	95
4	88	68
5	77	48

Table 4: Represents the minimum support and execution time for FP-Growth Tree & FP- Growth Tree Partition projection

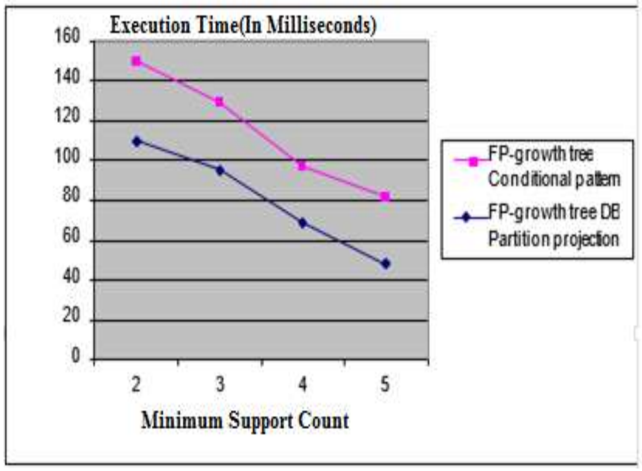

Figure 6: Graph representing the comparison of FP-Growth Tree & FP- Growth Tree with Partition projection when minimum support varying

Number of records	Execution time (In millisecond) **FP-Growth Tree**	Execution time (In millisecond) **FP- Growth Tree- partition projection**
200	87	64
300	97	77
400	140	123
500	201	188

Table 5: Comparison between FP- Growth Tree with conditional and FP- Growth Tree with Data base Partition projection technique

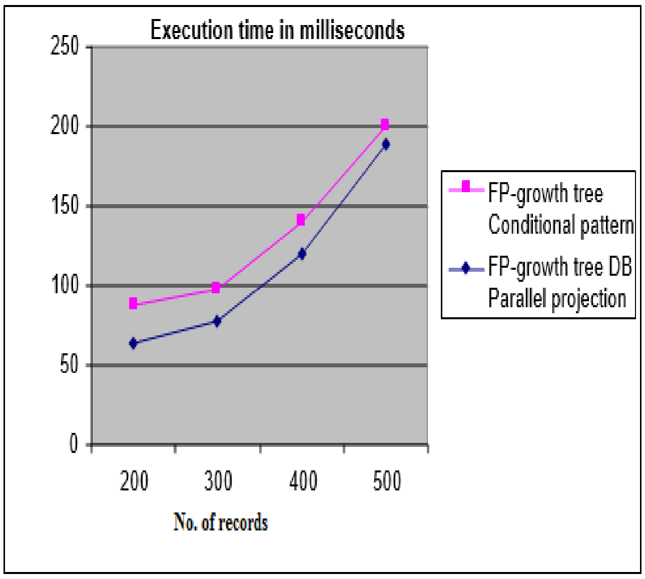

Figure 7: Representation of the comparison of FP-Growth Tree and FP- Growth Tree with Parallel projection when no. of records varying

Number of records	Execution time (In millisecond) FP-Growth Tree-partition projection	Execution time (In millisecond) FP-Growth Tree-parallel projection
200	90	83
300	101	69
400	148	134
500	204	192

Table6: Comparison between FP- Growth Tree VS FP-Growth tree with Parallel projection on the basis of number of records varying

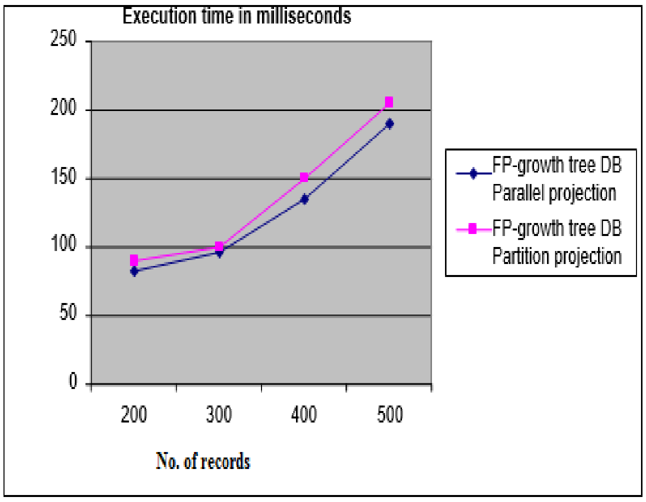

Figure 8: Graph representing the variation of FP-Growth Tree with Parallel Projection and FP-Growth Tree with Partition projection when number of records varying

Minimum Support Count	Execution time (In millisecond) **FP-Growth Tree**	Execution time (In millisecond) **FP-Growth Tree-partition projection**	Execution time (In millisecond) **FP- Growth Tree- parallel projection**
2	134	153	102
3	124	131	85
4	84	95	60
5	74	79	45

Table 7: Representation of the minimum support count and execution time for FP-GrowthTree, FP-GrowthTree parallel projection and FP-GrowthTree Partition projection

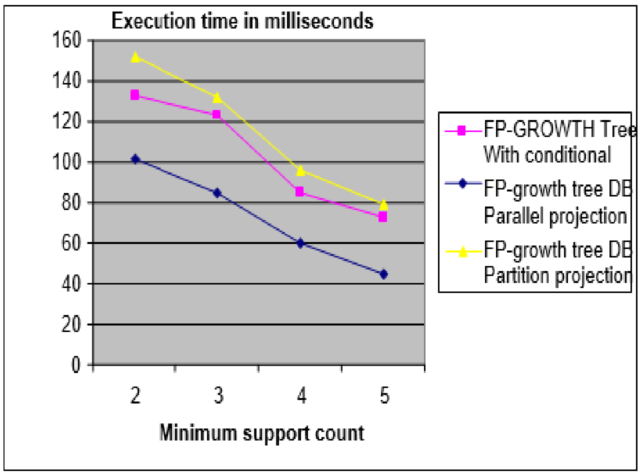

Figure 9: Graph representing the variation of FP-Growth Tree, Parallel Projection and Partition projection when minimum support count varying

V. CONCLUSION

The transactional databases are projected and the list of frequent items is mined. First the least frequent item from the parallel projected database would be mined and continue further. A unique approach has been introduced which shows significant improvement over the results. To observe the actual performance all the three techniques, it should be tested under the same environment. As it may differ in various ways, sometime the dataset may differ, sometimes the input/output devices may perform variably. A console based application has been provided here in which there is no GUI control. In this approach the selection of procedure (parallel or partition) is not dynamic. The user will decide which type of technique he wants to use.

REFERENCES

[1] R.C. Agarwal, C. Aggarwal, and V.V.V. Prasad. “A tree projection algorithm for generation of frequent itemsets”,Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining), 2000.

[2] R. Agrawal and J.C. Shafer. “Parallel mining of association rules”,IEEE Transactions on Knowledge and DataEng., 8(6):962–969, December 1996.

[3] R. Agrawal and R. Srikant. “Fast algorithms for mining association rules”, In Proc. of the 20th VLDB Conference,pages 487–499, Santiago, Chile, 1994.

[4] S. Brin, R. Motwani, and C. Silversteim” Beyond market baskets: Generalizing association rules to correlations”, In Proc. of 1997 ACM-SIGMOD Int. Conf. on Management of Data, Tucson, Arizona, 1997.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. “Introduction to Algorithms”MIT Press, McGraw-Hill, NewYork, NY, 1990.

[6] E. W. Dijkstra, W. H. Seijen, and A. J. M. Van Gasteren. Derivation of a termination detection algorithm for adistributed computation. Information Processing Letters, 16–5:217–219, 1983.

[7] Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar. Introduction to Parallel Computing: Designand

Analysis of Algorithms, 2nd Edition. Adison Wesley Publishing Company, Redwood City, CA, 2003.

[8] E.H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. IEEE Transactions on Knowledge and Data Eng., 12(3):337–352, 2000.

[9] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. of 2000 ACMSIGMODInt. Conf. on Management of Data, 2000.

[10] Mahesh V. Joshi, George Karypis, and Vipin Kumar. Universal formulation of sequential patterns. Technicalreport, Universit of Minnesota, Department of Computer Science, Minneapolis, 1999.

[11] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. Technicalreport, Department of Computer Science, University of Minnesota, 1995. Available on the WWW at URLhttp://www.cs.umn.edu/˜karypis/metis.

[12] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In Proceedings of Supercomputing, 1998. Also available on WWW at URL http://www.cs.umn.edu/˜karypis.

[13] George Karypis and Vipin Kumar. Unstructured tree search on simd parallel computers. Journal of Parallel and Distributed Computing, 22(3):379–391, September 1994.

[14] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In Proc. of the FirstInt'l Conference on Knowledge Discovery and Data Mining, pages 210–215, Montreal, Quebec, 1995.

[15] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, May 1994. Available at http://www.mpi-forum.org.

[16] Andreas Mueller. Fast sequential and parallel algorithms for association rule mining: A comparison. TechnicalReport CS-TR-3515, College Park, MD, 1995.

[17] J. Park, M. Chen, and P. Yu. An efficient parallel data mining for association rules. In Proceedings of the 4th International Conference on Information and Knowledge Management, 1995.

[18] J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. In Proc. Of 1995 ACM-SIGMOD Int. Conf. on Management of Data, 1995.

[19] J.S. Park, M.S. Chen, and P.S. Yu. Efficient parallel data mining for association rules. In Proceedings of the 4thInt'l Conf. on Information and Knowledge Management, 1995.

[20] S. Parthasarathy, M. Zaki, M. Ogihara, and W. Li. Parallel data mining for association rules on shared-memorysystems. Knowledge and Information Systems, 3(1):1–29, 2001.

[21] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Ping, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu.Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings 2001 InternationalConference on Data Engineering, 2005.