International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

The Parallelization of Algorithm based on Horizontal Partition for Association Rule Mining on Large Data Set

Anil Vasoya
Ph. D Scholar
Sant. Gadge baba Amravati University, Amravati
Maharashtra, India

Dr. Nitin Koli
Deputy Registrar
Sant. Gadge baba Amravati University, Amravati
Maharashtra, India

***Abstract*:- Data mining can abstract important facts such as frequent item set from large data set – but sometimes it is difficult to achieve all frequent item set if these datasets are split into many clusters when there is a large dataset. In this paper, a detailed comparison has been made for the frequent pattern mining algorithms such as Apriori, ECLAT and FP- Growth based on without partitioning and with horizontal partitioning of dataset by applying actual minimum support as well as with dynamic minimum support. Experimental results shows that after partitioning also we can maintain similar accuracy and have obtained 100% frequent item set in contrast with frequent item set achieved without partitioning. This novel concept is useful to search frequent item set from the large dataset when we applied the concept of partitioning. It also shows that till which extent minimum support should be decreased with respect to number of cluster to achieve maximum accuracy or 100% frequent item set.**

Keywords: Association rule Mining, Frequent Items Data Set, Apriori Algorithm, ECLAT, FP- Growth, Minimum support, parallel computing.

I. INTRODUCTION

Data mining is concentrate on discovering interesting hidden relationships in large databases. It provides a tool for knowledge discovery to extract unknown and potentially interesting knowledge from data set [1]. One amongst the important techniques utilized in DM is association rules. Association rule is employed to finds interesting patterns among large set of knowledge items, which is an important task of information mining [2, 3].

Finding frequent itemsets in transaction databases has been proven to be useful in retail business applications [6][11]. Many algorithms are proposed to hunt out frequent itemsets from a very large database. However, there is no published implementation that outperforms every other implementation on every database with every different support [7]. Generally, many implementations are supported these main algorithms: Apriori [2], frequent pattern growth (FP-growth) [4] and ECLAT. The Apriori algorithm discovers the frequent itemsets from an awfully large database through a series of iterations. The Apriori algorithm is required to huge candidate itemsets, compute the support, and prune the candidate itemsets to the frequent itemsets in each iteration. The FP-growth algorithm discovers frequent itemsets with less the time.

The Apriori algorithm has been look over in numerous ways. One review of the Apriori algorithm is to partition a transaction database into divide partitions DB1, DB2, DB3, …, DBn. Partitioning a transaction database may improve the performance of frequent itemsets mining by adjust each partition into main memory for fast access and allowing incremental generation of frequent itemsets. Our implementation is a partition whole dataset in different number of clusters horizontally and generate frequent item set using Apriori algorithm, FP_Growth and ECLAT using parallel computing.

It observed that once large dataset partition into many cluster then it lost many important frequent item set because of many reason like customer buying pattern, sessional product requirement, seller discount offer etc. Although many algorithms have been proposed, the execution accuracy of frequent pattern mining is still a challenge to the researchers due to the data outburst. Later section of this paper we will see efficient parallel computing approach using dynamic minimum support.

The rest of this paper is organized as follows:

- Related work is presented in section II.
- Proposed methodology is presented in section III.
- Results and Discussion are presented in Section IV.
- Conclusions and Future work are discussed in Section V.

II. LITERATURE REVIEW

In order to improve the performance of association rule mining, many researchers tried to distribute dataset into the mining computation over more than one processor or a traditional computer have multicore for executing a task. One way of increasing the computational speed is by using all core within a single computer.

The most famous is the Apriori algorithm which has been brought in 1993 by Agrawal et al. [1] which uses association rule mining[2][3][5] [6][8][9][10]. Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps: 1. Minimum support (threshold) is applied to find all frequent item-sets in a database. 2. These frequent item-sets and the minimum confidence constraints are used to form rules. Advantage of this algorithm, it is easy to find frequent item sets if database is small

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

but it has two deadly bottlenecks. First, It needs great I/O load when frequently scans database and Second, It may produce overfull candidates of frequent item-sets.

Another accomplishment in the development of association rule mining and frequent pattern mining is FPGrowth Algorithm which overcomes the two deficiencies of the Apriori Algorithm . Efficiency of FP-Growth is based on three salient features: (i) A divide-and-conquer approach is used to extract small patterns by decomposing the mining problem into a set of smaller problems in conditional databases, which consequently reduces the search space (ii) FP-Growth algorithm avoid the complex Candidate Itemset generation process for a large number of candidate Itemsets, and (3) To avoid expensive and repetitive database scan, database is compressed in a highly summarized, much smaller data structure called FP tree [4].

Equivalence class clustering and bottom up lattice traversal (ECLAT) [7] with vertical data format uses intersection of transaction ids list for generating candidate Itemset. Each item is stored with its list of Transaction ids instead of mentioning transaction ids with list of items This algorithm uses breath first search (BFS) and the generation of candidate itemset is needed. Equivalence class clustering and bottom up lattice traversal (ECLAT) algorithm accommodates 'Depth First Search' approach and requires the generation of candidate itemset.

Proposed a system to reduce the general items and items to be assembled and consequently improve sales in supermarkets. The approach is particularly suitable for analyzing customer performance in relation to purchases of goods and improving the point of sale in supermarkets. An example is customer buying habits that use the Belgian shopping market database, where purchase data, prices and price are used to determine customer behavior and display related information to support sales. In results it has been shown that the time and memory required for the generation of raw materials is reduced by the generation of well-made rules [11].

Develop a system which proposed for frequent item set mining, where author developed parallel analytical model by integrating R on hadoop [12]. It shows that as the frequent item set is important for inference engine and for analysis it is important to resolve the problems which arise from large dataset. The proposed system takes data into a preprocessing optimizer and it runs it on multistage algorithm. it uses Map reduce application along with some association rules. It showed that a same list is used for operations of mapreduce and is handled with multiple algorithms to give minimized output. One of the drawbacks can be that it doesn't support cyclic flow of data or increamental data.

Proposed inverted matrix which is a new disk based Parallel association rule mining with minimum inter-processor communication [13]. Apriori algorithm which works on a shared nothing algorithm is not scalable. The inverted matrix has two phases where a global frequent pattern is achieved where the transactional dataset is converted into inverted matrix. The dataset is replicated among many parallel nodes using different support level of association rules, by doing so a global frequent pattern can be generated without communication between node and the load is balanced. The inverted matrix uses combines both horizontal layout and vertical layout and makes the best of both the approaches. The inverted matrix is based on COFI tress. The results of this algorithm acquired 8 nodes with 16 clusters. The paper proposes that we need to have embarrassingly parallel algorithms to store massive side and also to allow random access, and also to allow no communication between nodes.

Emphasized on the importance of finding frequent item set of large dataset. One of the most effective algorithim to find frequent item set is the Apriori which trims the candidate item set and also finds the support. Hash tress is used in apriori algorithm. The focus of this paper is based on implementing apriroi algorithm using trie structure and analyze its performance of parallel processor. Partitioning helps in quick access and allows the itemsets to be generated incrementally. Trie structure is an effective way to store itemset and to access them . A trie gets created incrementally based on the count of item set . This implementation solved most of the problems of frequent mining faster than most of the implementations but had an issue in load balancing[14].

K. W. Lin and Y. Luo propose a plan to find fast frequent itemset in cloud computing by utilizing cloud nodes and also prevent the data from getting leaked. The data getting leaked becomes an issues because when we divide the dataset and each processor is distributed with a part to mine , there is an exchange among nodes which increases the load and the database is duplicated to every node. This framework has a kernel cloud which can access the database and has all the information related to the available nodes. As the FP-tree stores data in compressed form as compare to apriori algorithm.

III. PROPOSED METHODOLOGY

It is difficult to generate frequent item set from large data set in any existing algorithms because existing algorithms create large candidate set hence it's impossible to fit into primary memory. This issue can be resolve with the help of parallel computing concept. Divide dataset into many clusters so that it is easy for CPU to generate FIS and it will not face any issue related to memory.

Problem Identified:

As we know, when dataset divides into many clusters or partitioned the dataset it will lost many FIS in process of partition processing. Hence, we proposed a concept dynamic minimum support where it will decrease the minimum support up to certain limit with respect to number of cluster and actual minimum support and next section will show with the help of dynamic support we can achieve almost similar support that we achieved with out clustering.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

IV RESULT AND DISCUSSION

Table 1 shows % of decrease in Dynamic minimum support value w.r.t No. of cluster and actual minimum support.

Table 1: % of decrease in Dynamic minimum support value w.r.t No. of cluster and actual minimum support.

No. Of Clusters	% Dynamic Min. Support
1	100.00
3	67.70
5	58.86
7	54.20
9	51.17
11	48.99
13	47.30
15	45.95
17	44.83
21	43.06
23	42.34
25	41.70
27	41.13
31	40.14
33	39.71
35	39.31
37	38.94
39	38.59

Fig 1 shows the deviation of Dynamic Minimum Support over Static Minimum Support with increase in No. of clusters. Dynamic Min. Support is calculated as,

Dynamic Min. Support = Actual Min. Support / Log $_{10}$ (No. of Clusters)

It indicates the logarithmic decrease in Dynamic Minimum support in order to achieve higher accuracy of FIS in contrast to lower accuracy of FIS with Static Minimum Support over clustering for Horizontal Approach.

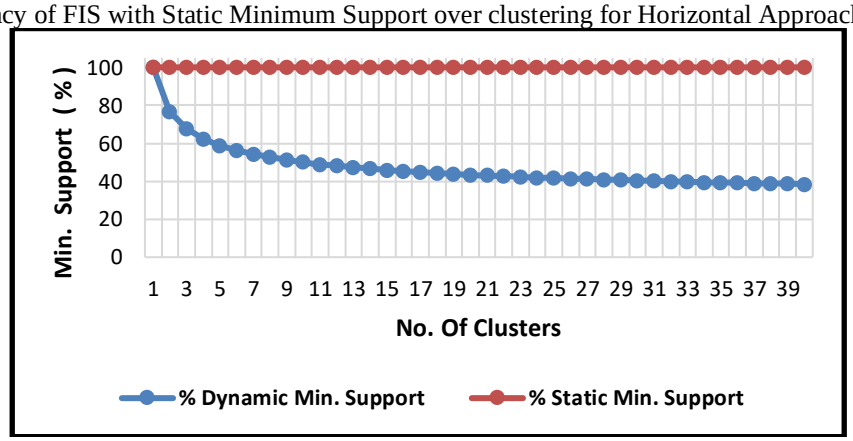

Fig. 1: Variation of Dynamic / Static minimum support (%) w.r.t No. of cluster.

The Table 2 shows execution time comparison of Normal approach (without clustering) w.r.t Horizontal Approach (with Dynamic / Static Min. support and clustering) for Apriori algorithm over 1 Lakh to 10 Lakh transaction with Minimum support = 0.01 and No. of clusters = 12.

Table 2: Execution Time required by Apriori algorithm with clustering (Dynamic / Static Min. support) and without clustering.

No. Of Transactions	Min. Support	No. Of Clusters	Execution time of Apriori Algorithm (sec)		
			Normal	Static Support	Dynamic Support
100000	0.1	12	89	53	345
200000	0.1	12	186	101	529
300000	0.1	12	349	153	738
400000	0.1	12	452	207	994
500000	0.1	12	576	246	1362
600000	0.1	12	686	273	1650
700000	0.1	12	728	322	1843
800000	0.1	12	795	379	2062
900000	0.1	12	1091	438	2191
1000000	0.1	12	1201	471	2260

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

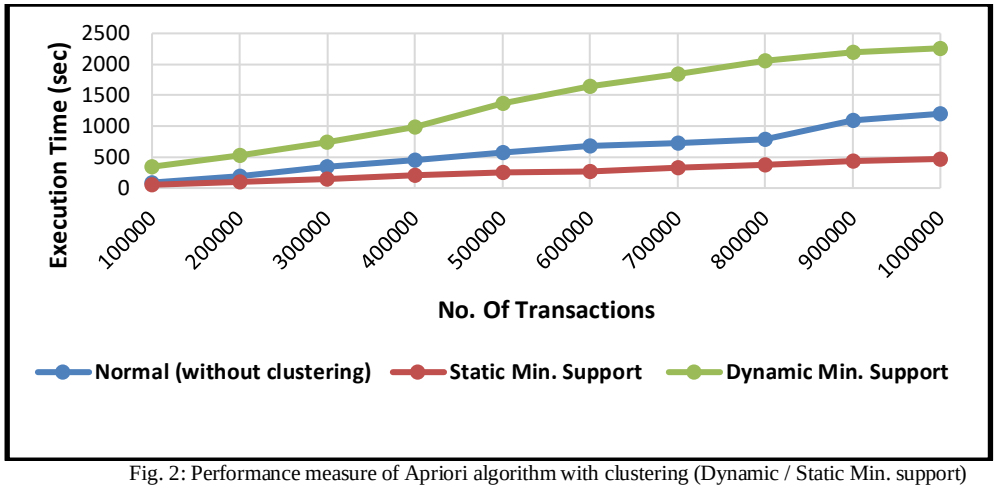

Fig. 2: Performance measure of Apriori algorithm with clustering (Dynamic / Static Min. support) and without clustering.

Fig. 2 shows that Apriori algorithms requires much lesser execution time with static Min. support and in Normal approach in contrast with Dynamic Min. support. This is due to much higher No. of candidate set generated by Apriori algorithm at lower Dynamic support value.

The Table 3 shows the comparison of No. of Patterns generated in Normal approach (without clustering) w.r.t No. of patterns generated in Horizontal approach (with Dynamic / Static Min. support and clustering) for Apriori algorithm over 1 Lakh to 10 Lakh transaction with Minimum support = 0.01 and No. of clusters = 12.

Table 3: No. of Patterns generated by Apriori algorithm with clustering (Dynamic / Static Min. support) without clustering.

No. Of Transactions	Min. Support	No. Of Clusters	No. Of Patterns (Apriori Algorithm)		
			Normal (without clustering)	Static Support	Dynamic Support
100000	0.1	12	115	88	115
200000	0.1	12	115	90	115
300000	0.1	12	116	93	116
400000	0.1	12	116	94	116
500000	0.1	12	117	98	117
600000	0.1	12	116	100	116
700000	0.1	12	118	100	118
800000	0.1	12	118	101	118
900000	0.1	12	119	102	119
1000000	0.1	12	118	102	118

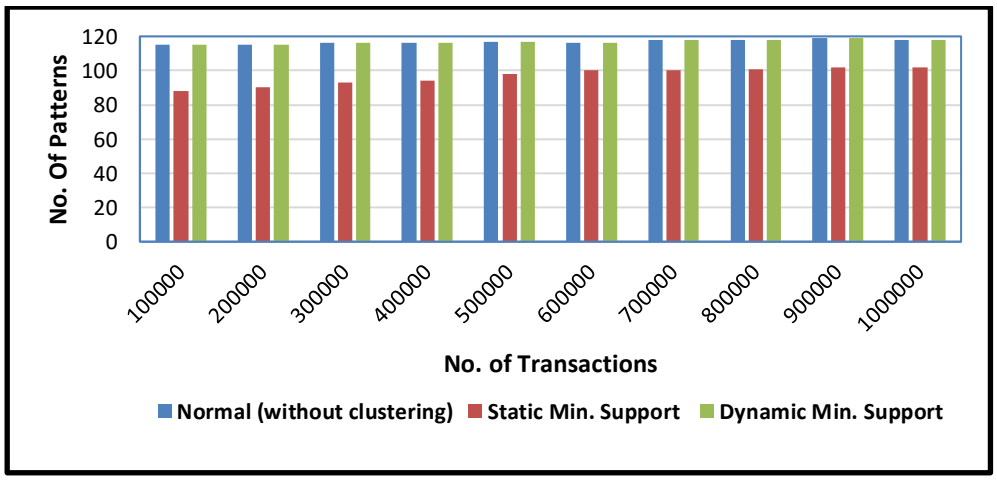

Fig. 3: No. of Patterns generated over 1L-10L Transactions. (Apriori Algorithm)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

Fig. 3 shows that for Apriori algorithm the accuracy of No. of Patterns generated with Dynamic Min. support is almost equal to 100% while with Static Min. support it drops to approx. 75-85% in contrast with No. of Patterns generated in Normal approach(without clustering).

The Table 4 shows execution time comparison of Normal approach (without clustering) w.r.t Horizontal Approach (with Dynamic / Static Min. support and clustering) for FP-Growth algorithm over 1 Lakh to 10 Lakh transaction with Minimum support = 0.01 and No. of clusters = 12.

Table 4: Execution Time required by FP-Growth algorithm with clustering (Dynamic / Static Min. support) and without clustering.

No. Of Transactions	Min. Support	No. Of Clusters	Execution time of FP-Growth Algorithm (sec)		
			Normal	Static Support	Dynamic Support
100000	0.1	12	32	15	193
200000	0.1	12	88	28	41
300000	0.1	12	165	56	63
400000	0.1	12	242	68	72
500000	0.1	12	336	70	86
600000	0.1	12	438	90	116
700000	0.1	12	607	106	131
800000	0.1	12	726	109	136
900000	0.1	12	847	123	139
1000000	0.1	12	995	140	144

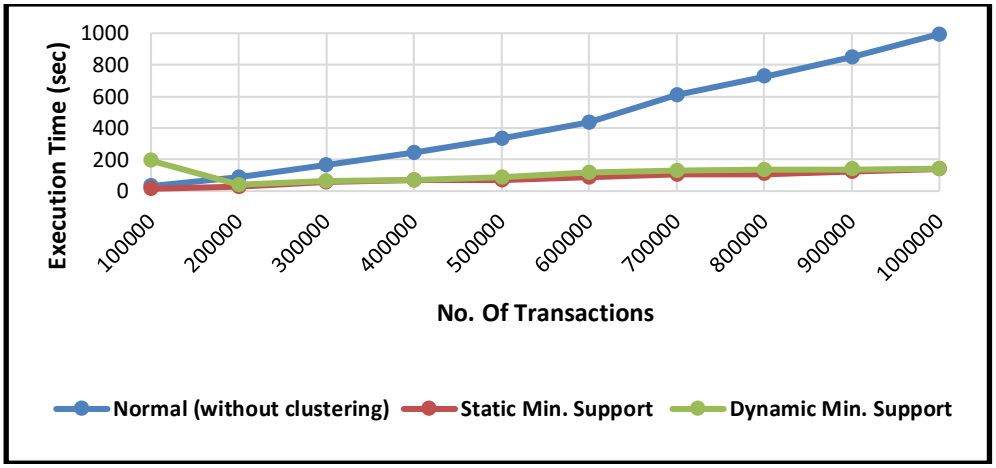

Fig. 4: Performance measure of FP-Growth algorithm with clustering (Dynamic / Static Min. support) and without clustering.

Fig. 4 shows that FP-Growth algorithms requires much lesser execution time in Horizontal approach (with Dynamic / Static Min. support and clustering) in contrast with Normal approach. This is due to clustering which results into smaller tree structures in Horizontal approach with lesser branching factor.

The Table 5 shows the comparison of No. of Patterns generated in Normal approach (without clustering) w.r.t No. of patterns generated in Horizontal approach (with Dynamic / Static Min. support and clustering) for FP-Growth algorithm over 1 Lakh to 10 Lakh transaction with Minimum support = 0.01 and No. of clusters = 12.

Table 5: No. of Patterns generated by FP-Growth algorithm with clustering (Dynamic / Static Min. support) and without clustering.

No. Of Transactions	Min. Support	No. Of Clusters	No. Of Patterns (FP-Growth Algorithm)		
			Normal (without clustering)	Static Support	Dynamic Support
100000	0.1	12	111	84	96
200000	0.1	12	111	87	96
300000	0.1	12	112	90	98
400000	0.1	12	112	91	98
500000	0.1	12	113	95	98
600000	0.1	12	112	97	99

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

700000	0.1	12	114	97	99
800000	0.1	12	114	98	99
900000	0.1	12	115	99	100
1000000	0.1	12	114	99	99

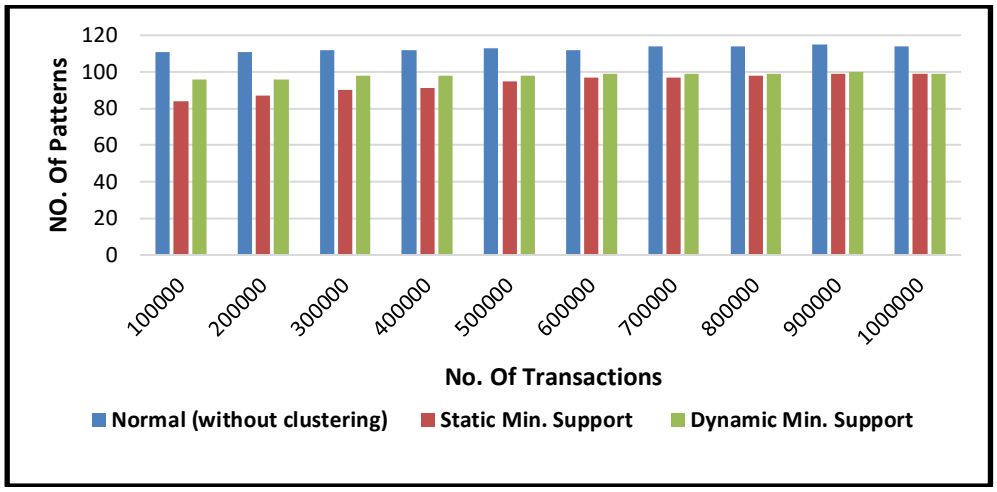

Fig. 5: No. of Patterns generated over 1L-10L Transactions. (FP-Growth Algorithm)

Fig. 5 shows that for FP-Growth algorithm the accuracy of No. of Patterns generated with Dynamic Min. support is around 85% while with Static Min. support it drops to approx. 75% in contrast with No. of Patterns generated in Normal approach(without clustering).

Table 6: Execution Time required by ECLAT algorithm with clustering (Dynamic / Static Min. support) and without clustering.

No. Of Transactions	Min. Support	No. Of Clusters	Execution time of ECLAT Algorithm (sec)		
			Normal	Static Support	Dynamic Support
100000	0.1	12	31	17	55
200000	0.1	12	64	34	63
300000	0.1	12	93	52	80
400000	0.1	12	137	63	92
500000	0.1	12	169	70	106
600000	0.1	12	141	91	109
700000	0.1	12	183	104	114
800000	0.1	12	219	119	118
900000	0.1	12	285	137	120
1000000	0.1	12	355	152	126

The Table 6 shows execution time comparison of Normal approach (without clustering) w.r.t Horizontal Approach (with Dynamic / Static Min. support and clustering) for ECLAT algorithm over 1 Lakh to 10 Lakh transaction with Minimum support = 0.01 and No. of clusters = 12.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

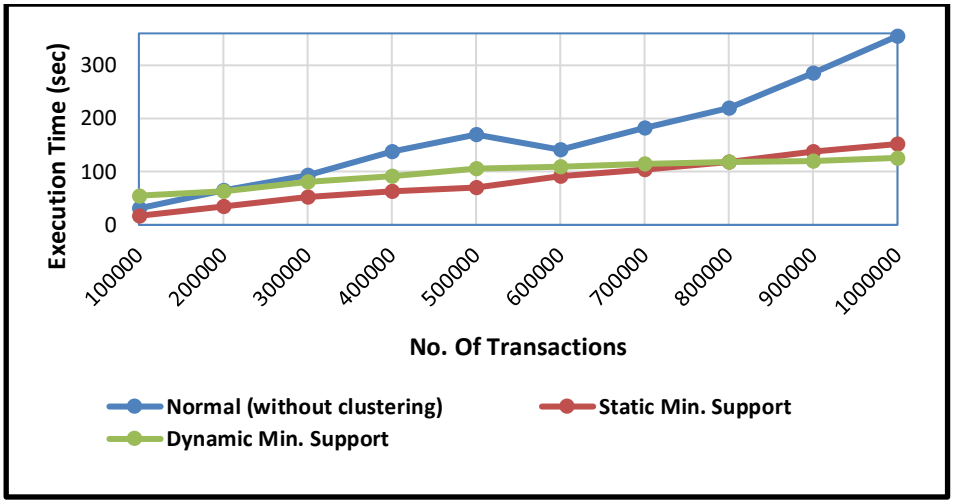

Fig. 6: Performance measure of ECLAT algorithm with clustering (Dynamic / Static Min. support) and without clustering.

Fig. 6 shows that ECALT algorithms requires a bit lesser execution time in Horizontal approach with Dynamic / Static Min. support using clustering in contrast with Normal approach.
The Table 7 shows the comparison of No. of Patterns generated in Normal approach (without clustering) w.r.t No. of patterns generated in Horizontal approach (with Dynamic / Static Min. support and clustering) for ECLAT algorithm over 1 Lakh to 10 Lakh transaction with Minimum support = 0.01 and No. of clusters = 12.

Table 7: No. of Patterns generated by ECLAT algorithm with clustering (Dynamic / Static Min. support) and without clustering.

No. Of Transactions	Min. Support	No. Of Clusters	No. Of Patterns (ECLAT Algorithm)		
			Normal (without clustering)	Static Support	Dynamic Support
100000	0.1	12	115	88	115
200000	0.1	12	115	90	115
300000	0.1	12	116	93	116
400000	0.1	12	116	94	116
500000	0.1	12	117	98	117
600000	0.1	12	116	100	116
700000	0.1	12	118	100	118
800000	0.1	12	118	101	118
900000	0.1	12	119	102	119
1000000	0.1	12	118	102	118

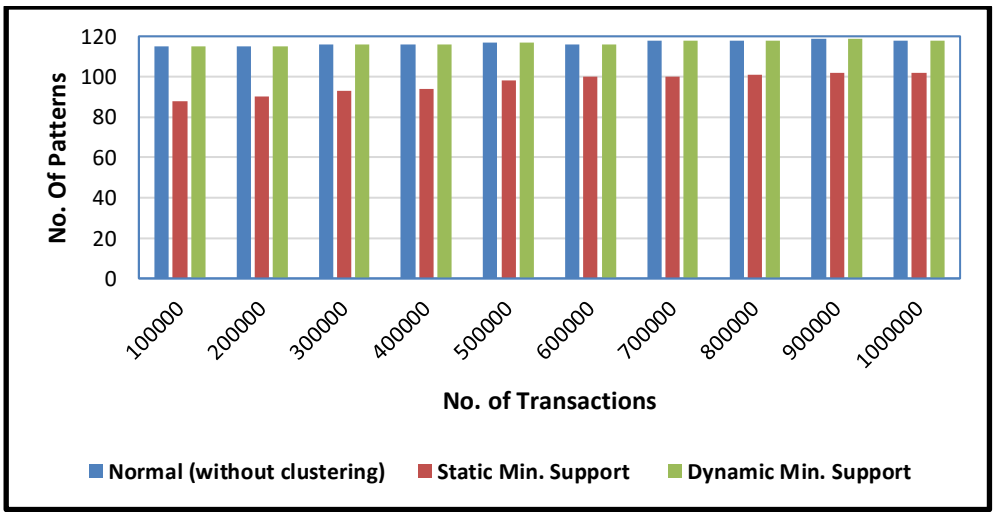

Fig. 7: No. of Patterns generated over 1L-10L Transactions. (ECLAT Algorithm)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 12, December-2020

Fig.7 shows that for ECLAT algorithm the accuracy of No. of Patterns generated with Dynamic Min. support is almost equal to 100% while with Static Min. support it drops to approx. 75-85% in contrast with No. of Patterns generated in Normal approach (without clustering).

V. CONCLUSION

It conclude that in all above algorithms accuracy (FIS) sustained by Dynamic Min. support is almost equal to 100% whereas with Static Min. support it drops to approx. 80% with an increase in No. of clusters. In future execution time can be reduced with the help of additional resources i.e. Multiprocessor environment. Also this research can be extend in future to different category of dataset and with more number of transactions.

REFERENCES

[1] Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA, May 1993.
[2] Gosain and M. Bhugra, "A comprehensive survey of association rules on quantitative data in data mining," 2013 IEEE Conference on Information & Communication Technologies, Thuckalay, Tamil Nadu, India, 2013, pp. 1003-1008.
[3] S. P. Aditya, M. Hemanth, C. K. Lakshmikanth and K. R. Suneetha, "Effective algorithm for frequent pattern mining," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, 2017, pp. 704-708.
[4] Jian Pei, Jiawei Han, "Mining Frequent patterns without candidate generation," in SIGMOD Proceedings of the 2000 ACM SIGMOD international conference on Management of data, New York, NY, USA, 2000, pp. 1-12.
[5] S. Qianxiang and W. Ping, "Association rules mining based on improved PSO algorithm," 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), Beijing, 2017, pp. 145-149.
[6] C. Yadav, S. Wang and M. Kumar, "An approach to improve apriori algorithm based on association rule mining," 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, 2013, pp. 1-9.
[7] Mohammed J. Zaki, "Scalable Algorithms for Association Mining," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, pp. 372-390, 2002.
[8] L.Wang et al., "Efficient Mining of Frequent Item Sets on Large Uncertain Databases, "IEEE Transactions on Knowledge and Data Engineering, vol.24, no. 12, pp. 2170 – 2183,Dec. 2012.
[9] Anil Vasoya, Novel Approach to Improve Apriori Algorithm using Transaction Reduction and Clustering Algorithm, International Journal of Applied Information Systems (IJAIS), pp. 37-44, 2014.
[10] Anil Vasoya, Dr. Nitin Koli,"Mining of association rules on large database using parallel and distributed computing", 7th International Conference on Communication, Computing and Virtualization 2016, Procidia, Computer science, pp.221 – 230,2016
[11] Yanbin Ye and Chia-Chu Chiang, "A Parallel Apriori Algorithm for Frequent Itemsets Mining," Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), Seattle, WA, 2006, pp.87-94.
[12] K. Poorva, H. K. Anushree, K. V. Mahesha, T. R. Pavithra, D. C. Vinutha and S. B. chandini, "Parallel Analytical Model for Frequent Itemset Mining," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 517-519.
[13] M. El-Hajj and O. R. Zaiane, "Parallel association rule mining with minimum inter-processor communication," 14th International Workshop on Database and Expert Systems Applications, 2003. Proceedings., Prague, Czech Republic, 2003, pp. 519-523.
[14] S. Einakian and M. Ghanbari, "Parallel implementation of association rule in data mining," 2006 Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory, Cookeville, TN, 2006, pp. 21-26.