ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

FI-DBSCAN: Frequent Itemset Ultrametric Trees with Density Based Spatial Clustering Of Applications with Noise Using Mapreduce in Big Data

V.Swathi Kiruthika[1], Dr.V.Thiagarasu[2]

M.Phil Scholar, Dept. of Computer Science, Gobi Arts & Science College, Gobichettipalayam, Tamilnadu, India

Associate Professor, Dept. of Computer Science, Gobi Arts & Science College, Gobichettipalayam, Tamilnadu, India

ABSTRACT: Data mining is gaining importance due to huge amount of data available. Retrieving information from the warehouse is not only tedious but also difficult in some cases. The existing algorithm does not provide fast computation and better result. Frequent itemset using density based spatial clustering is used in the proposed system so that support is counted by mapping the items from the candidate list into the buckets which is divided according to support known as Hash table structure. As the new itemset is encountered if item exist earlier then increase the bucket count else insert into new bucket. Thus in the end the bucket whose support count is less the minimum support is removed from the candidate set.

KEYWORDS: Bigdata, Hadoop, MapReduce, HashTables, HashFunction, Frequent Itemsets, DBSCAN, MinPts.

I. INTRODUCTION

As a volume of database increases day by day traditional frequent itemset mining algorithms becomes inefficient. As a solution to this problem parallel mining of frequent itemsets using DBFIUT (Density Based Frequent Itemset Ultrametric Tree) algorithm is implemented on MapReduce framework. More importantly, in the existing parallel-FIUT algorithms lack a mechanism that enables automatic parallelization, load balancing, data distribution and fault tolerance on large computing cluster. Therefore, in the proposed system using DBFIUT algorithm rather than traditional FIU- Tree algorithm because to avoid building conditional patterns and to achieve compressed storage and to detect the noise using DBSCAN (Density Based Spatial Clustering of Application with Noise) algorithm. Proposed methodology builds this using Hadoop framework. The working flow of DBFIUT algorithm on MapReduce framework consists of three MapReduce job. Synthetic datasets are used for the experimental analysis.

II. LITERATURE REVIEW

Min Chen [2014] proposed an Apriori algorithm using MapReduce framework. Apriori is a classic algorithm to generate huge amount of candidate itemsets by repeating scanning of the database. To improve the performance of the Apriori algorithm, MapReduce parallelization techniques has been proposed. However it has some disadvantages that processor has to scan the database multiple times and to exchange an excessive number of candidate itemsets with other processor. Apriori like frequent itemsets mining algorithm suffer from I/O overhead and synchronization overhead.

Riondato et al (2012) proposed a parallel randomized algorithm for approximate association rules mining in MapReduce. PARMA algorithm creates multiple small random samples of the transactional dataset and running an algorithm samples independently and in parallel. The final collections of frequent itemsets or association rules from samples are aggregated and filtered to provide an output. In this algorithm final result is an approximation of the exact output. PARAM algorithm is implemented in parallel MapReduce framework.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

Sagiroglu, S.; Sinanc, D. (20-24 May 2013),"Big Data: A Review" describe the big data content, its scope, methods, samples, advantages and challenges of Data. The critical issue about the Bigdata is the privacy and security. Big data samples describe the review about the atmosphere, biological science and research life sciences etc. By this paper, it concludes that any organization in any industry having big data can take the benefit from its careful analysis for the problem solving purpose. Using Knowledge Discovery from the Bigdata it is easy to get the information from the complicated data sets.

Puneet Singh Duggal, Sandulescu, V. ; Halcu, I. ; Neculoiu, G. ;(17-19 Jan. 2013),"A Big Data implementation based on Grid Computing", Grid Computing offered the advantage about the storage capabilities and the processing power and the Hadoop technology is used for the implementation purpose. Grid Computing provides the concept of distributed computing. The benefit of Grid computing center is the high storage capability and the high processing power. Grid Computing makes the big contributions among the scientific research, help the scientists to analyze and store the large and complex data.

Zhou (2010) proposed a FP-Growth with MapReduce framework. Frequent itemsets mining plays an essential role in data mining tasks. As a volume of data sets increases day by day, most of the existing parallel mining algorithms running on a single machine suffer from performance degradation. To improve the performance of the FP-Growth algorithm, many parallelization techniques have been proposed. A solution to this problem Frequent Pattern growth algorithm is implemented using MapReduce framework. FP growth has a disadvantage that it wants to scan the database multiple times and the recursive traversing of tree increases the computing time when it comes to multidimensional database.

K.W. Lin (2011) proposed a novel frequent pattern mining algorithms for very large database in cloud computing environments. As a volume of data increases or the minimum support count decreases however both the execution time and the memory requirement increases greatly. A solution to this problem distributed computing techniques has been utilized to improve the scalability and execution efficiency. This paper proposed a method to discover the frequent itemsets from large database using cloud computing techniques. In this paper construction and storage of FP tree using disk as the secondary memory. FP-Tree from the disk is proposed and it improves the performance in terms of efficiency and scalability.

Yu K. and Zhou Y (2010) proposed a DH-TRIE Frequent Pattern Mining on Hadoop using JPA. The FP growth is a traditional frequent itemsets mining algorithm in data mining when working with large volume of datasets and high dimensional datasets. FP growth has a disadvantage that it wants to scan the database multiple times and the recursive traversing of tree increases the computing time when it comes to multidimensional database. In this paper a distributed DH-TRIE frequent itemset mining algorithm is proposed based on Hadoop, the open cloud computing model, which solved the three problems (random-write, globalization and duration). Flexibility and scalability are improved in this algorithm.

P. Viswanath and V. S. Babu proposed that there are number of clustering techniques but this paper mainly focuses on the widely used DBSCAN clustering density based approach where the efficiency of GenClus in detecting quality clusters over gene expression data. This work presents a density based clustering approach which finds useful subgroups of highly coherent genes within a cluster and obtains a hierarchical structure of the dataset where the sub clusters give the finer clustering of the dataset. This approach ultimately forms a distributed clustering algorithm. This approach helps to cluster the data locally and independently from each other and transmitted only aggregated information about the local data to a central server. Grid-based DBSCAN clustering technique quantizes the data set in to a no of cells and then work with objects belonging to these cells. The merging of grids and consequently clusters does not depend on a distance measure. It is determined by a predefined parameter.

III. PROPOSED ALGORITHM

FIUT with DB SCAN can be derived from apriori by introducing additional control. To this purpose of proposed work makes use of an additional hash table that aims at limiting the generation of candidates in set as much as possible. The proposed system also progressively trims the database by discarding attributes in transaction or even by discarding entire transactions when they appear to be subsequently useless. The actual FIUT approach is not suitable for a large multidimensional database which is frequently updated. In that case, the incremental clustering approach is much better. The system with incremental concept saves lot of time and effort efficiently whereas the existing system has already suffering with some drawbacks and these problems are mainly faced in dynamic large databases by the existing system. When some records are added to existing data, then it deals with this problem of scanning the whole database again. The time complexity is very high due to rescanning the whole database. It requires more effort as well. The Existing system is not efficient with respect to time and effort. That's why new system with incremental clustering approach using DBSCAN with FIUT is more suitable to use in a large multidimensional dynamic database.

In proposed method, support is counted by mapping the items from the candidate list into the buckets which is divided according to support known as Hash table structure. As the new itemset is encountered if item exist earlier then increase the bucket count else insert into new bucket. Thus in the end the bucket whose support count is less the minimum support is removed from the candidate set. In computing, a **hash table** is used which is a data structure used to implement an associative array, a structure that can map keys to values. A **hash function** is used in hash table to compute an index into an array of buckets or slots, from which the desired value can be found.

DBSCAN (Density Based Spatial Clustering of Applications with Noise)

One of the most common clustering algorithms and also most cited in scientific literature is Density Based Spatial Clustering of Applications with Noise (DBSCAN) which has the ability to produce arbitrary shape of clusters. Clusters are identified by looking at the density of points. Regions with a high density of points depict the existence of clusters whereas regions with a low density of points indicate clusters of noise or clusters of outliers. DBSCAN grows clusters according to a density based connectivity analysis. It defines a cluster as a maximal set of density-connected points. The key idea of density-based clustering is that for each object of a cluster the neighbourhood of a given radius has to contain at least a minimum number of objects (MinPts), i.e. the cardinality of the neighbourhood has to exceed some threshold. The algorithm DBSCAN was designed to efficiently discover the clusters and the noise in a database according to the above definitions. DBSCAN requires two parameters: ε (eps) and the minimum number of points required to form a cluster (Minpts). It starts with an arbitrary starting point that has not been visited. This point's ε-neighbourhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labelled as noise.

Advantages of Proposed System

- Additional hash table is added into the proposed system which reduces the database scan process than the existing system.
- Three mapreduce job modules are involved in the proposed methodology for mining all the frequent itemsets.
- It reduces the database scan than the existing system due to the division of three modules.
- Minimum support count plays the important role in the proposed system. Therefore, the running time of the proposed algorithm is reduced.
- Scalability of the proposed algorithm is higher when it comes to parallel mining of an enormous amount of data.

Frequent Itemset Ultrametric Tree with Dbscan

DB-FIUT is a new method for mining frequent itemsets from the database. DB-FIUT has four major advantages over traditional FIU-tree like: it involves only two round of scanning which minimizes I/O overhead. Then the DB-FIUT is a highly improved way to partition a database, which considerably reduces the search space by adding additional hash tables using hash functions. Next is here only frequent items in each transaction are inserted as nodes into the DB-FIUT for compressed storage. At last all frequent itemsets are generated without traversing the tree recursively by checking the leaves of each DB-FIUT which reduces computing time significantly. DB-FIUT consists of two phases to generate the frequent itemsets from the transactions by two rounds of scanning the database.

1 Generating one Itemsets and K Itemsets

Phase1 consists of two round of scanning the database. At the first round of scanning the database frequent one item will be generated based on the minimum support count. At the second round of scanning the database all k-items will be generated by pruning the infrequent items from each transaction.

2 Generating Frequent K Itemsets

Phase2 consists of a two process decompose each 'h' itemsets into 'k' itemsets. After decomposing process the repetitive construction of K-FIU-Tree and all 'k' frequent itemsets are generated by checking the leaves of FIU-Tree where 'k' is from M down to 2. After decomposing process 'k' itemsets are generated that are used for the construction of K D B FIU Tree. Initially the root is labelled as null. Then each 'k' itemsets are inserted into the

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

tree. If first frequent item exists as one of the children of the root, then it denote the child as a temporary 1st root, if it is not exist then add a new node for this item as a child of the root node and denote it as temporary 1st root. Then the sth frequent item of the k itemset, where ‘s’ is from 2 to k - 1, check if the sth frequent item exists as one of the children of the temporary (s-1)th root, then denote the child as a temporary sth root. If it does not exist, then add a new node for this item as a child of the temporary (s-1)th root and denote it as a temporary sth root. This process is repeated until K-FIU Tree is constructed. By checking the leaf node all k frequent items will be generated. Each phase of Density based Frequent Itemset Ultrametric Tree is explained with an example. Consider the 5 transactional databases D as shown in the Table 1.

Table1: Database D itemsets

T.ID	ITEMS BOUGHT
100	a,c,d,f,g,i,m,p
200	a,b,c,f,l,m,o
300	b,f,h,j,o
400	b,c,k,s,p
500	a,c,e,f,l,m,n,p

During the phase 1 at the first round of scanning the database frequent one itemsets will be generated with the minimum support count value 2. Table 2 shows the frequent one itemset of the database D. During the phase 1 at the second round of scanning the database all ‘k’ itemsets will be generated by pruning the each infrequent item from each transactional datasets.

Table 2: Frequent 1 itemsets

Item Name	No. of Items
A	3
B	4
C	5
F	4
M	3
P	4

Table 3 shows all ‘k’ itemsets. During the phase 2 K-FIU Tree is constructed by using the ‘k’ itemsets generated after decomposing each ‘h’ itemsets into ‘k’ itemsets. i.e ‘acfmp’ and ‘abcfm’ can be decomposed into 4-itemsets like ‘abcf:1’, ‘abcm:1’, ‘abfm:1’, ‘acfm:1’,‘bcfm:1’, ‘acfm:2’, ‘acfp:2’, ‘acmp:2’, ‘afmp:2’, and ‘cfmp:2’. Similarly it can be decomposed into K to 2 items.

Table 3: All K Itemsets

Itemsets	Items Bought
5-itemsets	a,c,f,m,p
	a,b,c,f,m
4-itemsets	Ø
3-itemsets	b,c,p
2-itemsets	b,f

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

5-DBFIUT- when k =5 insert all 5 itemsets into the tree. Initially create the root node as null and then each five itemsets (‘abcfm:1’) 5 nodes should be created. ‘a’ is a temporary root bode of ‘b’, ‘b’ is a temporary root node of ‘c’ like this items should be inserted. At the leaf node its path ‘abcfm’ with the count one is founded. Then insert the next 5 itemset ‘acfmp:2’, the first item a shares the child node of the root with the path ‘abcfm’ and then new branch is formed starting from a. At the leaf node, its path ‘acfmp’ with the count 2 is founded. Finally by checking the minimum support count, frequent 5 itemsets are generated L5= {Ø}. Figure 1 represents the 5-DBFIUT.

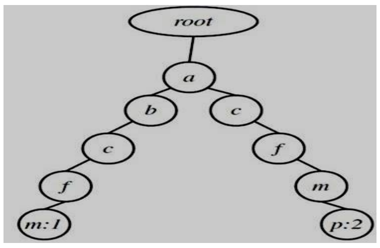

Figure 1: 5-DBFIUT

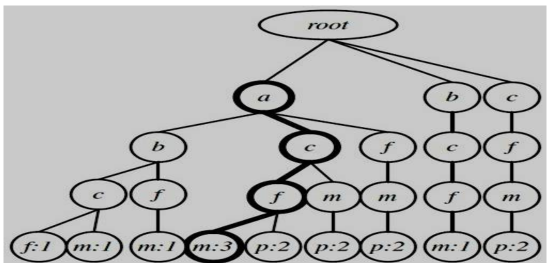

Figure 2: 4-DBFIUT

4-DBFIUT when k=4 insert all 4 itemsets into the tree. All 4 itemsets are ‘abcf:1’, ‘abcm:1’, ‘acfm:2’, ‘acfp:2’, ‘acmp:2’, ‘abfm:1’, ‘acfm:1’, ’bcfm:1’, ’afmp:2’, and ‘cfmp:2’. These itemsets are used to construct the 4-FIUT. Then by checking the leaf node frequent four itemsets are generated. L4= {acfm}. Figure 2 represents THE 4-DBFIUT when k=3 insert all 3 itemsets into the tree. All 3 itemsets are ‘bcp:1’, ‘abc:1’, ‘abf:1’, ‘abm:1’, ‘acf:1’, ‘acm:1’, ‘afm:1’, ‘bcf:1’, ‘bcm:1’, ‘afp:2’, ‘amp:2’, ‘cfm:2’, ‘cfp:2’, ‘cmp:2’, ‘bfm:1’, ‘cfm:1’, ‘acf:2’, ‘acm:2’, ‘acp:2’, ‘afm:2’ and ‘fmp:2’. These itemsets are used to construct the 3-DBFIUT. Then by checking the leaf node frequent three itemsets are generated. L3= {cfm, afm, acm and acf}. Figure 3 represents the 3-DBFIUT.

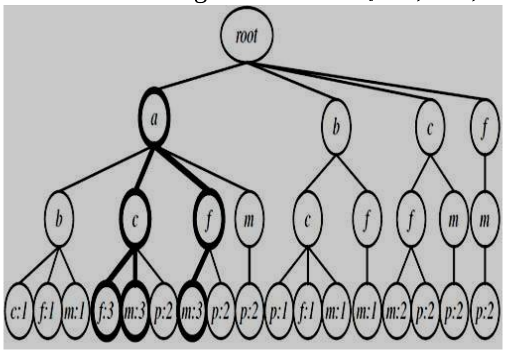

Figure 3: 3-DBFIUT

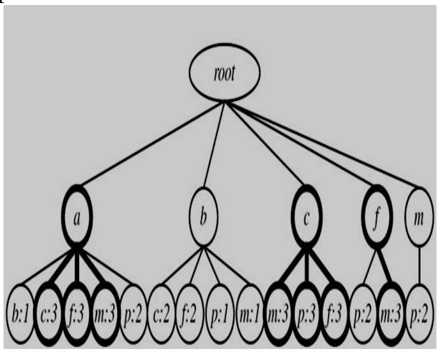

Figure 4: 2-DBFIUT

2-FIUT when k=2 insert all 2 itemsets into the tree. All 2 itemsets are ‘am:1’, ‘bc:1’, ‘bf:1’, ‘bm:1’, ‘cf:1’, ‘bf:1’, ‘bc:1’, ‘bp:1’, ‘cp:1’, ‘ab:1’, ‘ac:1’, ‘af:1’, ‘cm:1’, ‘fm:1’, ‘ac:2’, ‘af:2’, ‘am:2’, ‘ap:2’, ‘fm:2’, ‘fp:2’, ‘cf:2’, ‘cm:2’, ‘cp:2’, and ‘mp:2’ . These itemsets are used for the construction of 2-DBFIUT. Then by checking the leaf node frequent two itemsets are generated. L2= {cp, am, cf, cm, ac, af, fm}. Figure 3.4 represents the 2-DBFIUT.

IV. EXPERIMENTAL DESIGN

Modules Description

1 Frequent One Itemsets Generation

The first MapReduce job is responsible for mining all frequent one- itemsets. A transaction database is partitioned into multiple input split files stored by the HDFS across multiple data nodes of a Hadoop cluster. Number of mapper will be executed based on number of input split. Each mapper sequentially reads each transaction from its local input split, where each transaction is stored in the format of key value pair<LongWritable offset, Text record> by the record reader. Then, mappers compute the frequencies of items and generate local one-itemsets. Next, these one-itemsets with the same key emitted by different mappers are sorted and merged in a specific reducer, which further

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

produces global one itemsets. Finally, infrequent items are pruned by applying the minsupport; and consequently, global frequent one-itemsets are generated and written in the form of pair<Text item, LongWritable count> as the output from the first MapReduce job. Importantly, frequent one-itemsets along with their counts are stored in a local file system, which becomes the input of the second MapReduce job.

2 K Itemsets Generation

Given frequent one-itemsets generated by the first MapReduce job, the second MapReduce job applies a second round of scanning on the database to prune infrequent items from each transaction record. The second job marks an itemset as a k-itemset if it contains k frequent items ($2 \leq k \leq M$, where M is the maximal value of k in the pruned transactions). Each mapper of the second job takes transactions as input. Then, the mapper emits a set of pair <ArrayWritable itemsets, Longwritable ONE>, in which itemsets is composed of the number of the items produced by pruning and the set of items. These pairs obtained by the second MapReduce job's mappers are combined and shuffled for the second job's reducers. After performing the combination operation, each reducer emits key/value pairs, where the key is the number of each itemset and the value is each itemset and its count. More formally, the output of the second MapReduce job is pair <IntWritable itemnumber, MapWritable<ArrayWritable k-item, LongWritable SUM>>. Figure1.2 outlines the pseudocode of the second job's Map and Reduce functions. It is important to ensure that frequent items in each transaction should retain their lexicographical order in order to facilitate the next phase.

3 Frequent K Itemsets Generation

The third MapReduce job a computationally expensive phase is dedicated to: 1) decomposing itemsets; 2) constructing k-FIU trees; and 3) mining frequent itemsets. The main goal of each mapper is twofold: 1) to decompose each k-itemset obtained by the second MapReduce job into a list of small-sized sets, where the number of each set is anywhere between 2 to $k - 1$ and 2) to construct an DBFIU-tree by merging local decomposition results with the same length. The third MapReduce job is highly scalable, because the decomposition procedure of each mapper is independent of the other mappers. In other words, the multiple mappers can perform the decomposition process in parallel.

Such an FIU-tree construction improves data storage efficiency and I/O performance; the improvement is made possible things to merging the same itemsets in advance using small DB-FIUT trees. Non leaf nodes include item-name and node-link; leaf nodes include item-name and its support. By parsing the key-value pair (k2, v2), the reducer is responsible for constructing k2-DBFIU-tree and mining all frequent itemsets only by checking the count value of each leaf in the k2-DBFIU-tree without repeatedly traversing the tree.

PARAMETERS FOR EVALUATION

Minimum Support Count

Minimum support count plays the important role in mining frequent itemsets. When user increase the minimum support threshold the running time of the proposed algorithm reduces. A small minimum support slows down the performance of the evaluated algorithms. This is because an increasing number of items satisfy the small minimum support when the minsupport is decreased; it takes an increased amount of time to process the large number of items. Figure 5 shows the execution time of four different minimum support counts.

Scalability

In this experiment, evaluate the scalability of the proposed algorithm when the size of input dataset grows dramatically. The parallel mining process is slowed down by the excessive data amount that has to be scanned twice. The increased dataset leads to a long scanning time. An output of the second MapReduce job are distributed and stored in intermediate files based on the length of itemset; these files are accessed by the third MapReduce job as an input. Further, the decomposed results are written into these external files. In summary, the scalability of the proposed algorithm is higher when it comes to parallel mining of an enormous amount of data. Figure 6 shows the running time of different sized datasets.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

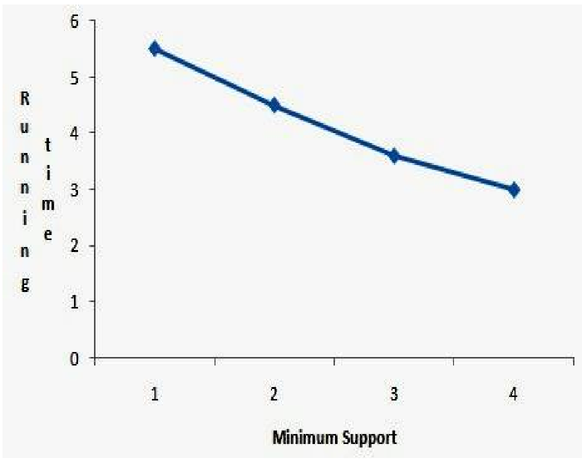

Figure 5 Minimum Support

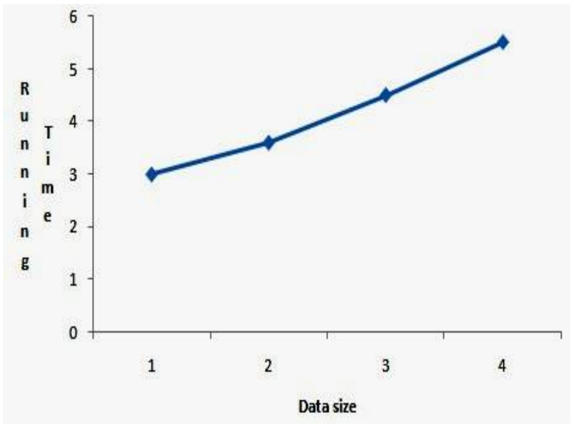

Figure 6 Scalability

EXPERIMENTAL SETUP

In the large set of samples market basket dataset only 300 samples are taken, in that 87 market basket dataset are classified as frequent itemsets and 22300 market basket dataset are classified as News market basket dataset. According to the algorithm, 8700 Frequent itemsets market basket dataset are classified as 3900 Frequent itemsets Negative market basket dataset and 4800 Frequent itemsets Non-Negative market basket dataset which are shown in the below table 4, table 5 shows the experimental result of proposed system and figure 7 shows the comparison chart between existing and proposed system.

Table 4: Frequent itemsets classification

Total Number of Frequent itemsets	50	-
Correctly Classified Frequent itemsets	50	100% Accuracy
Incorrectly Classified Frequent itemsets	-NUL-	0% Accuracy
Mean absolute error	0.0553	-
Root mean squared error	0.0593	-

Table 5: Experimental result of existing and proposed system

Algorithm	False positive	False Negative	Entity Verified	Performance Accuracy (%)
FIUT (Existing)	56	38	24	93.6
DB-FIUT (Proposed)	54	40	35	95

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

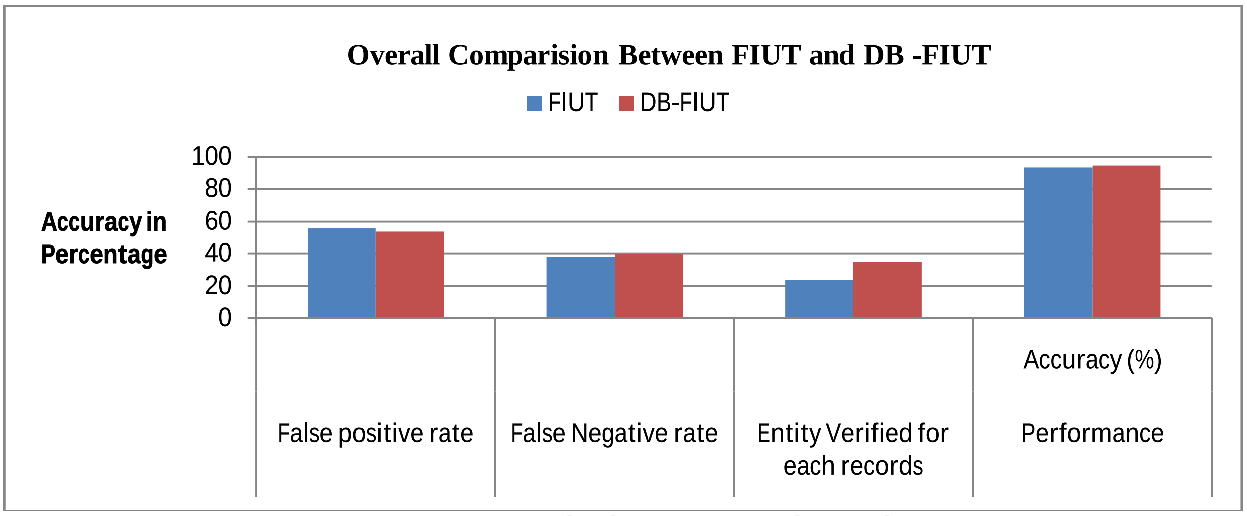

Figure 7: Comparison chart between existing and proposed system

V. Conclusion and Future Work

The parallel mining of frequent itemsets (FI) using Frequent Itemset Ultrametric tree (FIUT) with Density Based Spatial Clustering of Applications with Noise (DBSCAN) on MapReduce framework is used in the proposed system to solve the scalability and efficiency problem in an existing frequent itemset. In Proposed system, it incorporates the Density Based Frequent Itemset Ultrametric Tree by adding additional hash tables rather than using conventional FP trees, thereby achieving compressed storage and avoiding the necessity to build conditional pattern bases. The proposed algorithm integrates three MapReduce jobs to accomplish parallel mining of frequent itemsets. The first MapReduce job is responsible for mining all frequent one- itemsets. The second MapReduce job applies a second round of scanning on the database to prune infrequent items from each transaction record. At the end of the third MapReduce job all frequent K-itemsets are generated. Synthetic datasets is used in the experiments to evaluate the performance of the proposed DB-FIUT algorithm on MapReduce framework. As a future enhancement, in the MapReduce framework, distributed cache technique can be used in distributed read-only data file. A new technique called LRU (Least Recently Used) cache technique which reduces the recompilation time and it also keeps tracking of what item was used when. Distributed cache technique consisted with least memory space so LRU cache can be implemented in the future work to allocate memory space simultaneously.

References

1. A.B. Patel, M. Birla and U. Nair, "Addressing big data problem using Hadoop and Map Reduce", “Nirma University International Conference on Engineering (NUiCONE)”, Ahmedabad, 2012.
2. Agrawal, R. and Srikant, R. “Mining sequential patterns” IEEE,1995.
3. Andrea Pietracaprina , Geppino Pucci , Matteo Riondato , Francesco Silvestri , Eli Upfal, “Space-round tradeoffs for MapReduce computations”, “Proceedings of the 26th ACM international conference on Supercomputing”, June 25-29, 2012.
4. B.G.Obula Reddy1, Dr. Maligela Ussenaiah2, “ Literature Survey On Clustering Techniques”, IOSR Journal of Computer Engineering, Vol. 3,pp. 01-12, July-August 2012.
5. Chang E.Y., Li H., Wang Y., Zhang D. and Zhang M. , “PFP: Parallel FP-growth for query recommendation”, “in Proc. ACM Conf. Recommend.Syst.”, Lausanne, Switzerland, 2008.
6. Chang W.L., Chen P.L. and Lin K.W., “A novel frequent pattern mining algorithm for very large databases in cloud computing environments’, “in Proc. IEEE Int. Conf. Granular Comput. (GrC)”, Kaohsiung, Taiwan, 2011.
7. Chunyan H., Hong S., Huaxuan Z. and Shiping s. (2013), ‘The study of improved FP-growth algorithm in MapReduce’ in Proc. 1st Int. Workshop Cloud Comput. Inf. Security, Shanghai, China, 2013.
8. D. Huang, Y. Song, R. Routray and F. Qin, "Smart Cache: An Optimized MapReduce Implementation of Frequent Itemset Mining", “Cloud Engineering (IC2E), 2015 IEEE International Conference on, Tempe”, AZ, 2015.
9. Dean J. and Ghemawat S., ‘MapReduce: Simplified data processing on large clusters’, Commun. ACM, 2008.
10. Dean J. and Ghemawat S., ‘MapReduce: A flexible data processing Tool’, Commun. ACM, 2010.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

11. Glory H. Shah, C. K. Bhensdadia, Amit P. Ganatra ,"An Empirical Evaluation of Density-Based Clustering Techniques",International Journal of Soft Computing and Engineering, Vol. 2, pp. 216-223, March 2012.
12. Hsueh S.C., Lin M.Y. and Lee P.Y. , "Apriori-based frequent itemset mining algorithms on MapReduce", "in Proc. 6th Int. Conf. Ubiquit. Inf. Manage. Commun. (ICUIMC)", Danang, Vietnam, 2012.
13. J. Neerbek, "Message-driven FP-growth," in Proc. WICSA/ECSA Compan. Vol., Helsinki, Finland, 2012.
14. K. Mumtaz et al, "A Novel Density based improved kmeans Clustering Algorithm – Dbkmeans", International Journal on Computer Science and Engineering, Vol. 02, pp. 213-218, 2010.
15. Liang F., Kirsh I., Shi Z. and Yang L., "DH-TRIE frequent pattern mining on Hadoop using JPA", "in Proc. IEEE Int. Conf. Granular Comput. (GrC), Kaohsiung", Taiwan,2011.
16. M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, "PARMA: A parallel randomized algorithm for approximate association rules mining in MapReduce," in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage., Maui, HI, USA, 2012.
17. M.Parimala, Daphne Lopez, N.C. Senthilkumar "A Survey on Density Based Clustering Algorithms for Mining Large Spatial Databases", International Journal of Advanced Science and Technology Vol. 31, pp. 59-66, June-2011.
18. Mukherjee, A.; Datta, J.; Jorapur, R.; Singhvi, R.; Haloi, S.; Akram, W., "Shared disk big data analytics with Apache Hadoop".December,2012.
19. S. Hong, Z. Huaxuan, C. Shiping, and H. Chunyan, "The studyof improved FP-growth algorithm in MapReduce," in Proc. 1st Int. Workshop Cloud Comput. Inf. Security, Shanghai, China, 2013.
20. S. Sagiroglu and D. Sinanc, "Big data: A review", "Collaboration Technologies and Systems (CTS)", International Conference on, San Diego, CA, 2013.
21. S. Rathi and C. A. Dhote, "Using parallel approach in pre-processing to improve frequent pattern growth algorithm", "Information Systems and Computer Networks (ISCON)", International Conference on, Mathura, 2014.
22. Sanjay Chakrobarty, Prof. N.K.Nagwani," Analysis and Study of Incremental DBSCAN Clustering Algorithm", International Journal of Enterprise Computing and Business,Vol. 1, Issue 2 ,July 2011.
23. Y.-J. Tsay, T.-J. Hsu, and J.-R. Yu, "FIUT: A new method for mining frequent itemsets," Inf. Sci., vol. 179, no. 11, pp. 1724–1737, 2009.
24. Yu K. and Zhou Y., "Parallel TID-based frequent pattern mining algorithm on a PC cluster and grid computing system", "Expert Syst. Appl.", volume 37,2010.