IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 6, June 2017

Parallel Frequent Itemset Mining for Big Datasets using Hadoop-MapReduce Paradigm

Seema Tribhuvan[1], Bharti. P. Vasgi[2]

Student, Information Department, Sinhgad College of Engineering, Pune, India [1]

Professor, Information Department, Sinhgad College of Engineering, Pune, India [2]

Abstract: Frequent Itemset Mining is one of the classical data mining problems in most of the data mining applications. There are various parallel mining algorithms available for frequent itemsets mining, such as Apriori, Fp-Growth algorithms. However, these parallel mining algorithms lack features like automated parallelization, fine load balancing, and distribution of data on large clusters. To address these issues the most effective recent method is using the enhanced version of Apriori algorithm (EA). In this technique three MapReduce tasks are implemented to complete the mining of big datasets by using the parallelism among computing nodes of clusters to improve the performance of frequent pattern mining on hadoop clusters. After third map reduce job, frequent patterns will be produced as a final outcome.

Keywords: frequent item sets, Enhanced Apriori, Hadoop, MapReduce.

I. INTRODUCTION

Data mining for big datasets is the practice of examining large pre-existing databases in order to create new information. There are several data mining methods available like clustering, classification and association rule mining. Among these one of the important method is the association rule mining which falls in two steps i) generating the frequent itemsets ii) generating association rules from all frequent itemsets. Frequent Itemset Mining (FIM) is a method for discovering interesting relations between variables in large databases. To find FIM, two well-known algorithms are Apriori and FP Growth (Frequent Pattern). Apriori uses the candidate generation approach and has to repeatedly scan the database. To reduce the time required for scanning of database and without generating candidate itemsets the next approach is FP Growth. Sequential FIM algorithm has the problem of performance deterioration when it work on vast amount of data on a single machine. To address this issue parallel FIM algorithms were proposed. Use of MapReduce programming model to solve the above problem which can handle large datasets through number of clusters. This distributed approach is combined with FIM to overcome the drawbacks of sequential FIM and to increase performance [3]. So the approach is called as Fidoop as it uses Hadoop- MapReduce model. In this approach FIUT (Frequent Items Ultrametric Tree) algorithm used to provide compressed storage, to avoid conditional pattern bases, to reduce I/O overhead [3].

Using MapReduce model it is easy to handle the large datasets across number of clusters. This distributed method is combined with FIM to overcome the drawbacks of sequential FIM and hence performance get increased. Instead of FIUT with mapreduce, an Enhanced Apriori algorithm (EA) is used to reduce number of scans on datasets. It mainly focuses on data partitioning method and load balancing of datasets. It takes less time compared to old apriori algorithm. Mappers and reducers works in parallel to boost speed and fine load balance among clusters.
The remainder of the paper is arranged as follow. Section II explains the related studies of frequent itemset mining methodology. Section III describes the proposed system and its architecture. Section IV Section V illustrate various algorithms and methods to be used in proposed system and results. Section VI concludes the work and briefs future research.

II. RELATED WORK

Frequent item set mining used in wide range of application areas such as decision support, web usage mining, bioinformatics, etc. There are variety of algorithms proposed by different researchers for Frequent Itemset Mining. Each of it has its own advantages and disadvantages. Following is the review of some of the research papers from various conferences and publications.

The proposed research work [1] presents the problem related with extraction of frequent items from huge datasets. It presents rules that have minimum transactional support and minimum confidence. The proposed algorithm that

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 6, June 2017

calculates the itemsets for one pass and adjust among the number of passes over data and itemsets. This calculation uses cropping technique to avoid certain itemsets. Thus gives right related itemsets from large databases. Benefit of this algorithm is, buffer management technique which are not fit in the memory in one pass and so it shifts to next pass. Benefit of this approach is, it enhances parallelization and so improves execution. But Lacks in automatic parallelization.

Parallel FP Growth algorithm using balanced partitioning (BPFP) [2] is another important proposed system for pattern mining. It works in two stages, in first stage of BPFP, based on conditional pattern load is computed. In next stage, this load is get divided into many groups. For this purpose, MapReduce paradigm is used on FP algorithm.

The remarkable system that gives the parallel mining of frequent itemsets using Hadoop-MapReduce programming paradigm which not only gives parallelization but also handles load balancing, fault tolerance, and data distribution on huge cluster [3]. The approach is called as Fidoop. To get compressed storage and avoid conditional pattern bases, fidoop integrates FIUT structure than FP trees. There are three Mapreduce jobs who plays important steps in entire frequent itemset mining approach.

The first MapReduce job creates one-itemsets. Transactional database divided into many input files stored by HDFS on data nodes of Hadoop clusters. Each mapper serially reads the input and compute the frequencies of items and create one-itemsets. Reducer merges and sorts the mapper's one-itemsets, in the form of output pair <Text item, LongWritable count>, and stored in local file named as F-list, which will become input for second Mapreduce job.
The second MapReduce job scans the database and with reference list of one-itemset from first MapReduce job, cut the infrequent items from each transaction record. And provide k-itemsets. The output of reducer is key/value pair with key as the number of each itemset and value is each itemset and its count.

The third MapReduce job is difficult one as it does i) Decomposing itemsets, ii) building k-FIU tress and Mining of frequent itemsets. Third job is more scalable as decomposing of each mapper in not dependent on other mappers. The output of the Map function is in the form of key value pairs where keys are number of items in itemsets and value are FIU tree with leaf and nonleaf nodes. The reducer builds the k2-FIU tree and mine the frequent itemsets just by examining count of each leaf in k2FIU tree without repetitively navigating tree.

Advantages of this technique are i) Use of many mappers to decompose h-itemsets in parallel increases data storage efficiency and I/O performance. ii) Provides scalability and better load balancing. But the approach is works well with homogeneous clusters not with heterogeneous clusters.

III. PROPOSED SYSTEM ARCHITECTURE

Using mapreduce programming model it is possible to implement frequent itemset mining algorithm which gives parallelism. So to reduce drawbacks of traditional system and get automated parallelization, load balancing and efficient data distribution. Proposed system uses Enhanced Apriori algorithm (EA) to solve the problems of FIUT algorithm. Through this method is it is possible to reduce time that is required for scanning the transactions.

The proposed system uses new data partitioning technique to balance the load on clusters using similarity metric. The following diagram shows the proposed system model. The system will be distributed in two major parts. First part is responsible for data preprocessing, data partitioning and clustering i.e. loading data to Hadoop distributed file system (HDFS) framework. Second, the mapreduce jobs will handle the entire mining process of the frequent patterns.

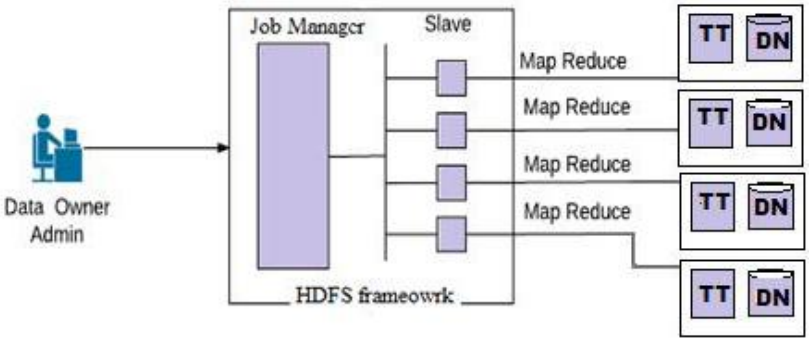

Figure 1 HDFS and MapReduce for Frequent Itemsets

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 6, June 2017

Job manager will handle the data partitioning and clustering task and also schedules the tasks for task trackers. Slave nodes responsible for mapredcue tasks. Task Tracker (TT) tracks the job and interact with job manager. Data node (DN) where all portioned data is managed by HDFS. Mapreduce jobs will mine the entire dataset in and returns the result to respected slave nodes. Finally, the results get merged and provide the final frequent patterns.

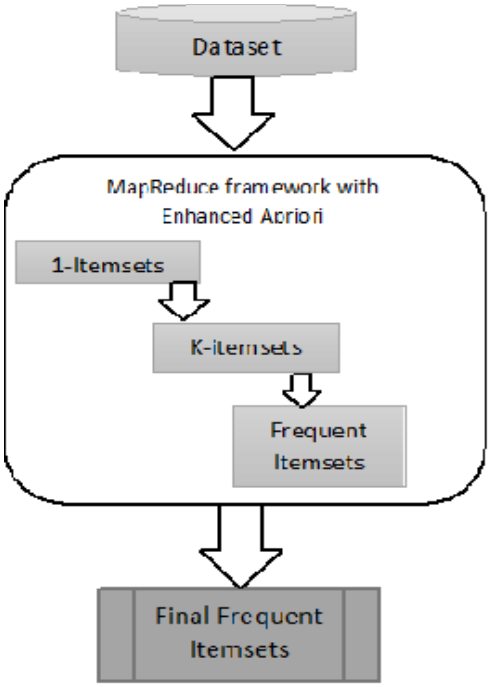

Figure 2 Frequent Itemset mining using MapReduce and Enhanced Apriori

Figure 2 shows the working flow of the frequent itemset mining with Enhanced Apriori algorithm using MapReduce programming model. It uses the hashing and pruning method to generate frequent itemsets. This method attempts to generate large itemsets efficiently and reduces the transaction database size.

For Example, below in table 1 shows sample transactions. After finding and pruning all 1-itemsets with minimum support count >2, we get next k-itemsets as shown in table 2

TID	Items
100	Bread, Cheese, Eggs, Juice
200	Bread, Cheese, Juice
300	Bread, Milk, Yogurt
400	Bread, Juice, Milk
500	Cheese, Juice, Milk

100	(B, C) (B, E) (B, J) (C, E) (C, J) (E, J)
200	(B, C) (B, J) (C, J)
300	(B, M) (B, Y) (M, Y)
400	(B, J) (B, M) (J, M)
500	(C, J) (C, M) (J, M)

When generating L1, the algorithm also generates all of the 2-itemsets for each transaction, hashes them to a hash table and keeps a count.

The two digits are then coded as modulo 8 number (dividing by 8 and using the remainder). This is the bucket address. That is hash function given by,

$$H(x, y)=[(\text{order of } x)*10=(\text{order of } y)] \% 8$$

A count of the number of pairs hashed is kept. Those addresses that have a count above the support value have the bit vector set to 1 otherwise 0. All pairs in rows that have zero bit are removed.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 6, June 2017

Bit vector	Bucket number	Count	Pairs	C_2
1	0	3	(C, J) (B, Y) (M, Y)	(C, J)
0	1	1	(C, M)	
0	2	1	(E, J)	
0	3	0		
0	4	2	(B, C)	
1	5	3	(B, E) (J, M)	(J, M)
1	6	3	(B, J)	(B, J)
1	7	3	(C, E) (B, M)	(B, M)

From above table we will get C2 as below.

(C,J)	3
(J,M)	3
(B,J)	3
(B,M)	3

Hashing is used for reducing the size of the candidate k-itemsets. It means itemsets that are generated from frequent itemsets from iteration k-1, ck, for k>1. It prunes both number of trasactions and items in each transaction after every iteration, thus efficiency can be achieved. Hash tables stores the previous results to avoid unnecessary repeatation of frequent itemsets. It sets bit vector 1 if particular itemset is present if not sets to 0.

Algorithm:
1. Scan the Dataset. Create all possible 2-itemsets.
2. Let the size of Hash table = 8.
3. For every bucket allocate a candidate pairs using the ASCII values of the itemsets.
4. Each one bucket in the hash table has a count, which is increased by 1 every item an item set is hashed to that bucket.
5. If the bucket count is equal or above the minimum support count, the bit vector is set to 1. Otherwise it is set to 0.
6. The candidate pairs that hash to locations where the bit vector bit is not set are removed.
7. Adjust the transaction dataset to include only these candidate pairs.

For example, after scanning each transaction in the dataset to generate the frequent 1-itemsets, L1, from the candidate 1-itemsets in C1, it is possible to generate all of the 2-itemsets for every transaction, hash(i.e) map these into the different buckets of a hash table structure, and increase the corresponding bucket counts . A 2-itemset whose corresponding bucket count in the hash table is under the support threshold cannot be frequent and thus it should be removed from the candidate set. Thus a hash based apriori may substantially reduce the number of the candidate k-itemsets studied.

In this algorithm, for entire transaction all 1-itemsets are counted. At the same time all the possible 2-itemsets in the current transaction are hashed to a hash table. It uses a hash table to reduce the number if candidate itemsets. When the support count is provided, the algorithm determines the frequent itemsets. It generates the candidate itemsets as like the Apriori algorithm.

IV. CONCLUSION

To deal with the issues of existing methods like parallel mining and efficiency of frequent itemset mining algorithms, new approach is considered by providing distributed and parallel framework of mapreduce. For achiving efficient parallelism Enhanced Apriori algorithm is used with mapreduce model. To overcome the disadvantages of FIUT the proposed system uses the enhanced Apriori algorithm.

There are various frequent itemsets mining algorithms available each with its own advantages and disadvantages. Apriori and FP Growth algorithms are used for mining patterns. Also each has its drawbacks. In Apriori way, database needed to scan frequently and generate many candidate keys that increases I/O and synchronization problems. These problems are overcome by FP Growth algorithm, which also has drawback of in-memory constriction of trees.

The drawbacks of Apriori and FP-Growth are overcome by the approach of FIUT algorithm. FIUT needs to scan database only twice and it reduces search space also no need to traverse entire tree just to check leaves to find frequent items.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 6, June 2017

The proposed system uses Enhanced apriori algorithm which uses hash- based algorithm. It prunes the itemsets as well as dataset by pruning infrequent itemsets. Thus efficiency can be achieved. The applications of frequent itemset mining can be range from sentiment analysis, market basket analysis, web mining, knowledge driven business decisions, to find accident patterns, click stream analysis, web link analysis, genome analysis, drug design (Molecular Fragment Mining)

REFERENCES

[1] R. Agrawal, T. Imieli´nski, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Rec., vol. 22,no. 2, pp. 207–216, 1993.
[2] Le.Zhou, Zhiyong Zhong, Jin chang , " Balanced parallel FP-growth with MapReduce", "Proc. IEEE Youth conf. Inf. Comput. Telecomm.(YC-ICT), Beijing, China, 2010, PP.243-246.
[3] YalingXun, Jifu Zhang and Xian Qin, "Fidoop: Parallel mining of frequent Itemsets using MapReduce ", IEEE Trans.on sys.man and cybernetics, Vol. 46,No.3, March 2016
[4] Sandy Moens,Emin Aksehirli and Bart Goethals, " Frequent Itemset Mining for BigData", intl. conf on Bigdata, IEEE 2013
[5] 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.
[6] Sheela gole and Bharat Tidke, "Frequent Itemset Mining for BigData in social media using ClustBigFIM algorithm", Intl Conf.on Pervasive Computing, IEEE 2015.
[7] M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh, "Apriori-based frequent itemset mining algorithms on MapReduce," in Proc. 6th Int. Conf. Ubiquit. Inf. Manage. Commun. (ICUIMC), Danang, Vietnam, 2012.
[8] Kiran Chavan, Priyanka Kulkarni, Pooja Ghodekar, S. N. Patil," Frequent itemset mining for Big data ", IEEE,Green Computing and Internet of Things (ICGCIoT), 2015.
[9] 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.
[10] Wei Lu,Yanyan Shen,Su Chen,Beng Chin Ooi,"Efficient Processing of k Nearest Neighbor Joins using MapReduce"2012.
[11] Jaiwei han, Jian pei, Yiwen yin, " Mining frequent patterns without candidate generation : a frequent Pattern Tree", Data Mining and knowledge discovery, 2001
[12] Zahra Farzanyar and Nick Cercone, "Efficient mining of frequent itemsets in social network data based on MapReduce framework", Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.
[13] M. J. Zaki, "Parallel and distributed association mining: A survey" IEEE Concurrency, pages 14–25, 1999.
[14] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM vol 51, jan 2008.