International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

Data Partitioning in Frequent Itemset on Bigdata Using Hadoop

Sindhuja[1], M. Sridevi[2]
[1]Department of CNIS, G Narayanamma Institute of Technology and Science, Hyderabad, Telangana, India
[2]Assistant Professor, Department of CNIS,G Narayanamma Institute of Technology and Science, Hyderabad, Telangana, India

ABSTRACT

Generally FIM is one of primary concerns in data mining. Whereas the problems of FIM have been studied, that standard and better solutions scale. This is generally the case when i) the sum of data tend to be extremely large and/or ii) A MinSup threshold is very low. In this paper, I propose a highly measurable and parallel frequent item set mining (PFIM) algorithm that is Parallel Absolute Top Down. PATD algorithm renders the mining process of very large amount of databases (Terabytes of data) easy and compact. Its mining process is completed for just parallel jobs, which dramatically reduce the mining runtime, communication cost and energy power utilization overhead, in a disseminated computational platform. Based on an intellectual and efficient data partitioning approach describe IBDP, PATD algorithm mines every data partition separately, relying on entire minimum support (A MinSup) as of a Relative one. PATD contain extensively evaluated using real-world data sets. My experimental results advise that PATD algorithm is considerably more capable as well as scalable than alternative approaches.
Keywords: Big Data, Data Mining , Frequent Itemset ,Machine Learning, MapReduce

I. INTRODUCTION

As Association Rule Mining follows a particular method is intended to find out frequent patterns, correlation, association from datasets such as relation, transaction databases. Example: In real world, when consumers purchase a sandwich it is probable to get ketchup along. This is exactly how the association rules mining works, Such that chronological pattern mining is a process of connecting a topic of data mining with identifying the similar patterns. When these are put in use a problem occurs in FIM is a system or a process which get place in a particular way for example: an artist prefers to paint the background first and then filling in the details, therefore this pattern is followed frequently by him. FIM creates fragments of mining instance of a particular portion; this is done due to lofty input or output intensity. Because of which it is essential to rapidity up the process, which is solid to achieve .By introducing FIM which uses MapReduce to solve the issue i.e., when a dataset in data mining application is huge the sequential FIM algorithm running on a single machine results is catastrophic.

Machine learning algorithms are classifies as being supervised or unsupervised. Supervised algorithms need humans to gives together input plus desired output; unsupervised algorithms no require to be taught with desired result data. Instead, they can use iterative approach called deep learning to review data with arrives at conclusions. Unsupervised knowledge algorithms are worn for more complex processing jobs for supervised learn systems. This methods involved in machine learning as same to that of data mining it need searching during data to seem for pattern and adjust program actions accordingly. This happens because use mechanism learns to personalize online ad delivery in approximately real time. Past personalized marketing, extra regular machine learning comprise fraud detection, spam filtering, network security threat detection, and building reports feeds.

Frequent Itemset Mining

Datasets in current data mining application become excessively large. So, it might cause load balancing, some redundant transactions transmitted among computing nodes this causes to work load and network traffic so that here improving presentation of FIM. It

has convenient way of significantly shortening data mining time of the application. Generally I use sequential FIM algorithm but, those algorithms able to run in an single machine that suffer a worse performance due to limited computational and storage resources, to defeat this difficulty I am focusing on parallel FIM algorithms operating on clusters.

Mapreduce Programming Model

MapReduce is a assure parallel also scalable programming model for data-intensive applications plus scientific analysis. A Mapreduce programming express large distributed computations as a sequence of the parallel operations on data sets of key pairs. A mapreduce calculation has two stages namely, the Map along Reduce phases. Hadoop is an open source execution of the mapreduce programming model. It is worn for procedure huge datasets by paralleling them amongst the computing nodes of a cluster. By optimizing the parallel FIM, it results in load balancing. Apriority and FP-growth are the categories of FIM. The apriority generates list of candidates list, using the bottom up approach it scans for the frequent item sets are groups the frequently used candidates list. To lessen the time taken for scanning FP-growth algorithm was introduced which is scalable and efficient, it compresses the storage by constructing the prefix tree, which eliminate the generation of candidates and saves the time which is required for scanning.

The disadvantage of FP-growth is that is infeasible in constructing the in memory FP tree, this becomes even difficult when it come to multi dimensional database. To overcome these faults the frequent items ultra metric tree (FIU-tree) is used due to advantages like reducing the input or output overhead, offer a usual system of partitioning a dataset, compressed storage , recursively traverse and also enables mechanical parallelization, load balancing, data distribution, with fault tolerance on huge computing clusters which was lacking in previously used algorithms. To solve the above mentioned problems I incorporate a parallel mining FIM algorithm call FIDOOP using MapReduce.

Data Partitioning in Hadoop Clusters

A partitioning a division of logical database or else its constituent elements through different autonomous part. Database partition is usually ended for manageability, performance or availability reasons, load balancing. Hadoop mapreduce make a decision that the job begins set of partitions it may divide the data. In presented data partition technique FIM aim is to balancing computation load by equally distributed data among nodes. However, the correlations amid the records is often ignored which will lead to poor data locality, resources wastage, network overhead will be increased I build up FiDoop-DP is a parallel FIM technique in such a huge data set is partitioned across a hadoop clusters data nodes has to advance data locality.

FiDoop-DP using the MapReduce programming form is proposed. The objective of FiDoop-DP is to get better the presentation of parallel FIM on Hadoop clusters. It is the Voronoi diagram-based data partition system, such exploits correlations amongst transactions. It places very analogous transactions through data partition to advance locality with no creating an extreme number of unnecessary transactions. The proposed FiDoop- DP, which may flexibly configured to produce a extensive range of data sets to match the requirements of diverse test requirements.

The FiDoop-DP contains four steps specifically, the next mapreduce job is the spirit of the project where we perform Voronoi based partitioning to get out the entire frequent item sets.

In the initial mapreduce job, ecach mapper serially reads all transaction from the local input and split on a data node to produce local 1-itemsets,next all 1-itemsets sharing the same key. The output of these reduces include the all over frequent 1-itemsets beside with their counts. Thenext step sorts the entire frequent 1-items in an declining order of frequency, the sorted are saved in cache names F list, which becomes to the succeeding mapreduce work in FiDoop-DP.

The next MapReduce job apply a second-round examine the database to repartition database to form a complete dataset for item groups in the map phase. Apiece reducer conduct local FP-Growth based on the partitions to generate all frequent patterns.

The final MapReduce job aggregates the subsequent MapReduce jobs output to make every one of final frequent patterns for each item.

II. RELATED WORK

[1]Yaling Xun, Jifu Zhang, Xiao Qin, "FiDoop-DP: Data Partitioning in FIM on Hadoop Clusters", 2016.It explains, A data partitioning approach term FiDoop-DP by the Map Reduce programming model. The overarching aim of FiDoop-DP to get better the performance and similarity metric to make easy data-aware partitioning. As an upcoming research direction, I would relate this metric to examine advance load balancing strategies on a heterogeneous Hadoop cluster.

[2] I.Pramudiono& M.Kitsuregwa," Fp-tax: Tree structure based generalized association rule mining", 2004. This paper describe, examination of data portioning issues in parallel FIM. Major focus is on map-reduce. Future job is improvement of Fidoop which develop correlation amongst traction to partition large datasets in Hadoop.

[3] X.Lin," Mr-apriori: Association rules algorithm based on mapreduce",2014. Mainly spot on classical Algorithm linking and snip step using prefix Itemset based storage by hash table. It spot a few limits of Apriori algorithm.

[4]S. Hong, Z.Huaxuan, C. Shiping, plus H.Chunyan," The learn of better fp-growth algorithm in mapreduce",2013.This explain, make cloud platform for perform the parallel FP-growth algorithm based on linked list plus PLFPG. This algorithm disparity upper efficiency also scalability.

[5]M. Liroz-Gistau, R. Akbarinia, D. Agrawal, E. Pacitti, and P. Valduriez," Data partitioning for minimize transport data in mapreduce",2013.It state that, Map Reduce job is execute more distributed system poised of a master plus set of workers. Input is separating into numerous splits along assigned to map tasks. Upcoming job is evasion to present the repartitioning in parallel. III.

III. IMPLEMENTATION

Parallel Counting: The initial MapReduce job count the sustain values in the whole items residing in the database to discover all frequent items or frequent 1-itemsets in parallel. This step simply examines the database once.

Sorting frequent 1-itemsets to FList: The succeeding step sorts these frequent 1-itemsets in a lessening order of frequency; the sorted frequent 1-itemsets are cached in a catalog named FList. A Non-MapReduce procedure due to its cleanness and the centralized control Parallel.

FP-Growth: These steps of Pfp, where the map stage plus lessen stage execute the after two significant functions. Mapper - Grouping items plus generating group-dependent dealings. First, the Mappers separate every item in FList into Q groups. The record of group is referred as group list or GList, where each group is allocated a unique group ID (i.e., Gid). Then, the transactions are partitioned into various groups according to GLists. That is, all mappers of outputs more than one key-value pair, where a key is a group ID with its corresponding value is a produced group-dependent transaction. Reducer - FP-Growth on group-dependent partitions, Local FPGrowth is perform to create local FIM.

Apiece reducer conducts local FPGrowth by handing out different group-dependent partition one after other, and exposed patterns are output in the final. The following MapReduce work is a performance bottleneck of the complete data mining process. The map tasks apply a second-round scan to sort and prune each transaction according to FList, followed by grouping the sorted frequent 1-itemsets in FList to form group list GList. Next, each transaction is placed into a group-dependent data partition; multiple data partitions are constructed.

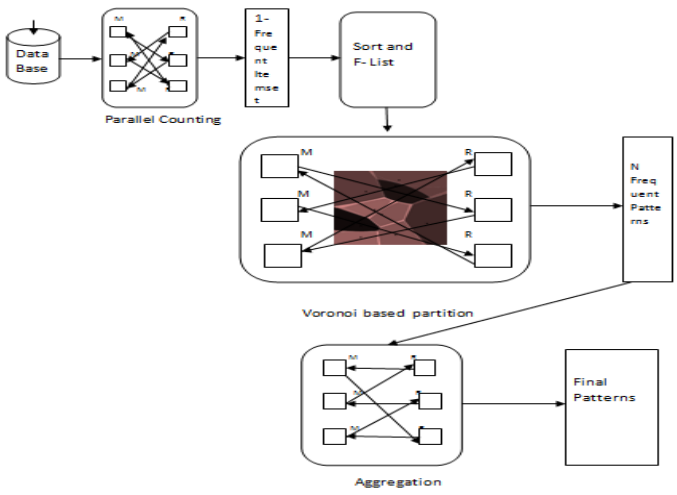

Figure 1. PATD System Process.

Each data partition corresponds with group identified by Gid. The above partitioning approach ensures data completeness with esteem to lone group of GList. A downside is that such data completeness comes at the cost of data redundancy, because a transaction might have duplicated copies in multiple data partitions.so, I used a voronoi based partitioning technique by which I find every FIM and decrease the redundant transactions

Aggregating: The final MapReduce job create final results by aggregate the output produce in Step second mapreduce programming.

IV. APPLICATION AWARE DATA PARTITIONING

Various efficient data partitioning strategies proposed to get better the appearance of parallel computing organizations. For case, Kirsten et al. develop two general partitioning plans for generating entity match jobs to shun memory bottlenecks and load inequity taking to the count individuality of input data, Aridhi et al. proposed a novel density-based data division method for estimated large-scale frequent sub graph mining to stability computational load amongst a set of machines. Kotoulas et al. built a data distribution mechanism based on clustering in elastic regions.

Data Partitioning Techniques

Generally our FIM plays a main role in data partitioning when I have discuss in the bove proposed system and as we discussed about Voronoi based partition and Distance metric. Now, I am available to discuss about pivot elements which plays a main role in voronoi based partitioning to dividing a space as a number of regions. For finding pivots I use K-means technique.

K-means clustering means, technique of vector quantization, it is fashionable for clustering study in data mining. K-means clustering plan is to divider n watching fit in to the nearest mean, serving to a prototype of the cluster. This results of partition the data liberty in voronoi cells. Next to the choice of pivots, I calculate the distances from respite of the objects of these pivots to determine a partition to which each object belongs. I develop the LSH-based strategy to implement grouping plus partitioning process. To which MinHash is employes as a foundation for LSH.

MinHash is a quick solution to estimate how similar two sets. It is gradually more becoming a popular solution for large-scale clustering problems. MinHash is divided as two steps in first step the huge data set is formed into a signature. The data is represented in an m X n matrix. N is symbolizing as transactions and M is representing as objects. Rows denote the objects and Column denotes the transactions.

$$hmin(T) = x, \text{ where } h(x) = Min^{n}\ i=1\ (h(xi))$$

Here set the item as one if the item is in the transaction or t is zero. On the basis of this matrix I make a signature matrix, for every transaction I find a hash value for which I have an minimum hash value, we make it an transaction. In minhashing I performed a set of comparisons. So, to overcome that problem I introduced a partitioning method known as LSH-based partitioning were it scans the every frequent itemsets for the first time only.

1) If || p-q| | = R, then P rH (h(p) = h(q)) = P1
2) If ||p-q||=cR, then P rH (h(p) = h(q)) = P2

V. DATA CHARACTERISTIC DIMENSIONALITY

FiDoop-Dp to efficiently lessen the quantity of needless transactions. In contract a dataset with high dimensionality have a extended average transaction span; so, data partitions formed by FiDoop-DP has no distinct discrepancy. Redundant transactions may liable to formed for partitions that lack distinct characteristics. The profit offered by FiDoop-DP for high dimensional datasets become insignificant.

Data Correlation

FiDoop-DP judiciously group's item with soaring relationship has to grouped and clustering like dealings jointly. In this mode, the quantity of unnecessary dealings reserved on several nodes is significantly reduced. Thus, FiDoop-DP is conducive to cutting reverse both data transmission traffic plus computing load.

VI. ALGORITHM USED: IBDP

1 //Job1 **Input:** Non-overlapping data partitions S = {S1; S2,…….Sn}of database D
Output: Centroids

2 //Map Task 1

3 Map(key: Split Name: K1, value = Transaction (Text Line): V1)
4 Tokenize V1, to separate all items
5 emit (key: Item, value: Split Name)
6 //Reduce Task 1
7 Reduce(key: Item, list(values))
8 while values: hasNext () do
9 emit (key :(Split Name) values: next (Item))
10 //Job2 Input: Database D Output: Overlapping Data Partitions
11 //Map Task 2
12 Study earlier job1 result one time in a key, values (DS), somewhere key: SplitName and values: Items
13 map (key: Null: K1, value = Transaction (Text Line): V1)
14 for SplitName in DS do if Items. Item ?V1?Ø; then
15 emit (key: SplitName, value: V1)
16 //Reduce Task 2
17 reduce (key: SplitName, list (values))
18 while values: hasNext () do
19 emit (key: (SplitName), values: next: (Transaction))

VII. EXPERIMENTAL RESULTS

Here Dataset link is used https://data.gov.in/catalog/stateut-wise-traffic-accidents-month-occurrence The method is simulated in JAVA and it requires WAMP server tool, Eclipse, and Hadoop for IDE and MySQL database. This part present the answer of proposed FiDoop using map reduces. The results are analyzed and evaluated like

- ✓ Accuracy
- ✓ Memory usage
- ✓ Execution time

Accuracy

Show the comparison of accuracy for both existing and proposed method. The correctness is improved compared than existing system. Accuracy is define the ratio is corrected predictions and t eh whole number of predicted values.

$$Accuracy = \frac{\text{Number of correct predictiotions}}{\text{Total of all cases to be predicted}} = \frac{a+d}{a+b+c+d}$$

Where a – true positive, b - false negative, c – false positive, d – truenegative

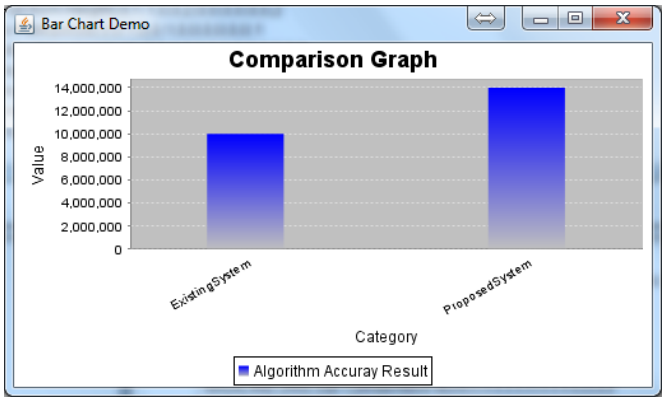

Figure 2. Accuracy

Execution time

It shows the comparison of execution time in Apriori, FP-Growth and FiDoop. The FiDoop provide the less execution time compared than other methods. That process is moved during their implementation as of lone memory division to dissimilar memory segment and it provide the delay until their run time. These are generated in hardware and mostly occur in general purpose OS.

CPU Time= I X CPI X T
I – number of instructions in the program
CPI – Average cycle pair instruction
T – clock cycle time

Memory usage

The memory utilization for Apriori, FP-Growth and FiDoop. The Fi-Doop used the less memory utilization compare than other methods.

VIII. CONCLUSION

I proposed a reliable and efficient MapReduce based parallel FIA, explicitly PATD that has exposed extensively efficient in conditions of runtime plus scalability, data communication like energy consumption. PATD takes the gain of efficient data partitioning method IBDP. IBDP permit for an optimized data placement on MapReduce. It allow PATD algorithm to carefully plus quickly mine extremely large databases. Such ability to utilize very low lowest supports is mandatory when dealing with Big Data and essentially hundreds of Gigabytes like what I have completed in our experiments. Our outcome show that PATD algorithm outperforms other existing PFIM alternatives, also makes the dissimilarity among inoperative and a victorious extraction.

IX. REFERENCES

[1]. Yaling Xun, Jifu Zhang, Xiao Qin, FiDoop-Dp Data Partitioning in Frequent Itemset Mining on Hadoop clusters, 2016.

[2]. I. Pramudiono and M. Kitsuregawa, "Fp-tax: Tree structure based generalized association rule mining," in Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. ACM, 2004, pp. 60-63.

[3]. X. Lin, Mr-apriori: Association rules algorithm based on mapreduce, a in Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on. IEEE, 2014, pp. 141"144.

[4]. S. Hong, Z. Huaxuan, C. Shiping, and H. Chunyan, aoeThe study of improved fp-growth algorithm in mapreduce, in 1st International Workshop on Cloud Computing and Information Security. Atlantis Press, 2013.

[5]. M. Liroz-Gistau, R. Akbarinia, D. Agrawal, E. Pacitti, and P. Valduriez, aoeData partitioning for minimizing transferred data in mapreduce,a in Data Management in Cloud, Grid and P2P Systems. Springer, 2013, pp. 1a"12.

[6]. Y. Xun, J. Zhang, and X. Qin, Fidoop: Parallel mining of frequent itemsets using mapreduce, IEEE Transactions on Systems, Man, and Cybernetics: Systems, doi: 10.1109/TSMC.2015.2437327, 2015.

[7]. W. Lu, Y. Shen, S. Chen, and B. C. Ooi, Efficient processing of k nearest neighbor joins using mapreduce,a Proceedings of the VLDB Endowment, vol. 5, no. 10, pp. 1016a"1027, 2012.

[8]. J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of massive datasets. Cambridge University Press, 2014.

[9]. B. Bahmani, A. Goel, and R. Shinde, Efficient distributed locality sensitive hashing,a in Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, 2012, pp.2174a"2178.

[10]. P. Uthayopas and N. Benjamas, Impact of i/o and execution scheduling strategies on large scale parallel data mining, Journal of Next Generation Information Technology (JNIT), vol. 5, no. 1, p. 78, 2014.