ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

FiDoop-DP: An Efficient Data Mining Technique on Heterogeneous Clusters

Juturu Chandana
M.Tech Scholar in Department of CSE
JNTUA College of Engineering, Ananthapuram, India

Abstract **- Data mining is the analysis step of the "Knowledge discovery in data bases process" [1]. Actually, it is very hard to mine item-sets which are frequently used in the transactions. To identify frequently used item-sets, parallel algorithms which are used for mining were developed. These parallel algorithms were developed to balance the data and to maintain equal partitions of data, among a group of nodes which are to be computed. Because of redundant transactions, there is a significant performance problem of parallel frequent item-sets mining. Therefore, a data partitioning technique has been developed. FiDoop-DP is a kind of data partitioning method which is used to divide the data based on item-sets of the transaction which are brought by the clients or customers. To know better about frequent item-sets i.e., products which are regularly sold together, an algorithm is used for time consumption while running data which is extremely large. This algorithm is named as Equivalence Class clustering and Lattice Traversal algorithm (ECLAT). This ECLAT algorithm is combined with the Map-Reduce functionality, and then it gives better solutions within small amount of time. At the same time ECLAT is combined with Local sensitive hashing technique for better performance of items which are present at locally present in the data nodes. By combining those two techniques, the performance of FiDoop increases. This is known by the time taken to mine frequent item-sets. The main goal of this paper is to mine the item-sets which are prominently used or sold in the market by that it can increase the sales of those products.**

Index Terms: Frequent Item-set Mining, Data Partitioning, ECLAT algorithm, Time management, Map-Reduce, Hadoop Cluster.

1. INTRODUCTION

Data mining is one of the most important technologies in this generation wherein it helps to mine our important data from data warehouses. Data warehouse is met for a storage repository which can store the data from many years ago. When Big data is booming in the industry i.e., data is growing with volume, velocity, varsity. There is a need to get conclusions of those data, to know whose play is important for our future purposes. For that, mining plays a prominent role to mine the data sets which are frequently used. It is known that mining the data from different repositories is also a tough task. Generally, parallel frequent item-sets mining techniques are introduced to trace the problem [1]. These techniques are focused on Load balancing but in order to retrieve the data within a time is impossible. This is all because of data is balanced at different locations using load balancing technique. There is no certain clear vision for data which is stored at particular place based on similarity.

In order to recover from this problem FiDoop-DP was introduced which is developed using MapReduce Programming model. FiDoop-DP (Data Partitioning) helps to enhance the execution of mining by using parallel frequent item set mining algorithms on Hadoop clusters [2]. FiDoop-DP acts as a platform to establish the correlation among different transactions to partition a huge dataset across different data nodes in a Hadoop cluster. In Hadoop cluster if data is aware of location where it is resided that helps to mine the data easily. But, this doesn't have any knowledge previously about the data which is stored at heterogeneous Hadoop clusters that is maintained across different nodes. In order to recover from this problem a scheme is implemented by adding FiDoop-DP data-aware technique on heterogeneous cluster at the level of HDFS data placement.

The organization of the paper is as follows: Section 2 presents the Related work. Section 3 presents the Proposed approach. Section 4 presents ECLAT scheme. Section 5 presents Experimental results and finally section 6 concludes the paper.

II. RELATED WORK

Normally, In Hadoop clusters, data partitioning schemes are used to divide the input data in equal parts but those schemes should partition the intermediate results. Due to this, data locality and data balance is completely ignored. So, FiDoop Data Partitioning helps to partition the input

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

transactions which help to decrease the amount of data transferred through the network during the shuffling phase. It is also helps to minimize local mining load. Traditional Frequent Item set Mining (TFIM) is a process of considering redundant transaction transmissions and redundant mining tasks. Redundant transactions consideration is the main reason for high network load and redundant mining cost. In order to address this issue FiDoop proposes partition transactions by considering correlations among transactions and items before the mining process. These results, transactions with highest similarity are grouped into one partition which in turn prevents the transactions from repetition at remote nodes. FiDoop adopts Voronoi diagram based data partitioning technique at second map reduce job, which helps to minimize unnecessary redundant transaction transmissions. Voronoi diagram divides the space into regions which are called Voronoi cells. It also considers set of points which are referred as pivots. For each pivot, there is a corresponding region consisting of all objects closer to it than to the other pivots. Based on the characteristic of Frequent Item set Mining (FIM), FiDoop adopt the similarity as the distance metric between the two transactions in voronoi diagram.

In traditional FiDoop–DP adopts the FP-growth algorithm in that initially it constructs FP-tree. Based on FP-tree it maps the items with another item which are avail in the transaction [4]. These items are grouped to form a set based on the support. FP-Growth algorithm is more efficient than Apriori algorithm since, it improves item-set mining by examining the database only twice. Based on frequent items sets Voronoi diagram is built for similarity. As Voronoi diagram considers the similarity as the distance metric, FiDoop adopts Jaccard's similarity as a distance metric [3]. It is a kind of statistic which helps to measure the distance between datasets. When the Jaccard's similarity is high, then the two data sets are very close to each other. In order to measure the distance between the data sets, first plot a model for each transaction in a database in terms of Sets.

In FiDoop, K-means selection technique is used for the selection of pivots. The procedure of selecting pivots is conducted as a data pre-processing phase. The main target of this technique is to partition the objects into clusters so that each object belongs to a cluster which is having the nearest mean. The result of this technique can be applied to partition the data space into Voronoi cells. By considering the computational cost problem of K-means, FiDoop performs the sampling on the transaction database before running the algorithm. Selection of initial pivots plays a vital role in clustering performance [7, 8]. So, K-means++ which is an extension to K-means is used for selecting pivots. Once K data clusters are generated, select center point of each cluster as a pivot for the voronoi diagram based data partitioning.

Once selection of pivots task is completed, there should be a need for calculating the distances among remaining objects to these pivots to determine a partition to which each object belongs to. So, FiDoop uses MinHash and Locality Sensitive Hashing based strategy for grouping and partitioning process. MinHash helps to perceive quick solution to calculate similarity between two sets [5, 6].MinHash is a best technique for large-scale clustering process. MinHash uses very smaller representations named as "signatures" for replacing the large sets composed of "minhash" of the characteristic matrix. This can simply call it as a matrix representation of data sets. Then, MinHash calculates an expected similarity of two data sets based on the signatures.

Locality Sensitive Hashing (LSH) is another technique used for data partitioning which helps to upgrade the performance of MinHash [4]. Using these techniques can avoid the comparisons of a large number of element pairs. Through MinHash, this can repeatedly evaluate an enormous pairs. But, using LSH it scans all the transactions at once and identify all the pairs that are likely to be similar. By using LSH it helps to map the transactions in the feature space to a number of buckets, this helps to find the similar transactions are likely to be mapped into same buckets. So using these more similar items are hashed into a same bucket based on the items which are having high probability than the dissimilar items.

III. PROPOSED APPROACH

Generally, Heterogeneous systems defined as the system which provides results by considering individual systems which have their own database. Integration of those systems is considered at last stage. Due to heterogeneous systems results are independent; there is a need of translation for communication between different DBMS. In order to communicate, requests are to be made in the DBMS language at their individual locations. If the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

hardware is different and the products of DBMS are same then, translation is easy. But, if the products of DBMS were different, the translation is complicated and involves mapping of the data structure from one model to another model. If both products hardware and software are different, then the above two types of translations need to be used. This makes the processing complex, in order to overcome this gateways are used, which can convert the language and model of each different DBMS into the language and model of the required systems.

The Hadoop implementation these days assumes the nodes which are to be computed in a cluster are homogeneous. But, there is also another cluster called Heterogeneous cluster whereas the nodes have different computing capacity. Data partition techniques are required in order to partition the input and intermediate data which are supported by the required computing capacities of the nodes that are stored in the cluster. Hadoop Distributed File System (HDFS) makes Hadoop MapReduce applications to transfer operations to be processed were sent towards the nodes where the actual application data resides. In Heterogeneous cluster, the computing capacity of nodes is different from one another. A high-speed node can process the data on a local disk of a node faster than the low-speed node. After a high-speed node completes the processing of its data, it must share the unprocessed data located in one or more low-speed nodes which helps to process the data fastly. Nodes are needed to be cautious about transferring the load from high-speed node to low-speed node because, if the transferred data is very large, it will affect the performance of Hadoop. In order to enhance the performance of Hadoop in heterogeneous clusters, it is necessary need to minimize the data transfer in between high-speed and low-speed nodes. This can be attained by selecting proper data partitioning scheme that will distribute and store data across different heterogeneous nodes by considering their computing capacities.

In this paper, FiDoop data partitioning technique had done in heterogeneous Hadoop clusters by using Equivalence class Clustering and Lattice Traversal algorithm. FiDoop main goal is to spilt-up input transactions to reduce local mining load as well as decrease the amount of data transferred through network at shuffle phase. So, consider the input data which has N number of map reduce jobs because of the clusters are heterogeneous. Every map reduce job consists of set of transactions (t1,t2...tn) and these transactions can be divided as a set of chunks (c1,c2...cn). Each one has map tasks (m1,m2...mn) and reduces tasks (r1,r2...rn) that are running on a heterogeneous cluster. Based on this, an intermediate key-value pairs set produced by the mappers as ((g1,d1), (g2,d2),..(gn,dn)) whereas, di is a collection of transactions which belongs to the group gi. After map tasks are completed, the shuffle phase applies the partitioning function to assign intermediate key-value pairs to reduce tasks based on the keys. If the intermediate key-value pair is partitioned into a reducer running on remote node, then there is a need of intermediate data shuffling.

FiDoop incorporates voronoi diagram based data partitioning, in order to reduce unnecessary redundant transaction transmissions [3]. The main idea of Voronoi diagram based partition is for every given dataset D; it selects O objects as pivots. Then all the objects are spitted into k disjoint partitions. Each object is assigned to the partition with its closest pivot. By considering this process, it covers the entire data space by splitting into K cells. FiDoop considers distance metric for dividing the similar data objects to the same partition. Jaccard's similarity is used as a distance metric for division.

MinHash is used as a partitioning strategy which is foundation for Locality Sensitive Hashing (LSH) based partitioning. LSH scans all the transactions at once in order to identify the similar pairs. Similar transactions are mapped into bucket. Likewise many similar transactions are mapped to many similar buckets. LSH ensures that two similar points are mapped into the same bucket with high probability as well as, it guarantees that two dissimilar points are less likely to be mapped into same buckets.

In order to mine, the most frequent item sets in Hadoop heterogeneous clusters ECLAT algorithm is used. This algorithm represents data from horizontal to vertical data format. Along with set intersection ECLAT uses depth first search algorithm. The procedure of this algorithm is as follows: At initial stage, the algorithm considers the transaction table with transaction identifiers and item set. Next, it generates item wise mapping with transaction ID with item set-1. It will continue the process by generating item wise mapping with transaction ID with item set-2. At final stage, it combines item set-1 and item set-2 with minimum support. The process is same for each and every transaction and at every node in Hadoop heterogeneous cluster. Using this algorithm for mining, each item set in the databases is no need to scan for every time. Using these method, depth-first search memory requirements is reduced. ECLAT algorithm can be considered as efficient

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

algorithm for mining frequent Item set on a large set of transactions which are performed in a Hadoop heterogeneous cluster.

A. ECLAT SCHEME

In this section, the implementation details of ECLAT based FiDoop-DP running on Hadoop clusters is presented, which consists of four steps (i.e., one sequential-computing step and three parallel Map Reduce jobs). Specifically, before launching the FiDoop-DP process, a pre-processing phase is performed in a master node to select a set of (k) pivots which serve as an input of the second MapReduce job that is responsible for the Voronoi diagram-based partitioning.

ECLAT means Equivalence class clustering and bottom up lattice traversal. For a transaction, in order to mine the recurrent item sets ECLAT algorithm is used. ECLAT helps to represent the data in vertical data format. ÉCLAT algorithm is normally a depth-first search algorithm based on set intersection. Generally, RARM and Apriori techniques uses horizontal data format (TransactionId, Items) wherein transaction identifiers are explicitly listed. Whereas ECLAT algorithm uses vertical database format and the data which is represented in vertical format (Items, TransactionId) Items and their corresponding transactions are maintained. All frequent Item-sets can be computed with intersection of TID-list. In first scan of database, a TID list (TranscationId) is maintained for each single item. k+1 Item-set can be generated from k Item-set using Apriori property and depth first search computation. (k+1) Item-set is an intersection of TID-set of frequent k-Item-set. This process continues, until no candidate Item-set can be found. One advantage of ECLAT algorithm is, we don't need to scan the database for counting the support values of k+1 item set.It is because support count information is already obtained from k Item-sets. This algorithm helps to avoid the overhead of calculating all the subsets of every transaction and comparing them against the candidate hash tree during support counting.

IV. EXPERIMENTAL RESULTS

The experimental results are carried out by initially taking the sample dataset [9] consists of the item-sets, which consists of the integer value of the data in the particular item in this dataset.

Procedure for mining:

1.Scan the database for items.

Transaction identifiers	Item sets
1	I0, I1, I2, I4
2	I0, I3, I5, I6
3	I1, I2, I3, I4, I6
4	I1, I4, I5, I6
5	I0, I1, I2, I5, I6
6	I1, I2, I5, I6

Table 1: Horizontal Data Format of a Database

2. Convert horizontal data format to vertical data format

Item	Transaction id set
I0	1, 2, 5
I1	1, 3,4, 5, 6
I2	1, 3, 5, 6
I3	2, 3
I4	1, 3, 4
I5	2, 4, 5, 6
I6	2, 3, 4, 5, 6

Table 2: Vertical Data Format of a Database

3. Calculate Support for all items

Support $=\frac{a}{b} * 100$ (in percentage)

Where, 'a' is No. of times, of that intended items come along with another intended item that are available in the transactions, 'b' is Total No. of transactions.

Items	Support
I0	3
I1	5
I2	4
I3	2
I4	3
I5	4
I6	5

Table 3: Support for items

4.Based on the items support, least valued support is considered as "minimum-support" for item-sets. Therefore, minimum-support is 2 i.e., 40%.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

5. Compare min-sup with support for each item, if the support is less than or equal to min-sup, then eliminate the item.

Table 4: After elimination δ1

Items	Support
I0	3
I1	5
I2	4
I4	3
I5	4
I6	5

6. Combine item {I0} with each remaining item ({I0,I1}, {I0,I2})..., and then calculate support. Repeat for all item

Table 5: Combined support with each item

Items	Support	Items	Support	Items	Support
{I0,I1}	2	{I1,I2}	4	{I2,I4}	4
{I0,I2}	2	{I1,I4}	3	{I2,I5}	2
{I0,I4}	1	{I1,I5}	3	{I2,I6}	4
{I0,I5}	2	{I1,I6}	4		
{I0,I6}	2				

Items	Support	Items	Support
{I4,I5}	1	{I5,I6}	4
{I4,I6}	2		

7. Compare min-sup with support for each item, if the support is less than or equal to min-sup, then eliminate the item.

Items	Support	Items	Support	Items	Support
{I1,I2}	4	{I2,I4}	4	{I5,I6}	4
{I1,I4}	3	{I2,I6}	4		
{I1,I5}	3				
{I1,I6}	4				

Table 6: Comparision of minimum support with item sets support δ 2

8. Repeat step-6 for frequent 3 item-sets.

Items	Support
{I1,I2,I4}	2
{I1,I2,I5}	2
{I1,I2,I6}	1
{I1,I4,I5}	3
{I1,I4,I6}	2
{I1,I5,I6}	3
{I2,I5,I6}	2

Table 7: Combining with other items

9. Repeat step-5.

Items	Support
{I1,I4,I5}	3
{I1,I5,I6}	3

Table 8: Final set δ3

10. Finally {I1,I4,I5} , { I1,I5,I6} are frequent item-sets.

The graph is plotted in fig.5 based on the comparison between FP-Growth algorithm and ECLAT algorithm. The parameters are minimum support and the time taken to run a dataset. Hence, proved that ECLAT with Local Sensitivity Hashing Technique runs faster than FP-Growth algorithm.

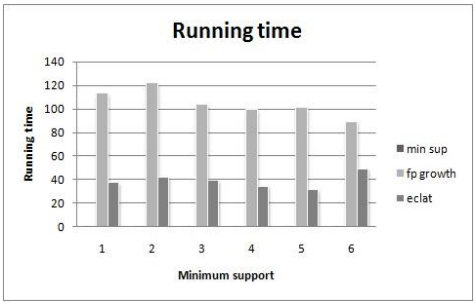

Figure 5: Graphical representation of minimum support and running time between ECLAT and FP-Growth algorithms.

The Graph plotted in fig.6 shows the comparision between ECLAT and FP-Growth by taking voronoi diagram as bases for clear representation.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

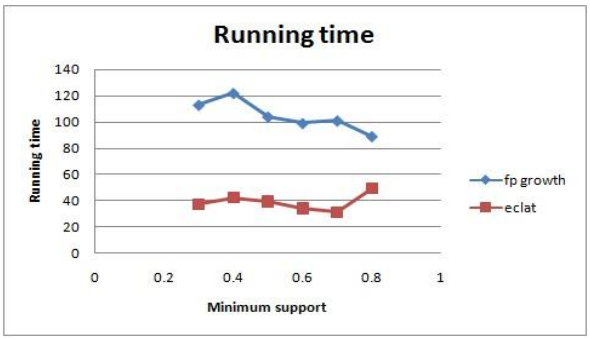

Figure 6: Graphical representation of minimum support and running time between ECLAT and FP-Growth algorithms based on Voronoi diagram

6. CONCLUSION

In Heterogeneous Hadoop cluster, Data partitioning and mining the data from different systems are the main issues. This is because of; different nodes need to be considered for processing the data. So, care should be taken while partitioning the data across different nodes. There may be nodes which are running fast as well as running slow. If partitions of data are not done perfectly, it reflects the performance of Hadoop. Data must be partitioned, thus in a way that, processing can be completed within feasible time even if the nodes are high-speed or low-speed. Else, after completion of high-speed nodes processing, we need to assign those nodes to low-speed nodes for sharing the data processing load. But, it is hard when the transferred data is very large. So, in order to overcome this problem, it is need to be cautious at initial stage which is data partitioning stage. In this paper FiDoop- Data partitioning technique is used to split the data across different nodes in heterogeneous cluster, with the help of ECLAT algorithm for mining. Automatic Updating of database can be done for further future purpose.

REFERENCES

[1] M. J. Zaki, "Parallel and distributed association mining: A survey," Concurrency, IEEE, vol. 7, no. 4, pp. 14–25, 1999.

[2] Yaling Xun, Jifu Zhang, Xiao Qin and Xujun Zhao "FiDoop-DP: Data Partitioning in Frequent Item-set Mining on Hadoop Clusters" IEEE Transactions on Parallel and Distributed Systems, pp.7-14, 2016.

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

[4] I. Pramudiono and M. Kitsuregawa, "Parallel fp-growth on pc cluster," in Advances in Knowledge Discovery and Data Mining. Springer,
2003, pp. 467–473.

[5] A. Stupar, S. Michel, and R. Schenkel, "Rank reduce–processing knearest-neighbor queries on top of mapreduce," in Proceedings of the 8th Workshop on Large-Scale Distributed Systems for Information Retrieval. Citeseer, 2010, pp. 13–18.

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

[7] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 7, pp. 881–892, 2002.

[8] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern Recognition Letters, vol. 31, no. 8, pp. 651–666, 20

[9] http://www.phillipefourier.comAccessed on June 22, 2017.