Global Journal of Computer Science and Technology: B
Cloud and Distributed
Volume 16 Issue 3 Version 1.0 Year 2016
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Comparative Analysis of Mapreduce Framework for Efficient Frequent Itemset Mining in Social Network Data

By Suman Saha & Md. Syful Islam Mahfuz
University of Chittagong

Abstract- Social networking sites are the virtual community for sharing information among the people. It raises its pularity tremendously over the past few years. Many social networking sites like Twitter, Facebook, WhatsApp, Instragram, LinkedIn generates tremendous amount data. Mining such huge amount of data can be very useful. Frequent itemset mining plays a significant role to extract knowledge from the dataset. Traditional frequent itemsets method is ineffective to process this exponential growth of data almost terabytes on a single computer. Map Reduce framework is a programming model that has emerged for mining such huge amount of data in parallel fashion. In this paper we have discussed how different MapReduce techniques can be used for mining frequent itemsets and compared each other's to infer greater scalability and speed in order to find out the meaningful information from large datasets.

Keywords: social networks, frequent itemsets mining, apriori algorithm, mapreduce framework, eclat algorithm.

GJCST-B Classification: *C.1.4,C.2.1,C.2.4 J.4*

COMPARATIVEANALYSISOFMAPREDUCEFRAMEWORKFOREFFICIENTFREQUENTITEMSETMININGINSOCIALNETWORKDATA

Strictly as per the compliance and regulations of:

Comparative Analysis of Mapreduce Framework for Efficient Frequent Itemset Mining in Social Network Data

Suman Saha [α] & Md. Syful Islam Mahfuz [σ]

Abstract- Social networking sites are the virtual community for sharing information among the people. It raises its popularity tremendously over the past few years. Many social networking sites like Twitter, Facebook, WhatsApp, Instragram, LinkedIn generates tremendous amount data. Mining such huge amount of data can be very useful. Frequent itemset mining plays a significant role to extract knowledge from the dataset. Traditional frequent itemsets method is ineffective to process this exponential growth of data almost terabytes on a single computer. Map Reduce framework is a programming model that has emerged for mining such huge amount of data in parallel fashion. In this paper we have discussed how different MapReduce techniques can be used for mining frequent itemsets and compared each other's to infer greater scalability and speed in order to find out the meaningful information from large datasets.

Keywords: social networks, frequent itemsets mining, apriori algorithm, mapreduce framework, eclat algorithm.

I. Introduction

Social network is a virtual network that allows peoples to create a public profile into under a domain so that peoples can communicate with each other's within that network. It has obtained remarkable attention in the last few years. Many social networking sites such as Twitter, Facebook, WhatsApp, Instragram, LinkedIn, Google+ through the internet are frequently used by the people. People can share information, news and many others through these social networks. Facebook is the most popular social sites which had more than 1.59 billion people in as of their last quarter [11]. Other sites like Instagram had 400 million peoples in September 2015, Twitter had 320 million peoples in March 2016, Google+ had 300 million peoples in October 2013, and LinkedIn had 100 million peoples in October 2015 [11]. Analysis can be performed over such Big data which plays a significant role to improve the productivity of different companies in both public and private sector. Storing huge amount of data won't have any value without KDD (Knowledge process in Database) which is process of finding information from database and extracted knowledge can be used for making effective business decision [12]. Frequent itemsets mining is a popular method to extract the frequent itemset over a dataset. It also plays an important role in mining associations, correlation, sequential patterns, causality, episodes, multidimensional patterns, max patterns, partial periodicity, emerging patterns and many other significant data mining tasks [2].

Author α: Is now serving as a Lecturer in CSE Dept. at Bangladesh University of Business and Technology (BUBT). He received his B.Sc (Engg.) degree in Computer Science and Engineering from University of Chittagong, Bangladesh in 2011. His research interests are Data Mining, Pattern Recognition, Image Processing, Wireless Ad Hoc Networks and Algorithms.
e-mail: sumancsecu04@gmail.com
Author σ: Received the B.Sc. degree in Computer Science and Engineering from Patuakhali Science and Technology University, Bangladesh in 2012. Currently, he is a Lecturer of Computer Science and Engineering at Bangladesh University of Business and Technology. His teaching and research areas include Data Mining, Wireless Transmission, Neural Network and Embedded System design.
e-mail: mahfuzisl@pstu.ac.bd

II. Research Background

Social networks generates huge amount of data possibly terabytes or more. These multidimensional data often referred to as Big data. So it is not efficient technique for mining such Big data on a single machine because of its limited memory space, RAM speed, and Processor capacity. So researchers have emphasized on parallelization for mining such data set to improve the mining performance. But there are several issues related with parallelization such as load balancing, partition the data, distribution of data, Job assignment, and data monitoring that need to solve. MapReduce framework has been introduced to solve this problem effectively. Cloud computing provides unlimited cheap storage and computing power so that it provides a platform for the storage and mining mass data [1].

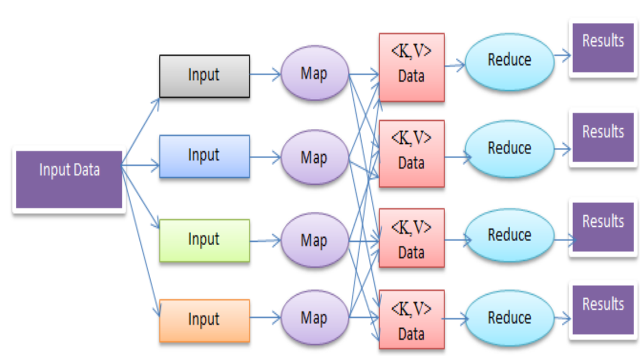

Figure 1: MapReduce Framework

MapReduce framework was proposed by Google in 2014. It is used for processing a large amount of data in parallel manner. It hides the problems like parallelization, fault tolerance, data distribution, and load balancing which allow users to focus on the problem without worrying about parallelization details [1]. Basically MapReduce framework works on key-value pairs. The input data is divided into several parts and stored into the different nodes. It uses two functions, one is map function and another is reducing function. Map function takes key-value pairs from each node as input and generates key-value pairs which indicate local frequent item set as output. Reduce function takes these local frequent itemsets as input and combine these key-value pair and generates output as key-value pairs which indicates the global frequent item set. The above process can be easily and effectively implement by using Hadoop MapReduce frame.

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner [13]. Hadoop is open software that built on Hadoop Distributed File Systems (HDFS). MapReduce framework and HDFS are running on the same node.

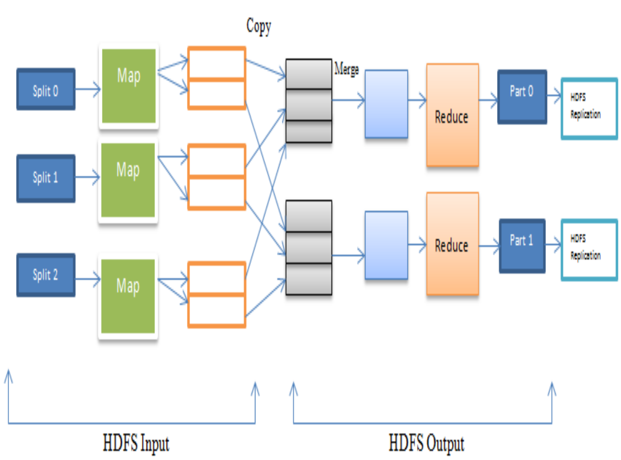

Figure 2: Hadoop MapReduce Framework

In MapReduce, a large dataset is broken into multiple blocks. Each block is stored on distinct nodes to form cluster. In Figure 2, dataset is partitioned into three blocks. Multiple maps (here three maps) are running simultaneously on different parts called split .One maps for each blocks. A local disk is used to store the output of the each map. A local disk has multiple partitions where output of maps is stored in all partitions. One partition corresponds to each reducer in the framework. Then one partition of each local disk is copied into each reducer. Here output maps are stored into three local disks. Each disk has two partitions. Partitions of the local disk are copied into two reducers.

III. Preliminaries

a) Problem Definition

Let D be a database that contains N transactions. Assume that we have S number of nodes. Database D with N transactions is divided into P equal sized blocks $\{D_1, D_2, D_{3......,}D_P\}$ automatically and assign each of the block D_i to the nodes. Each of the nodes contains N/P transactions. Consider an itemset I in the database D. Then I.supportCount indicates the global supportCount of I in D. We can call I is globally frequent if it satisfy the following conditions

supportCount $\geq s \times N$ where s is the given minimum support threshold.

b) Data Layout

Consider an itemset $I = \{I_1, I_2, I_3, I_4, I_5\}$ and D be database with 5 transactions $\{t_1, t_2, t_3, t_4, t_5\}$. Data Layout can be Horizontal Layout or Vertical Layout. Horizontally formatted data can be easily converted to Vertical format by scanning the database once. Following figures shows how Horizontal or Vertical can be represented of the above itemset and database transactions.

TID	Itemset
t_1	I_1, I_3, I_4
t_2	I_1, I_2, I_4, I_5
t_3	I_1, I_5
`t_4	I_1, I_4, I_5
t_5	I_2, I_3, I_5

Horizontal Layout

Item	TID set
I_1	t_1, t_2, t_3, t_4
I_2	t_2, t_5
I_3	t_1, t_5
I_4	t_1, t_2, t_4
I_5	t_2, t_3, t_4, t_5

Vertical Layout

Figure 3: Data Layout

These two different formats have the different way of counting the support of the itemset. In horizontal data format, whole database needs to scan k times to determine the support of itemset. For example, if we want to count the support of the itemset $I = \{I_1, I_2, I_3, I_4, I_5\}$ then we need to scan the all transactions from t_1 to t_5. After scanning then we get the support for the item $I_1 = 4$, $I_2 = 2$, $I_3 = 3$, $I_4 = 3$, $I_5 = 4$. In the similar way, if want to find the support of the 2-itemset for example (I_1, I_5) then again we need to scan the database and get support (I_1, I_5) is 3. But if we consider the vertical format then it needs only intersection of the TID list of itemset to get the support of the itemset. For example, If we want to get the support of both I_1, I_5 then we have to perform the intersection operation of $\{t_1, t_2, t_3, t_4\}$ with $\{t_2, t_3, t_4, t_5\}$ and get output of $\{t_2, t_3, t_4\}$. So support (I_1, I_5) is 3. So vertical data format reduces the number of times to scan the database very effectively.

c) Apriori Algorithm

Apriori algorithm is used for frequent itemsets mining and association rule learning over transactional databases [16]. It was proposed by R. Agrawal and Srikant in 1994. Apriori uses a Breadth-first search approach where frequent subsets are extended one item at a time, and groups of candidates are tested against the data. At first scanning the database D and count each item. Items that satisfy the minimum support are conceded frequent 1-itemset.Then generates candidates of 2-freqeunt itemset from frequent 1-itemset. Scan the database again for counting the frequency of candidate 2- itemset, compare candidate support count with minimum support and determine the 2-frequent itemset. In the similar way we can determine the frequent k-itemset and generates candidate k+1 itemsets by applying support and threshold conditions. Apriori algorithm is two-step process one is join and another is prune. Candidate k-itemset is generated by joining the k-1 frequent itemset. And monotonic property is exploited to prune the candidates which are infrequent [5]. This process continues until the candidate itemset is not NULL. Limitations of Apriori algorithm are finding the each frequent itemset requires one full scan of the database and candidate generation generates large number subsets.

d) Eclat Algorithm

Eclat algorithm was proposed by ZAKI in 2000 for finding frequent itemset. Eclat uses vertical formatted data rather than horizontal layout. As a result no need to scan the database to find the support of (k+1) itemsets, for k>=1 which achieves a good performance. Eclat is based on depth-first search to traverse the prefix tree. Eclat algorithm is very much similar with Apriori algorithm. Similar to Apriori frequent 1-itemset is generated by scanning the database D. Candidate 2-itemset are generated from frequent 1-itemset. Frequent 2-freqeunt itemset are generated from candidate 2-itemset by clipping the infrequent itemsets. This process continues until candidate itemset is not NULL. Different thing of Eclat from Apriori is that Eclat algorithm partition the search space and creates multiple non overlapping sub spaces. Monotonic property states that if an itemset or path in the tree is infrequent then all of its sub-trees are infrequent and same are pruned; only frequent itemsets are considered as prefix which gets added in a tree [5].Same prefix type's itemsets are categorized to the same class and candidate itemsets can be conducted only in the same class. Equivalence classes improve the efficiency of collecting candidate itemsets and also minimize the occupation of storage space. Eclat algorithm has the following limitations 1) Generation of candidate itemset is more than of Apriori because prior knowledge may not enough to clip the candidate itemsets. 2) If the itemset is much long then a great deal of time is needed to determine whether two itemset can be joined or not. 3) For the itemset of larger transactions, calculation of intersection is not much efficient. Although Eclat has some limitations but it has high efficiency and very effective.

IV. Different Mapreduce Technique for Finding Frequent Itemsets

a) PAriori algorithm

Parallel implementation of Apriori algorithm is very easy to implement in Map Reduce framework [23]. The whole database is partitioned into subparts and subparts are assigned into different node. As a result parallel counting of each node is possible. Combiner calculate locally intermediate sum of the data to reduce the data size and transformed over the network. Hash tables are used to check the data items that satisfy minimum support. These frequent itemset are stored in hash table and assigned to all the working processes. After that reducer finds the global frequent itemsets from the local itemset. These global frequent itemset at step i are inputted to the mapper for the next step i+1 and repeat the same procedure. Before inputted to the mapper, candidate itemset are generated from the global itemset and apply prune technique on the candidate itemset to reduce its size. Following figure shows the parallel implementation of Apriori algorithm for finding the frequent itemsets.

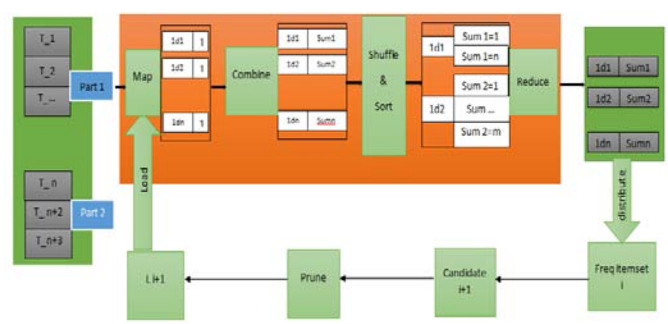

Figure 4: Parallel implementation of Apriori algorithm

b) MRApriori algorithm

Parallel implementation algorithm provides good scalability but repeated scanning of the whole database is still needed. MRAriori improves over the PAriori is that it needs only one full scan of the database. It scans only the intermediate data repeatedly that generally reduces per iteration. Singh (2014) proposed the MapReduce Apriori algorithm for finding the frequent itemsets [24]. Two main parts of Apriori algorithm. One is generating candidate itemsets and another is generating frequent itemsets from candidate itemsets. MRApriori algorithm is based on HDFS. HDFS divides the entire database into blocks and blocks are assigned to the different mappers running on multiple nodes. The input to the mappers is the (key, value) pairs where key is the transactional ID and value is the list of items. Output of the mappers is also (key', value') pairs where key' is the item in the transaction and value' is 1.Combiner performs the local for the key' of the same key value and inputted to the shuffle and exchange part. In shuffle and exchange part, given output from the combiner it makes a list of items of the form (key', list (value")) pairs and passes to the reducers. Reducers takes the pairs as inputs, sum up the values of respective keys and outputs the pairs (key', value"') pairs where key' is item and value'''as support count must satisfy the minimum support threshold. By merging the the outputs from all reducers frequent 1-itemset can be generated. If we find the frequent 2-itemsets then at first candidate 2-itemsets will be generated and then we have to find out frequent 2-itemsets from the candidate itemsets. To find the frequent k-itemsets, frequent 1-itemsets are inputted to the mapper and mapper generated candidate k-itemsets. A candidate itemsets is selected as key and value is 1 if mapper finds that item in the transaction list which is assigned to the mapper. All the remaining procedures are same.

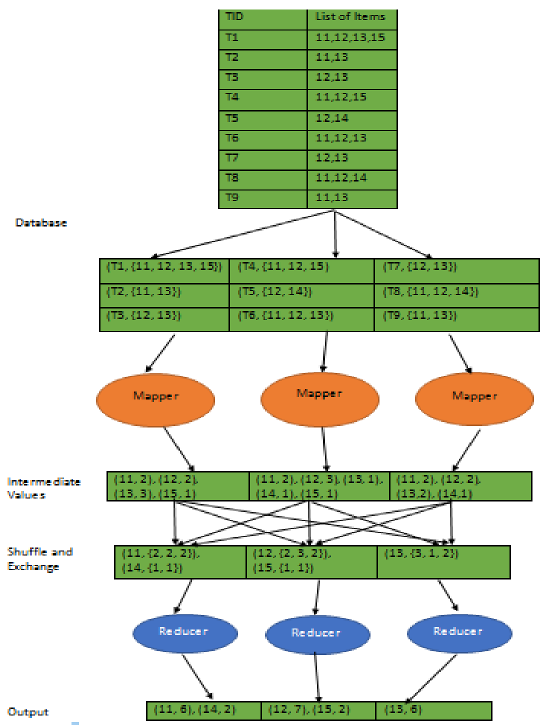

Figure 5: MRAriori procedure

c) *Parallel FP-growth Algorithm*

Parallel FP-Growth is the parallel version of FP-Growth [21]. Let we have N different computers. In sharding, Database DB transaction is partitioned into different parts called shard and stored on N different computers. In parallel counting step, generate the support values for all the items in DB using Map-Reduce pass. Each mapper loads a shard and discovers the Vocabulary I. Finally result is stored in F-list. Divide the all the items in F-list and generate group-dependent G-list in grouping items during grouping items step.

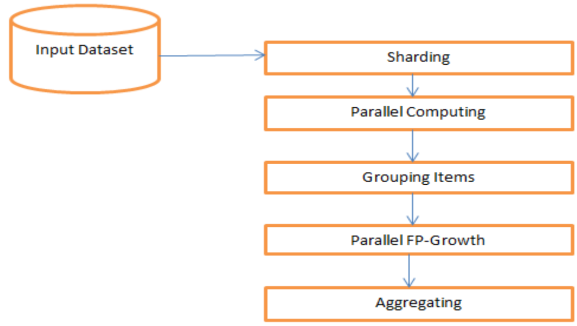

Figure 6: Block diagram of Parallel FP-Growth approach

Both F-list and G-list are small in size and possible to compute in a single computer. Each G-list has unique identifier (g_id).Parallel FP-growth works in two steps: one is mapper and another is reducer. Group dependent transactions are generated in mapper step. At first mapper reads the G-list. Each mapper is fed one shard and gives outputs of one or more key-value pairs where key indicates the group_id and value indicate the generated group dependent transaction list. For each group_id, map reducer creates a shard of group dependent transactions from all group dependent transactions. Then reduces processes each shard one after another. During the process, at first it creates a local FP tree and then growth its conditional FP-trees recursively while it may generates discovered pattern during this process. Finally results in parallel FP-growth are aggregating to generate the final result.

d) *Balanced FP-Growth*

Balanced FP-growth consists of two rounds of Map Reduce [22]. In Balanced FP-Growth, two major improvements are done over the Parallel FP-Growth. One is balanced partition of the database D to improve the parallelization and other is no aggregating operation is needed for finding frequent itemsets. Balanced FP-Growth consists of the following steps:

Sharding: Partition the database D into successive partitions and assigned into the different nodes. If we use Hadoop Map Reduce then just copy the database into the Hadoop Distributed File System. Hadoop automatically perform the Sharding.

Parallel Counting: One MapReduce technique is used for counting the entire items. One shard is inputted to exactly one mapper. The input is <key, value= Ti> pair to the mapper where Ti ⊂ database transaction. and output is also <key', value'> pair. Reducer calculates the sum of all the values that have the same key' and outputs <key', sum (values')> pair. Output of this phase frequent items called F-lists that is sorted in descending order based on frequency.

Balanced Grouping: To improve the parallelization of the overall mining, balanced grouping partition the F-list into G-list and balanced the load among the groups. It can be divided into two steps.

i. *Mining the load estimation:* In this step, estimate the load unit which is amount of work of running FP-Growth on conditional pattern base of each frequent item

ii. *Balanced Partition:* In this step, fairly partition the load units among different groups.

Parallel FP Growth: This step uses MapReduce phase again. In map phase, Original database D transactions are transformed to new group dependent transactions and construct FP tree. And the reducer recursively done the FP-Growth on the group dependent transactions in the reduce phase.

e) Dist-Eclat

In general, we partition the large database into equal sized sub database. Then mining the sub databases separately and combined them to obtain local frequent item sets. Finally all local frequent item sets are combined and use prune method to obtain global frequent item sets. As a result this approach comes with large communication cost and is prohibitive to implement in Hadoop. For effective mining and overcome this situation Distributed version of Eclat (Dist-Eclat) partition the search space rather than data space [20]. Dist-Eclat use depth first search approach for finding frequent item sets. As a result we need to store only limited number candidate item sets in memory.

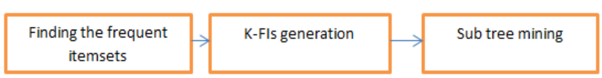

Figure 7: Dist-Eclat Procedure

Dist-Eclat works in the following three steps:

Finding the frequent item sets: At first vertical database is equally partitioned to create the sub database called shards and assigned them to the mappers. Mappers find the local frequent item sets from the shards. Combined all local frequent item sets which is done input of the reduce phase.

K-FIs Generation: This step generates k^{th} frequent itemsets. Each mapper is assigned the combined form of local frequent item sets. Then mapper finds the kth sized superset of the items using Eclat method. Finally a reducer assigns the frequent itemsets to the individual mappers.

Subtree mining: Eclat algorithm is used for mining the prefix tree from the assigned subsets.

f) BigFIM

There are some limitations associated with Dist-Eclat method. Firstly in Dist-Eclat, mapper needs the whole datasets to generate FIs. As a result large number tid-list may not fit in the memory. Secondly, mapper needs the complete dataset for mining the sub tree which is prohibitive in this Dist-Eclat. To overcome this limitation, BigFIM method can be used [20]. It is combination of both Apriori and Eclat algorithm for mining the large dataset. BigFIM consists of the following steps:

Generating k-FIs: BigFIM overcomes the difficulties arises for large tid list by constructing k-FIs using Apriori algorithm. At first database is partitioned into sub parts and each mapper receives sub part of the database. Mapper use Apriori algorithm to find out the local frequent item set. These local frequent item set inputted to the reduce function. Reducer combines all local frequent item set, pruned the item set and find out the global frequent item set. This global frequent item set are redistributed to all mappers as a candidate item set for the next step. This process is repeated to k times to find the k+1 FIs

Finding Potential Extensions: This step obtains tid-lists for (k+1)-FIs. Local tid-list are collected from all mappers by the reducer and combines them for generating global tid-list. And assign the computed global tid-list as a complete prefix groups to the mappers.

SubtreeMining: Here, mapper performed on individual prefix groups. Eclat algorithm is applied to mine the prefix groups as conditional database that fits into a memory for frequent item set.

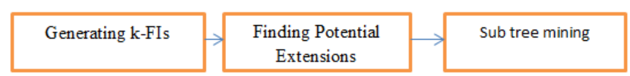

Figure 8: BigFIM Procedure

g) ClustBigFIM

ClustBigFIM provides the hybrid approach which is the combination of parallel k-means, Apriori, and Eclat algorithm [5]. It gives an approximate result that is very much close to original result with faster speed. ClustBigFIM has the following four steps for finding frequent itemsets from large datasets.

i. *Find Clusters*

At first clusters are generated using parallel k-means algorithm based on Compute_Dist function and combiner function.

ii. *Finding K-FIs*

Apriori algorithm is used for mining generated clusters in step 1.Mapper find the local support and Reducer calculate the global supports.Upto certain length k, Apriori used to find frequent k-length itemsets. But for higher length k+1, use pruning technique on the candidate itemsets to generate frequent itemsets.

iii. *Generate Single Global TID list*

From the generated prefixes, built a prefix tree and obtain tid_lists for k+1 frequent itemsets .Mappers computes the local tid_lists and reducer compute the single global tid_lists.

iv. *Subtree Mining*

Prefix groups are assigned to the mappers which is the conditional database that fits completely into the memory. Subtrees are mined independently by

Global Journal of Computer Science and Technology (B) Volume XVI Issue III Version I Year 2016

the mappers using depth first search. Longer frequent itemsets as prefixes are used for better load balancing.

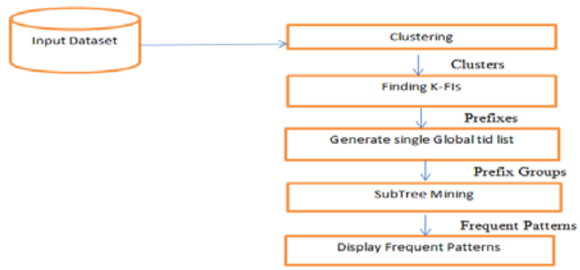

Figure 9: ClustBig FIM Procedure

V. Comparative Analysis

PApriori algorithm is very easy to implement in Map Reduce framework. It provides good performance and efficient for large database. But user needs to give number of reducers and repeated scanning of the full database in PApriori. MRApriori technique overcomes this situation of repeated scanning. It scans only the intermediate data repeatedly that generally reduces per iteration. It is also efficient and provides good performance for large database. But processing time of MRAprioi is same as PAriori. No significant reduction was done for faster execution in MRApriori over PApriori. Parallel version of Parallel-FP Growth is scalable. But if we consider this technique based on memory and speed then it is not efficient. Balanced FP-Growth is improved version of Parallel- FP Growth. It balances the load distributed among the nodes. And also executes faster than the parallel FP-Growth using singletons. But the way this technique partition the search space is not efficient. Dist-Eclat is distributed version of Eclat. Advantage of this technique is its faster execution of processing. But it is not scalable. To overcome the limitation of Dist-Eclat, BigFIM technique was proposed. BigFIM is the combination of both technique Apriori and Eclat. It removes the scalability problem of Dist-Eclat but it is not as much faster as Dist-Eclat. ClustBigFIM overcomes the speed problem of BigFIM. It is also hybrid approach that is combination of parallel k-means, Apriori, and Eclat algorithm. Advantage of this technique is that it requires less time than BigFIM for execution. It is also scalable. Table shows the comparison results of various MapReduce techniques interms of speed, scalability and execution time. Both Balanced FP-Growth and ClustBigFIM technique have high speed up, high scalability and less execution time but in Balanced FP-Growth partition the search space is not efficient.

Table 1: Comparative analysis

MapReduce Technique	Speedup	Scalability	Execution Time
PAriori	Low	High	More
MRApriori	High	High	More
Parallel FP-Growth	High	High	More
Balanced FP-Growth	High	High	Less
Dist-Eclat	High	Low	Less
BigFIM	Low	High	Less
ClustBigFIM	High	High	Less

VI Conclusion

Social network generated tremendous amount of data. So frequent itemset mining on these Big data can be extremely useful. But traditional mining methods become ineffective for mining such data because of large resource criteria and excess communication cost. MapReduce programming model as a parallel programmming model has emerged for mining such Bigdata. In this paper we analyses and studied different types of MapReduce technique such as PApriori, MRApriori, Parallel FP-Growth, Balanced FP-Growth, Dist-Eclat, BigFIM, ClustBigFIM etc. From the above discussion ClustBigFIM gives better result among all of them based on faster execution and scalability.

References Références Referencias

1. Farzanyar, Zahra, and Nick Cercone(2013). Efficient mining of frequent itemsets in social network data based on MapReduce framework. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 1183-1188. ACM.
2. Le Zhou; Zhiyong Zhong; Jin Chang; Junjie Li; Huang, J.Z.; Shengzhong Feng(2010). Balanced parallel FP-Growth with MapReduce," Information Computing and Telecommunications (YCICT), 2010 IEEE Youth Conference on , vol., no., pp.243,246.
3. J Manyika, M Chui, B Brown, J Bughin, R Dobbs, C Roxburgh, AH Byers (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, 1-137.
4. R. Agrawal and R. Srikant(1994).Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of the Twentieth International Conference on Very Large Databases (VLDB), pp. 487-499.
5. S.Gole and B. Tidke (2015). Frequent Itemset Mining for Big data in Social Media using ClusBig FIM algorithm. International Conference on Pervasive Computing (ICPC).

6. D. Khanaferov, C. Lue and T. Wang (2014). Social Network Data Mining Using Natural Language Processing and Density Based Clustering", IEEE international Conference on Semantic Computing.
7. J. Dean and S. Ghemawat: Mapreduce (2004). Simplified Data Processing on Large Clusters. In: Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI), pp. 137-150.
8. Zhang, Zhigang, Genlin Ji, and Mengmeng Tang (2013). MREclat: An Algorithm for Parallel Mining Frequent Itemsets. In Advanced Cloud and Big Data (CBD), 2013 International Conference on, pp. 177-180. IEEE.
9. Yahya, Othman, Osman Hegazy, and Ehab Ezat (2012). An efficient implementation of Apriori algorithm based on Hadoop- Mapreduce model.
10. Moens, Sandy, Emin Aksehirli, and Bart Goethals (2013). Frequent itemset mining for big data. In Big Data, 2013 IEEE International Conference on, pp. 111-118.
11. http://www.adweek.com/socialtimes/heres-how- ny-people-are-on-facebook-instagram-twitter-other-big-social-networks/637205.
12. Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996). The KDD process for extracting useful knowledge from volumes of data. Commun. ACM 39, 11, 27-34.
13. https://hadoop.apache.org/docs/r1.2.1/mapred_tuto rial.html.
14. http://www.drdobbs.com/database/hadoop-the-lay-of-the-land/240150854.
15. M. J. Zaki and K. Gouda (2003). Fast vertical mining using diffsets. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, pp. 326- 335.
16. https://en.wikipedia.org/wiki/Apriori_algorithm
17. N. Li, L. Zeng, Q. He, and Z. Shi (2012). Parallel implementation of Apriori algorithm based on MapReduce. In Proc. SNPD, pages 236–241.
18. J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister (2010). A runtime for iterative MapReduce. In Proc. HPDC, pages 810–818. ACM.
19. Zhiyong Ma, Juncheng Yang, Taixia Zhang and Fan Liu (2016). An Improved Eclat Algorithm for Mining Association Rules Based on Increased Search Strategy "International Journal of Database Theory and Application Vol.9, No.5 , pp.251-266 .
20. Moens, S.; Aksehirli, E.; Goethals, B. (2013). Frequent Itemset Mining for Big Data," Big Data, 2013 IEEE International Conference on , vol., no., pp.111-118, doi: 10.1109/BigData.2013.6691742
21. L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng(2010). Balanced parallel FP-Growth with MapReduce. In Proc. YC-ICT, pages 243–246.
22. Dawen Xia, Yanhui Zhou, Zhuobo Rong, and Zili Zhang, IPFP (2013). An improved parallel FP-Growth Algorithm for Frequent Itemset Mining, isiproceedings.org.
23. Suhel Hammoud (2011). MapReduce Network Enabled Algorithms for Classification Based on Association Rules", Brunel University.
24. S.Singh,R. Garg, P.K.Mishra (2014). Review of Apriori Based Algorithms on MapReduce Framework", ICC.