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

Suman Saha[1] and Md. Syful Islam Mahfuz[2]

[1] University of Chittagong

Received: 16 December 2015 Accepted: 2 January 2016 Published: 15 January 2016

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.

Index terms— social networks, frequent itemsets mining, apriori algorithm, mapreduce framework, eclat algorithm.

1 I. Introduction

ocial 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 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].

2 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].

MapReduce Framework S 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 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.

3 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.

4 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 support-Count of I in D. We can call I is globally frequent if it satisfy the following conditions supportCount ? s × N where s is the given minimum support threshold.

5 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.

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.

6 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 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 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.

7 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.

8 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.

9 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. **??**ingh (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 (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 1itemsets 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.

10 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:

11 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. 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.

12 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.

13 BigFIM Procedure g) ClustBigFIM

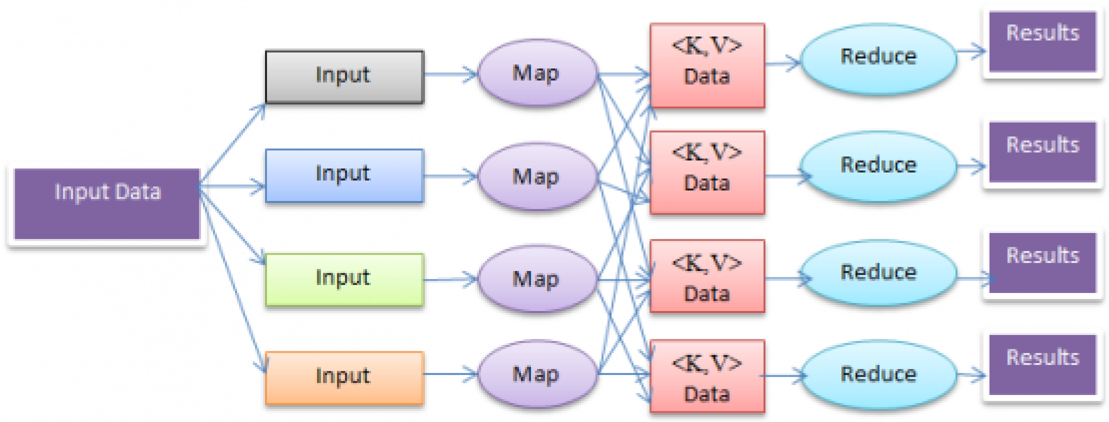
ClustBigFIM provides the hybrid approach which is the combination of parallel k-means, Apriori, and Eclat algorithm [5]. It gives an approximate result that is very much close to original result with faster speed. ClustBigFIM has the following four steps for finding frequent itemsets from large datasets.

At first clusters are generated using parallel kmeans algorithm based on Compute_Dist function and combiner function.

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.

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.

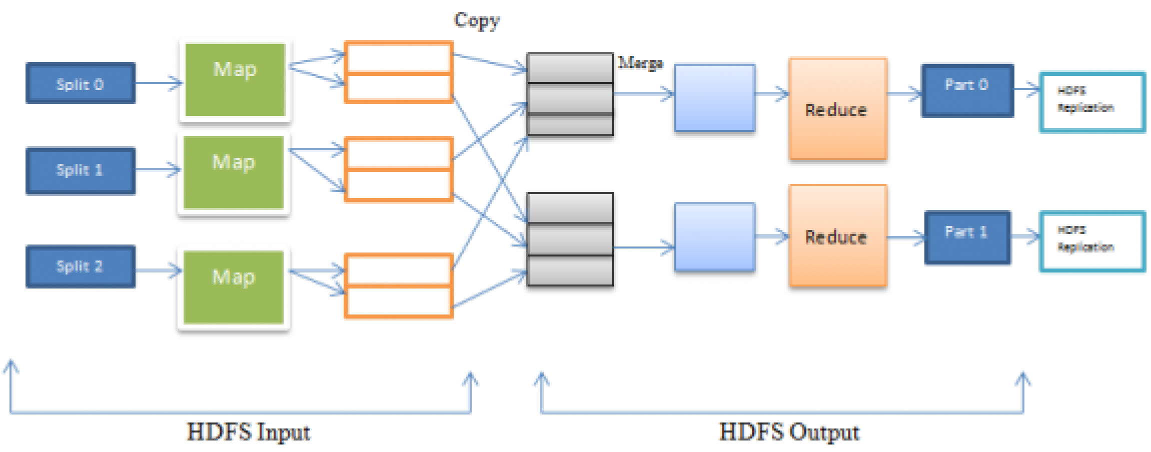
Prefix groups are assigned to the mappers which is the conditional database that fits completely into the memory. Subtrees are mined independently by [1]



1

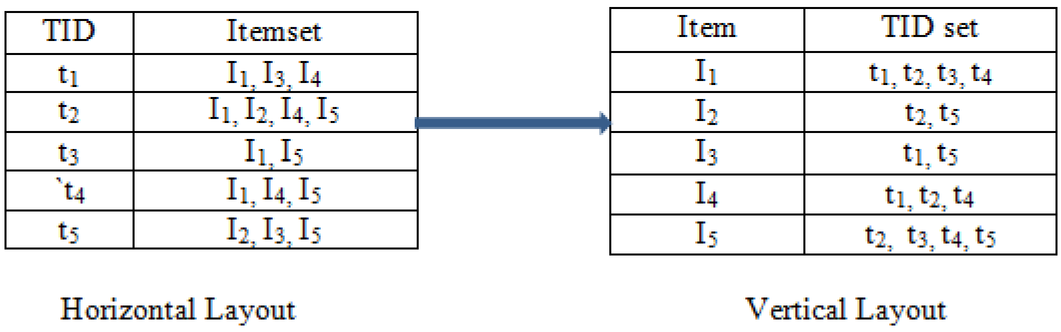
Figure 1: Figure 1 :

[1]© 2016 Global Journals Inc. (US)



2

Figure 2: Figure 2 :



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

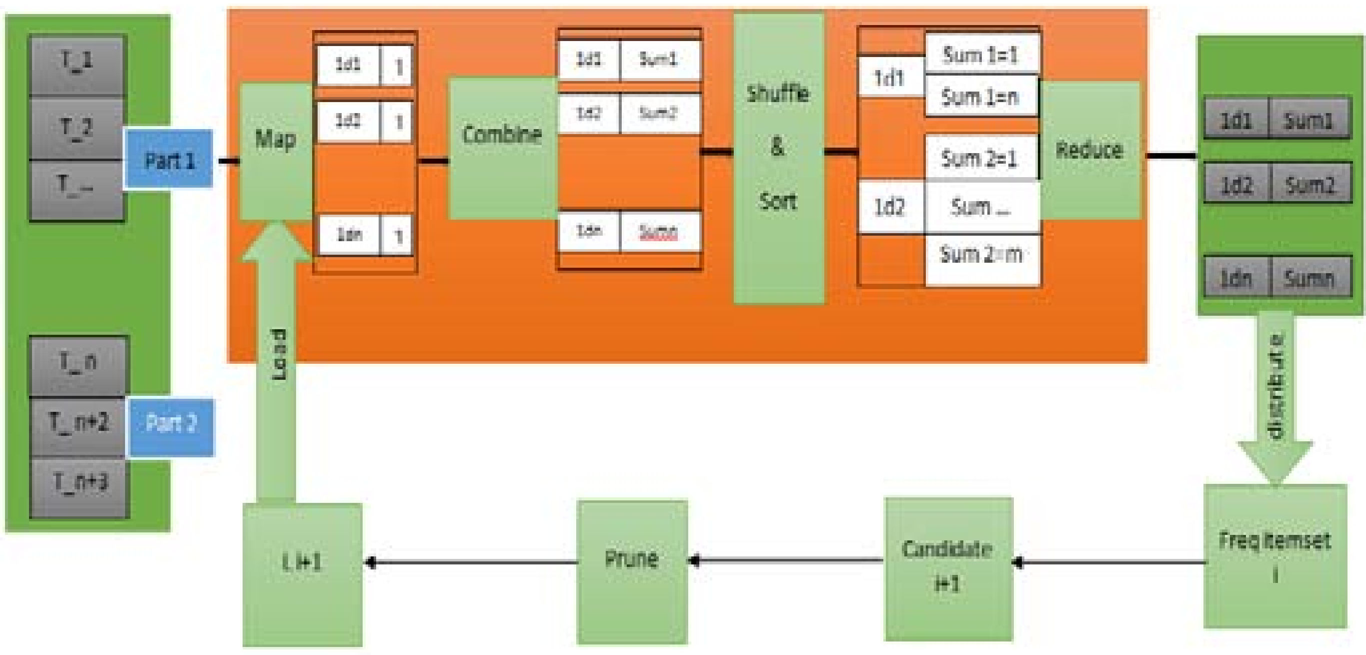
Horizontal Layout

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

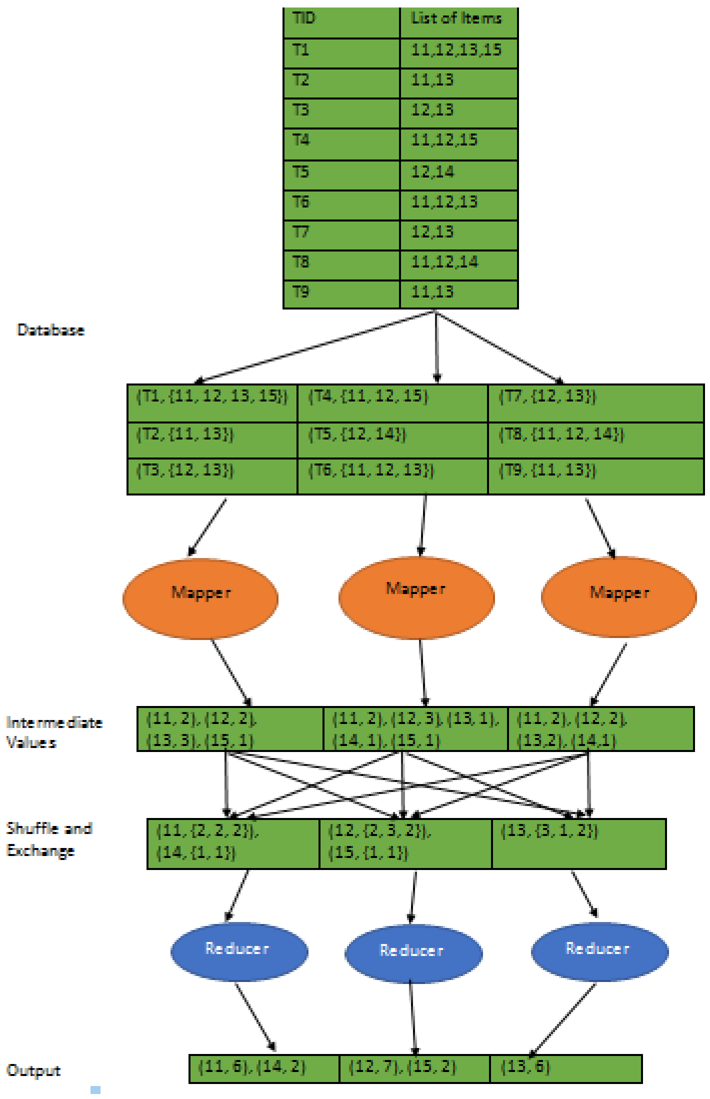
3

Figure 3: Figure 3 :



1

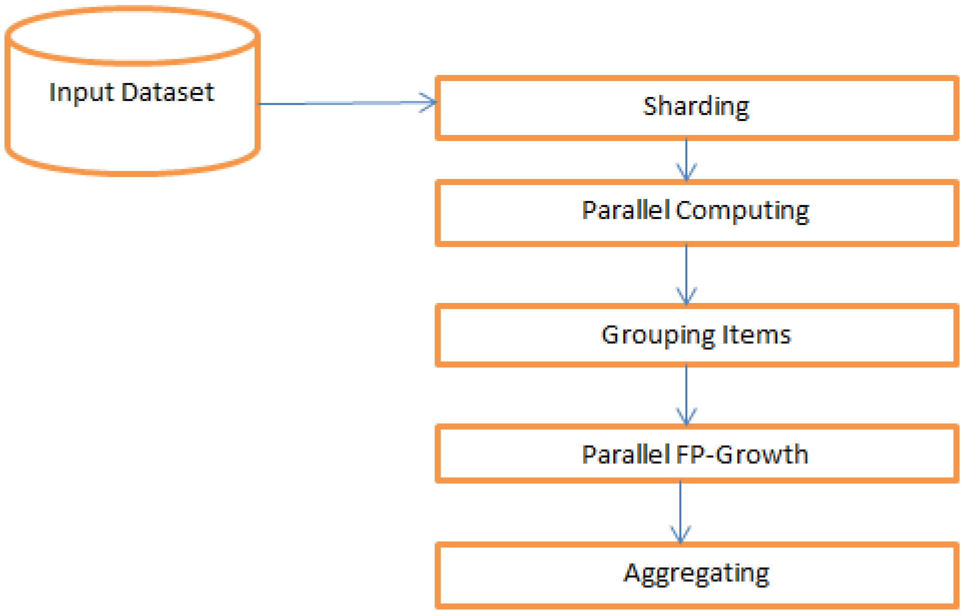
Figure 4: 1 .



TID	List of Items
T1	11,12,13,15
T2	11,13
T3	12,13
T4	11,12,15
T5	12,14
T6	11,12,13
T7	12,13
T8	11,12,14
T9	11,13

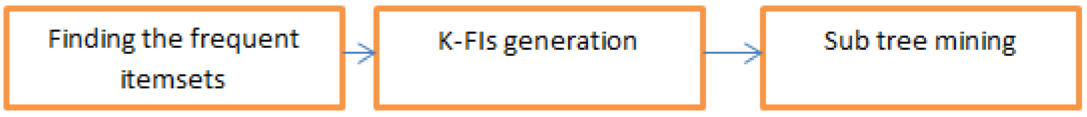
4

Figure 5: Figure 4 :



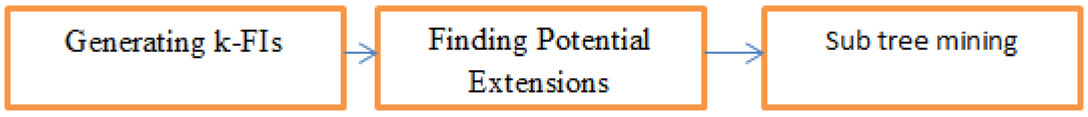
5

Figure 6: Figure 5 :



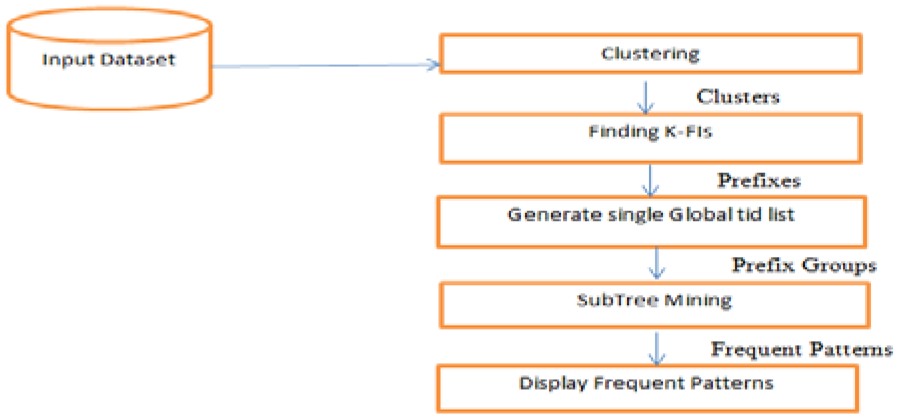
1

Figure 7: 1 B



6

Figure 8: Figure 6 :



7

Figure 9: Figure 7 :

1

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

Figure 10: Table 1 :

[Zhou] , Le Zhou .

[Zhong and Li] , Zhiyong Zhong , ; Jin Chang; Junjie Li .

[IEEE Youth Conference on] , *IEEE Youth Conference on* p. 246.

[Ekanayake et al.] , J Ekanayake , H Li , B Zhang , T Gunarathne , S.-H .

[Bae et al. ()] 'A runtime for iterative MapReduce'. J Bae , G Qiu , Fox , Twister . *Proc. HPDC*, (HPDC) 2010. ACM. p. .

[Yahya et al. ()] *An efficient implementation of Apriori algorithm based on Hadoop-Mapreduce model*, Othman Yahya , Osman Hegazy , Ehab Ezat . 2012.

[Ma et al. ()] 'An Improved Eclat Algorithm for Mining Association Rules Based on Increased Search Strategy'. Zhiyong Ma , Juncheng Yang , Taixia Zhang , Fan Liu . *International Journal of Database Theory and Application* 2016. 9 (5) p. .

[Xia et al. ()] 'An improved parallel FP-Growth Algorithm for Frequent Itemset Mining'. Dawen Xia , Yanhui Zhou , Zhuobo Rong , Zili Zhang . `isiproceedings.org` *IPFP* 2013.

[Zhou et al. ()] 'Balanced parallel FP-Growth with MapReduce'. L Zhou , Z Zhong , J Chang , J Li , J Huang , S Feng . *Proc. YC-ICT*, (YC-ICT) 2010. p. .

[Huang and Shengzhong Feng ()] 'Balanced parallel FP-Growth with MapReduce'. J Z Huang , Shengzhong Feng . *Information Computing and Telecommunications (YCICT)*, 2010. 2010.

[Manyika et al. ()] 'Big data: The next frontier for innovation, competition, and productivity'. J Manyika , B Chui , Brown , Bughin , C Dobbs , A H Roxburgh , Byers . *McKinsey Global Institute* 2011. p. .

[Farzanyar and Cercone ()] 'Efficient mining of frequent itemsets in social network data based on MapReduce framework'. Zahra Farzanyar , Nick Cercone . *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, (the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining) 2013. ACM. p. .

[Agrawal and Srikant ()] 'Fast Algorithms for Mining Association Rules in Large Databases'. R Agrawal , R Srikant . *Proceedings of the Twentieth International Conference on Very Large Databases (VLDB)*, (the Twentieth International Conference on Very Large Databases (VLDB)) 1994. p. .

[Zaki and Gouda ()] 'Fast vertical mining using diffsets'. M J Zaki , K Gouda . *Proceedings of the ninth ACM SIGK-DD international conference on Knowledge discovery and data mining*, (the ninth ACM SIGK-DD international conference on Knowledge discovery and data miningNew York, USA) 2003. p. .

[Moens et al. ()] 'Frequent itemset mining for big data'. Sandy Moens , Emin Aksehirli , Bart Goethals . *Big Data, 2013 IEEE International Conference on*, 2013. p. .

[Moens et al. ()] 'Frequent Itemset Mining for Big Data'. S Moens , E Aksehirli , B Goethals . 10.1109/Big-Data.2013.6691742. *IEEE International Conference on*, 2013. 2013. p. . (Big Data)

[Gole and Tidke ()] S Gole , B Tidke . *Frequent Itemset Mining for Big data in Social Media using ClusBig FIM algorithm. International Conference on Pervasive Computing (ICPC)*, 2015.

[Hammoud ()] *MapReduce Network Enabled Algorithms for Classification Based on Association Rules*, Suhel Hammoud . 2011. Brunel University

[Zhang et al. ()] 'MREclat: An Algorithm for Parallel Mining Frequent Itemsets'. Zhigang Zhang , Genlin Ji , Mengmeng Tang . *Advanced Cloud and Big Data (CBD), 2013 International Conference on*, 2013. IEEE. p. .

[Li et al. ()] 'Parallel implementation of Apriori algorithm based on Map-Reduce'. N Li , L Zeng , Q He , Z Shi . *Proc. SNPD*, (SNPD) 2012. p. .

[Singh et al. ()] *Review of Apriori Based Algorithms on MapReduce Framework*, S Singh , R Garg , P K Mishra . 2014. (ICC)

[Dean and Ghemawat: Mapreduce ()] 'Simplified Data Processing on Large Clusters'. J Dean , S Ghemawat: Mapreduce . *Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI)*, (the Sixth Symposium on Operating System Design and Implementation (OSDI)) 2004. p. .

[Khanaferov et al. ()] 'Social Network Data Mining Using Natural Language Processing and Density Based Clustering'. D Khanaferov , C Lue , T Wang . *IEEE international Conference on Semantic Computing*, 2014.

[Fayyad et al. ()] 'The KDD process for extracting useful knowledge from volumes of data'. Usama Fayyad , Gregory Piatetsky-Shapiro , Padhraic Smyth . *Commun. ACM* 1996. 39 p. .