Frequent Itemset Mining for Big Data in social media using ClustBigFIM algorithm

Sheela Gole
Department of Computer Engg
Flora Institute of Technology, Pune
Maharashtra, India
sheelagole1288@gmail.com

Bharat Tidke
Department of Computer Engg
Flora Institute of Technology, Pune
Maharashtra, India
batidke@gmail.com

***Abstract* - Tremendous amount of data is getting explored through IOT (Internet of Things) from variety of sources such as sensor network, social media feed, internet applications, called as Big Data. Big Data cannot be handled by conventional tools and techniques. Social networks are becoming dominant in communications over internet. The Big Data mining is essential in order to extract value from massive amount of data which could give better insights using efficient techniques. Association Rule mining and frequent itemset mining are popular techniques for data mining which needs entire dataset into main memory for processing, but large datasets do not fit into main memory. To overcome this limitation MapReduce is used for parallel processing of Big Data having features such as high scalability and robustness which helps to handle problem of large datasets. Proposed novel method, ClustBigFIM works on MapReduce framework for mining large datasets; ClustBigFIM is modified BigFIM algorithm providing scalability and speed in order to extract meaningful information from large datasets in the form of associations, emerging patterns, sequential patterns, correlations and other significant data mining tasks which gives better insight to make effective business decisions in competitive environment using faster and efficient parallel processing platform.**

Keywords— Association Rule Mining, Big Data, Clustering, Frequent Itemset Mining, Hadoop, MapReduce.

I. INTRODUCTION

Social networking is used widely as basic communication media with exponential growth. Sites such as Twitter, Facebook, Linked In and My Space are frequently used by people. Facebook had more than 845 million active users in February 2012 [1]. Research is done on Big Data which shows that it can create significant value in the word's economy improving productivity of companies and public sector [2]. Storing huge amount of data won't have any value without KDD (Knowledge discovery in Database) which is process of finding information from database and extracted knowledge can be used for making effective business decisions [3].

Discovery of association rules from large database is one of the problems in KDD (Knowledge Discovery in Database). Size and complexity of Big Data are challenges for discovering association rules and frequent itemset mining. "Market – Basket" model is based on relationships between elements [4]. Association rule mining and frequent itemset mining is popular techniques of data mining. It reveals frequency of items purchased together. The whole database scan is necessary in FIM, it might create challenge when datasets size is scaling, as large datasets does not fit into memory. Several approaches exist for association rule mining [5], [6], [7]. Frequent itemsets play an essential role in finding correlations, clusters, episodes and many other data mining tasks [8].Value discovered from frequent itemsets can be used to make decisions in marketing.

For massive datasets, mining association rules on single computer is not efficient as it is limited by the processor capacity, RAM, storage and other such factors [9]. Three parallel algorithms - Count Distribution, Data Distribution and Candidate Distribution has been presented by Agrawal and Shafer [10], minimizing the communication (Count Distribution), utilizing the aggregate main memory of system(Data Distribution) and reducing synchronization and incorporating load balancing (Candidate Distribution). New parallel association mining algorithms has been proposed by Zaki et al. [11]. Parallel programming has shared memory architecture and distributed data processing architecture. Shared memory architecture can be used by adapting frequent itemset mining algorithms but it does not provide enough scalability and distributed data processing architectures uses message passing as the communication channel in which low level languages are used for programming but higher level languages are more popular in business [12]. To handle incremental data efficiently parallel approach is proposed by Bhadane et al. [13].

Chen et al. [14] has proposed FAST (Finding Association Rules from Sampled Transactions) which is a Two-Phase sampling based algorithm that reduces errors due to sampling fluctuations. Instead of scanning entire database for finding association rules only sampled transactions are scanned in FAST.

MapReduce framework has been proposed by Google[15] is used for parallel processing of large datasets and it works on key-value pairs. Frequent itemset mining need to calculate support and confidence which can be done in parallel using MapReduce programming model. Faster processing can be achieved by calculating frequency of items using map functions which executes in parallel on set of hadoop clusters and reduce functions used to combine the local frequent items and give global frequent items.

978-1-4799-6272-3/15/$31.00(c)2015 IEEE

In this paper, proposed hybrid method ClustBigFIM is a modified BigFIM [16] algorithm for generating frequent itemsets which uses parallel k-means [17] and Apriori [18] for generating k-FIs and Eclat [11] for finding potential extensions.

The paper is organized as follows; Section II describes the basic notions related to frequent itemset mining. Section III gives the survey of different frequent itemset mining algorithms. Section IV gives overview basic MapReduce programming model and Section V explains proposed model. Section 6 gives conclusion of the paper including future work.

II. PRELIMINARIES

A. Problem Description

Let I be a set of items, $I = \{i_1,i_2,i_3,\ldots,i_n\}$, X is a set of items, $X = \{i_1,i_2,i_3,\ldots,i_k\} \subseteq I$ called k - itemset if it has k items. A transaction T contains set of items, denoted as T = (tid, I) where tid is transaction ID and I is an itemset. $T \in D$, where D is a transaction database. The cover of itemset X in D is the set of transaction IDs which contains items from X.

$$Cover(X, D) = \{tid \mid (tid, I) \in D, X \subseteq I\}$$

The support of an itemset X in D is number of transactions which contains items from X.

$$support(X, D) = |Cover(X, D)|$$

Frequency of an itemset X is depends on number of occurrences of X available in transaction T of database D. Probability is used for calculating the frequency of itemset.

$$Frequency(X, D) = P(X) = \frac{(support(X, D))}{(|D|)}$$

An itemset is said to be frequent when its absolute minimum support threshold σ_{abs}, with $0 \leq \sigma_{abs} \leq |D|$. When frequencies are considered instead of support, relative minimal frequency support is used.
Let X, $Y \subseteq I$ are two itemsets then monotonic property of support is,

$$X \subseteq Y \rightarrow support(Y) \leq support(X)$$

This is a downward closure property, used for pruning the itemsets which are infrequent in previous set of frequent itemset..

Items are represented in a fixed order in itemset. If itemsets have some first k elements same, then those are called common k-prefix of itemsets. The path from root to node in the prefix tree represents itemsets and same prefix is shared by sibling nodes. Projected database of an item i_1 is set of transactions in D that contains item i_1.

Apriori [18] uses a breadth – first search approach; frequency of an itemset is counted by scanning a database D which generates frequent k-itemsets and then candidate k+1 itemsets are generated from frequent k-itemsets by applying support and threshold conditions. Monotonic property is exploited to prune the candidates which are infrequent. The process of joining and pruning iterates till the frequent itemset or candidate set is NULL.

Eclat [11] uses a depth – first search for traversing the prefix tree to find frequent itemsets. It uses vertical database representation which is used to achieve a good performance. 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. Eclat requires vertical database D' needs to be stored in main memory. dEclat [19] developed by Zaki and Gouda store diffset; diffset is difference between candidate itemset of size k and prefix frequent itemsets of size k-1; support value is based on diffset gaining performance growth than Eclat; it is not efficient when the database is sparse.

Malek and Kadima [20] proposed a novel approach for discovering frequent itemsets from set of clusters of dataset using MapReduce which increases the performance. In map phase distance from centers are computed for each itemset and assigned to related cluster. In reduce phase partial sum of distances is collected and new itemsets list is computed.

B. Data Layout

The Horizontal and vertical database layouts are used in association rule mining, Apriori uses horizontal database layout whereas Eclat uses vertical database layout.

Horizontal Layout D and Vertical Layout D' can be represented as,

$$D = \{t_1,t_2,\ldots,t_l\} \text{ and } D' = \{(i_j, Cover(i_j, D))|i_j \in X\}$$

Consider the itemset X= {a, b, c, d, e} and D = {t_1, t_2, t_3, t_4, t_5, t_6}

TABLE I. HORIZONTAL LAYOUT

tid	Itemset
1	b, a, e, c
2	b, c
3	a, d
4	c, a, e
5	b, a, c
6	b, d

TABLE II. VERTICAL LAYOUT

Item	Tidset
a	1, 3, 4, 5
b	1, 2, 5, 6
c	1, 2, 4, 5
d	3
e	1, 4

C. Lexicographic Prefix Tree

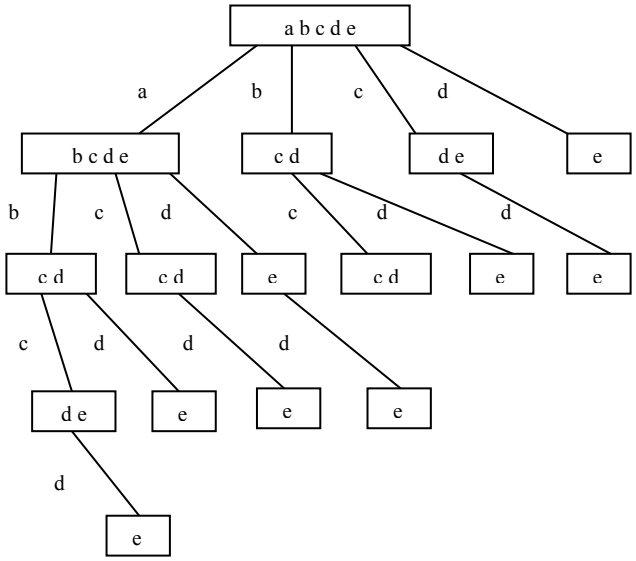

Fig. 1. Lexicographic tree illustration [21].

The Agrawal et al. [21] proposed tree projection algorithm which can be used to construct the lexicographic prefix tree for frequent itemset mining using breadth first search and depth first search approach; the proposed algorithm uses Eclat algorithm for finding potential extensions to k-FIs. Fig.1. shows lexicographic tree for data layouts given in above tables having frequent itemsets stored in node of a tree along with support of these itemsets. The root contains all frequent itemsets. Every edge in the tree is labeled with an item. Itemsets in any node are stored as singleton sets like $\{i_2\}$, $\{i_3\}$, ...,$\{i_k\}$ and actual itemsets contain itemsets on edges from root to current node. Let node $\{i_1, i_2, i_3, i_4, i_5\}$ of the tree in Fig.1. Node will have four doubletons, $\{i_1, i_2\}$; $\{i_1, i_3\}$; $\{i_1, i_4\}$;$\{i_1, i_5\}$. Lexicographic order is used for storing singleton items. Consider root has $\{i_1\}$, $\{i_2\}$, ...,$\{i_k\}$ then the nodes in level2 will contain $\{i_2\}$, $\{i_3\}$, ...,$\{i_k\}$;$\{ i_3\}$, $\{i_4\}$, …,$\{i_k\}$;$\{i_k\}$ and so on. For each candidate set Cover(X, D) is also stored. It uses the search space instead of data space. Path of tree will be stored at any time; Eclat can be started at any depth k of the prefix tree using all frequent k-FIs as seeds. Prefix trees are formed in such a way that siblings are sorted by their individual frequency in ascending order. Formally, $X = \{i_1, i_2, ...,i_k\}$, where support(i_a) ≤ support(i_b) ↔ a < b . This rule can be used for pruning the prefix tree at lower depths providing shorter run times.

III. RELATED WORK

Data mining literature already has sequential and parallel algorithms [10], [22], [23], [24], [25], [26], [27], [28] for finding frequent itemsets although there is need of parallel algorithms which can work on widely used distributed platform, MapReduce because there exist various issues of scalability and performance for existing parallel algorithms in the era of Big Data.

This section gives overview of the frequent itemset mining on MapReduce. Modification of Apriori on MapReduce has been proposed by Lin et al. [29]; Single Pass Counting(SPC), Fixed Passes Combined – Counting(FPC) and Dynamic Passes Counting (DPC) which do counting step parallel by distributing dataset to mappers. Li et al. [30] has proposed parallel version of Apriori based algorithm on MapReduce. Apriori based algorithm does not work on large datasets having long frequent itemsets. Hammoud has proposed MRApriori [31] approach for finding frequent itemsets by switching between vertical and horizontal layout iteratively which eliminates need of iterative scanning of data. It repeats scanning for other intermediate data which reduces with iteration.

MREclat [32] is a modified Eclat algorithm on MapReduce framework which generates a list of frequent itemsets, the list is partitioned into equivalence classes, then for each equivalence class frequent itemsets are computed using MapReduce framework.

Parallel FP-Growth (PFP) [33] has mined tag itemsets from which web page itemsets are generated, requiring two scans on database. By using MapReduce framework and its fault tolerant mechanism, task of mining massive scale data is converted into other small tasks which are not dependent on each other. Scalability and linear speedup over shared memory parallel FP growth algorithms is achieved with the exploitation of MapReduce framework. Grouping strategy of PFP has problems with memory and speed; to balance the groups of PFP Zhou et al. [34] has proposed algorithm for faster execution using single items which is also not an efficient way. Xia et al. [35] has been proposed Improved PFP algorithm for mining frequent itemsets from massive small files datasets using small files processing strategy.

Moens et al. [16] proposed two methods for frequent itemset mining for Big Data on MapReduce, First method Dist-Eclat is distributed version of pure Eclat method which optimizes speed by distributing the search space evenly among mappers, second method BigFIM uses both Apriori based method and Eclat with projected databases that fit in memory for extracting frequent itemsets.

Riondato et al. [36] has been proposed PARMA algorithm which finds collections of frequent itemsets in short time using sampling method. Mined frequent itemsets are approximate. It finds the sampling using k-means clustering algorithm. The sample list is called as clusters.

Malek and Kadima [20] proposed a novel approach which uses clustering technique for mining of frequent itemsets using MapReduce framework with increase in performance.

TWISTER [37] improves performance by iterative approach. NIMBLE [38] provide better programming tools for data mining. YAFIM [39] has proposed parallel frequent itemset mining algorithms on Spark RDD framework.

Table III gives comparative analysis of different frequent itemset mining techniques starting from basic frequent itemset mining algorithms to the frequent itemset mining algorithms which works on MapReduce framework.

TABLE III. COMPARATIVE ANALYSIS OF DIFFERENT FIM TECHNIQUES .

Author's Name	Technique	Works on MapReduce	Characteristics	Benefits	Limitations
Agrawal et al. [18]	Apriori	No	Level wise search, Monotonicity property and Easy to implement	Generates frequent itemsets and association rules	Scalability
Zaki et al. [11]	Eclat	No	Depth First Search, Works on vertical database and intersection of tid_list	Enhances locality and requires few scans to database	Degraded performance with larger number of transactions
Zaki et al. [19]	dEclat	No	Uses vertical database and diffsets over tidset	Significant performance improvements	For sparse database diffsets loses its advantage over tidset
Han et al. [26]	FP-Growth	No	Recursive approach, Employs FP-tree data structure	Focused search of smaller databases	Poor Performance
Lin et al. [8]	SPC, FPC, DPC	No	SPC- simple implementation of Apriori on MapReduce framework, FPC – Single MapReduce phase with merging of fixed passes and DPC - Dynamically combines passes	FPC and DPC provide efficient implementation of Apriori on MapReduce framework and reduce scheduling, invocation, increasing node utilization, workload balancing.	SPC increases scheduling and waiting overhead and FPC may get overloaded in case of large number of candidates
Li et al. [30]	PApriori	Yes	Sizeup, speedup and scaleup are used for performance evaluation	Efficient and Good performance for large database	User needs to give number of reducers
Hammoud [31]	MRApriori	Yes	Single scan of data in original format and Hybrid data structure, both horizontal and vertical	Efficient for large database and Good performance	No significant reduction in processing time
Li et al. [33]	Parallel FP Growth	Yes	Parallel version - FP-Growth, Independent mining of FP-tree and Grouping of items	Linear scalability	Not efficient in terms of memory and speed
Zhou et al. [34]	Balanced FP - Growth	Yes	Improvement in FP-Growth and Uses frequencies of frequent items to balance the groups of PFP	Faster execution using singletons with balanced distribution	Search space partition using single items is not most efficient way
Riondato et al. [36]	PARMA	Yes	Uses random sampling method	Minimizes data replication, Scaling linearly, Runs faster	Finds approximate collection of frequent itemsets
Moens et al. [16]	Dist-Eclat	Yes	Distributed version of Eclat	Speed	Scalability
Moens et al. [16]	BigFIM	Yes	Hybrid method (Apriori + Eclat)	Scalability	Speed

IV. RESEARCH METHODOLOGY FOR BIG DATA PROBLEM

Apache hadoop project exploits MapReduce framework and Hadoop Distributed File System to handle Big Data by providing framework for storing, processing and analyzing Big Data. This section gives overview of MapReduce framework and Hadoop Distributed File System.

A. MapReduce Framework

Google [15] proposed MapReduce framework which is a processing and execution model for distributed environment that runs on large clusters of commodity hardware [9], [40].

MapReduce framework has two phases, Map phase and Reduce phase. Map and reduce functions are used for large parallel computations specified by users. Map function takes chunk of data from HDFS in (key, value) pair format and generates a set of (key', value') intermediate (key, value) pairs. MapReduce framework collects all intermediate values which are bind to same intermediate key and same are passed to reduce function; it is formalized as,

map :: (key, value) → (key', value'); Value of map function is used by reduce function.

Intermediate key details are received by reduce function, that are merged together. The intermediate values are provided to reduce function through iterator, by using which too large values fit in memory, formalized as,

reduce :: (key', list (value')) → (key'', value'')

Output can have one or more output files which are written on HDFS. Examples such as Inverted Index, Term Vector per host Distributed Sort, Distributed Grep, count of URL access frequency can be completed through MapReduce framework.

B. HDFS

The HDFS is used for storing the Big Data using blocks. MapReduce framework can read the data in the form of splits. HDFS can handle various faults in the system and it can store massive datasets, by providing scalability and robustness. Failure is handled without loss of data. If any machine fails, the hadoop cluster can continue to work without disturbing the current work by re-assigning remaining work to other available clusters. The files are stored as blocks and blocks are replicated on various node which helps in availability of data [9], [40]. Replicated data also helps in fault tolerance. HDFS does not need any extra hardware than regular hardware and for data processing it exploits Map Reduce framework which is efficient.

V. PROPOSED MODEL

Many large enterprises and organizations are developing own solutions to Big Data problems. Proposed model ClustBigFIM works on Big Data and it overcomes the problem of speed of original BigFIM algorithm which uses hybrid approach for generating frequent patterns.

A. ClustBigFIM

Proposed method ClustBigFIM provides hybrid approach for frequent itemset mining for large data sets using combination of parallel k-means, Apriori algorithm and Eclat algorithm; by increasing scalability and performance which overcomes limitation of Big FIM. It gives the approximate results that are closer to the original results but with faster speed. Big FIM [16] overcomes the problems of Dist-Eclat [16] such as, mining of sub-trees requires entire database into main memory and entire dataset needs to be communicated to most of the mappers. BigFIM is a hybrid approach which uses Apriori algorithm for generating k-FIs, and then Eclat algorithm is applied to find frequent item sets. Candidate itemsets do not fit into memory for greater depths is the limitation of using Apriori for generating k-FIs in BigFIM [16] algorithm and speed is slow for BigFIM.

Fig. 2 shows basic block diagram of proposed model,

Proposed algorithm ClustBigFIM has below four steps which need to be applied on large datasets, steps are Find Clusters, Finding k-FIs, Generate Single Global TID list, Mining of Subtree.

1) Find Clusters

Proposed algorithm uses parallel k-means algorithm for generating clusters using Compute_Dist function and combiner function which takes centers as input; Generated clusters are mined using Apriori algorithm in next step.

2) Finding k-FIs

Frequent itemsets mining is done from clusters using Apriori algorithm. Local supports are searched using mappers and then global support is calculated by reducers. Apriori is used up to certain length to find frequent k-length prefixes, $P_k = \{p_1, p_2, \ldots, p_l\}$ like $P_4 = \{p_1, p_2, p_3, p_4 p_5, p_6\}$, Let $X, Y \subseteq I$ be two itemsets then

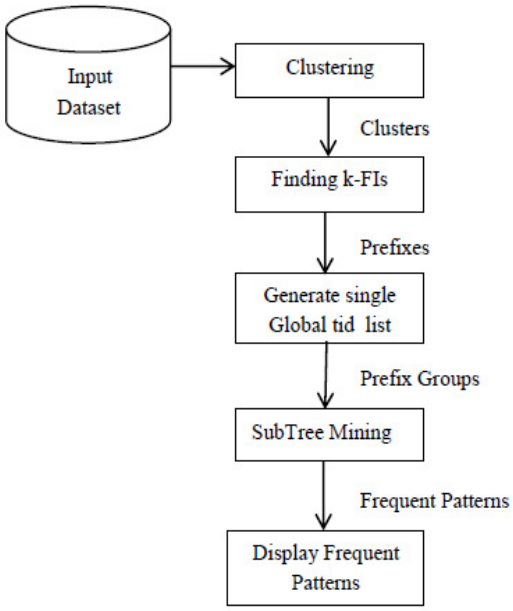

Fig. 2. Block Diagram of Proposed Model

monotonic property of support is, $X \subseteq Y \rightarrow support(Y) \leq support(X)$. This property is used while pruning the itemsets from candidate list in order to obtain next frequent itemset list.

3) Generating Single Global TID list

From computed prefixes in above step, prefix tree is built; tid_lists for (k+1) FIs are obtained which can be done similar to word counting. However supports and reducers compute single global tid_lists. Prefix tree are formed in a way that siblings are sorted by their individual frequency in ascending order. Formally, $X = \{i_1, i_2, \ldots, i_k\}$, where $support(i_a) \leq support(i_b) \leftrightarrow a < b$ which can be used for pruning. mapper computes local –tid_lists instead to local support.

4) Mining of Subtree

Prefix groups are distributed across mappers who defines projected database that completely fits into memory; using diffset projected database is mined with depth-first search. Subtrees of prefix trees are mined separately and uses longer frequent itemsets as prefixes for a better load balancing and Round Robin(P_i is assigned to the worker $P_k^{(i \bmod n)}$) used for assigning the load. P_k is partitioned into n groups ($P_k^1, P_k^2, \ldots, P_k^n$), where n is number of mappers. For P_4 in above step, consider there are 3 mappers then $\{p_1, p_2\}$ will be given to 1st mapper, $\{p_3, p_4\}$ to 2nd mapper and $\{p_5, p_6\}$. Each mapper mine the Subtree independently by working on conditional database which completely fits into main memory. Frequent itemsets mined by mapper's encoded using compressed trie and communicated to reducers for reducing the network traffic.

VI. CONCLUSION

The proposed algorithm has hybrid method for finding frequent item sets using parallel k-means, Apriori and Eclat algorithm on MapReduce framework. Parallel k-means can give approximate results but in short time; Apriori finds frequent itemsets having size k; Eclat algorithm finds potential extensions to frequent item sets and subtree mining by resolving memory problem. MapReduce platform can be used extensively for mining Big Data from social media as tradition tool and techniques cannot handle Big Data. Planning to apply frequent item set mining algorithm and MapReduce framework on stream of data which can be real time insights in Big Data.

REFERENCES

[1] Zahra Farzanyar 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 (ASONAM '13).

[2] J Manyika, M Chui, B Brown, J Bughin, R Dobbs, C Roxburgh, AH Byers. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, 1-137, 2011.

[3] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. 1996. The KDD process for extracting useful knowledge from volumes of data. Commun. ACM 39, 11 (November 1996), 27-34.

[4] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. SIGMOD Rec. 22, 2 (June 1993), 207-216. DOI=10.1145/170036.170072

[5] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. 2000. Algorithms for association rule mining — a general survey and comparison. SIGKDD Explor. Newsl. 2, 1 (June 2000), 58-64.

[6] Woo Sik Seol; Hwi Woon Jeong; Byungjun Lee; Hee Yong Youn, "Reduction of Association Rules for Big Data Sets in Socially-Aware Computing," Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on , vol., no., pp.949,956, 3-5 Dec. 2013.

[7] Jiawei Han. 2005. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[8] B. Goethals. "Survey on frequent pattern mining". Manuscript, 2003.

[9] T. White, Hadoop: The Definitive Guide. O'Reilly Media, Yahoo! Press, June 5, 2009.

[10] R. Agrawal and J. C. Shafer. Parallel mining of association rules. Ieee Trans. On Knowledge And Data Engineering, 8:962–969, 1996.

[11] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithms for discovery of association rules. Data Min. and Knowl. Disc., pages 343–373, 1997.

[12] G. A. Andrews. Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley, 2000.

[13] Chetashri Bhadane, Ketan Shah and Prajakta Vispute. An efficient parallel approach for frequent itemset mining of incremental data. International Journal of Scientific & Engineering Research, Vol. 3, Issue 2, February - 2012.

[14] Bin Chen, Peter Haas, and Peter Scheuermann. 2002. A new two-phase sampling based algorithm for discovering association rules. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02).

[15] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In Proc. OSDI. USENIX Association, 2004.

[16] Moens, S.; Aksehirli, E.; Goethals, B., "Frequent Itemset Mining for Big Data," Big Data, 2013 IEEE International Conference on , vol., no., pp.111,118, 6-9 Oct. 2013 doi: 10.1109/BigData.2013.6691742

[17] Weizhong Zhao, Huifang Ma, and Qing He. 2009. Parallel K-Means Clustering Based on MapReduce. In Proceedings of the 1st International Conference on Cloud Computing (CloudCom '09), Springer-Verlag, Berlin, Heidelberg, 674-679.

[18] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Proc. VLDB, pages 487–499, 1994.

[19] M. J. Zaki and K. Gouda. Fast vertical mining using diffsets. In Proc. ACM SIGKDD, pages 326–335, 2003.

[20] M. Malek and H. Kadima. Searching frequent itemsets by clustering data: towards a parallel approach using mapreduce. In Proc. WISE 2011 and 2012 Workshops, pages 251–258. Springer Berlin Heidelberg, 2013.

[21] R. Agrawal, C. Aggarwal, and V. Prasad, "A Tree Projection Algorithm for Generation of Frequent Item Sets," Parallel and Distributed Computing, pp. 350-371, 2000.

[22] J. Li, Y. Liu, W.-k. Liao, and A. Choudhary. Parallel data mining algorithms for association rules and clustering. In Intl. Conf. on Management of Data, 2008.

[23] E. Ozkural, B. Ucar, and C. Aykanat. Parallel frequent item set mining with selective item replication. IEEE Trans. Parallel Distrib. Syst., pages 1632–1640, 2011.

[24] M. J. Zaki. Parallel and distributed association mining: A survey. IEEE Concurrency, pages 14–25, 1999.

[25] L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, and P. Luo. Distributed data mining: a survey. Information Technology and Management, pages 403–409, 2012.

[26] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD Rec., pages 1–12, 2000.

[27] L. Liu, E. Li, Y. Zhang, and Z. Tang. Optimization of frequent itemset mining on multiple-core processor. In Proceedings of the 33rd international conference on Very large data bases, VLDB '07, pages 1275–1285. VLDB Endowment, 2007.

[28] Mingjun Song; Rajasekaran, S., "A transaction mapping algorithm for frequent itemsets mining," Knowledge and Data Engineering, IEEE Transactions on , vol.18, no.4, pp.472,481, April 2006

[29] M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh. Apriori-based frequent itemset mining algorithms on MapReduce. In Proc. ICUIMC, pages 26–30. ACM, 2012.

[30] N. Li, L. Zeng, Q. He, and Z. Shi. Parallel implementation of Apriori algorithm based on MapReduce. In Proc. SNPD, pages 236–241, 2012.

[31] S. Hammoud. MapReduce Network Enabled Algorithms for Classification Based on Association Rules. Thesis, 2011.

[32] Zhigang Zhang; Genlin Ji; Mengmeng Tang, "MREclat: An Algorithm for Parallel Mining Frequent Itemsets," Advanced Cloud and Big Data (CBD), 2013 International Conference on , vol., no., pp.177,180, 13-15 Dec. 2013 doi: 10.1109/CBD.2013.22

[33] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang. PFP: Parallel FP-Growth for query recommendation. In Proceedings of the 2008 ACM conference on Recommender systems, RecSys '08, pages 107–114, New York, NY, USA, 2008. ACM.

[34] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng. Balanced parallel FP-Growth with MapReduce. In Proc. YC-ICT, pages 243–246, 2010.

[35] Dawen Xia, Yanhui Zhou, Zhuobo Rong, and Zili Zhang, IPFP : an improved parallel FP-Growth Algorithm for Frequent Itemset Mining, isiproceedings.org, 2013.

[36] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal. PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce. In Proc. CIKM, pages 85–94. ACM, 2012.

[37] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: A runtime for iterative MapReduce. In Proc. HPDC, pages 810–818. ACM, 2010.

[38] A. Ghoting, P. Kambadur, E. Pednault, and R. Kannan. NIMBLE: a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In Proc. ACM SIGKDD, pages 334–342. ACM, 2011.

[39] Hongjian Qiu, Rong Gu, Chunfeng Yuan, and Yihua Huang. 2014. YAFIM: A Parallel Frequent Itemset Mining Algorithm with Spark. In Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW '14).

[40] Dittrich, Jens, and Jorge-Arnulfo Quiané-Ruiz. "Efficient big data processing in Hadoop MapReduce." Proceedings of the VLDB Endowment 5.12 (2012): 2014-2015