International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 9 • Number 46 • 2016

A Comprehensive Study of Map Reduce Frequent Itemset Mining: A Survey

Palvi Rani[a] and Ishan Ranjan[b]

[a,b]Dept. of Computer Science and Engineering, Sharda University, Greater Noida-201308, India. Email: [b]ishan.ranjan@sharda.ac.in

Abstract: Frequent Itemset Mining is the most popular and interesting technique to extract knowledge from commercial applications. Rapid increase in data has made a challenge to find legitimate information from massive amount of data. However, traditional frequent itemset mining techniques are not adequate to handle large dataset. MapReduce framework provides parallel computation approach for storing and analysing large dataset on commodity clusters in cloud computing. A number of FIM techniques based on MapReduce have been developed in recent researches. A critical review of these techniques required to develop an efficient FIM algorithm. In this paper, we have described various features and characteristics of FIM. We have also described the efforts and techniques to mine Frequent Itemsets on MapReduce in Big data. Further, the challenging issues and some open problems in the area have been discussed.

Keywords: Frequent Itemset Mining, Big Data, Association Rule Mining, Cloud Computing, MapReduce.

1. INTRODUCTION

Frequent Itemset Mining used for generating Association rules in the process of knowledge discovery. In other words, it's a process to extract knowledge from frequently occurring events. In 1993, Agrawal et. al. [1] has introduced the concept of Association rule mining for finding the relationship among different data itemsets in the given records. In 1994, they have developed a classic algorithm named Apriori Algorithm [2]. FIM is the necessary effort in the process of developing association rules. "Market Basket Analysis" accepted as modelling technique of Association Rule Mining. Along with market studies ARM also found in areas like Decision Support, GPS, social networking, Web Search Engines, telecommunication, alarm prediction etc. Tremendous amount of literature is available for progress and enhancements in the techniques.

From last few decades, development in information technology produces abundant amount of data. Social Networking, blogs, e-commerce applications etc. are major producers of data over the internet. Organizations continually absorb the data for better decision making. According to estimation by IBM, everyday almost 2500 trillion KB of data created and maximum data available in the world generated in just two years [3]. Cloud Computing's Mass storage and distributed computing architecture proficient to process the massive amount of data efficiently and effectively. Now, Massive amount of data over the internet introduces a term Big Data. Volume, velocity and variety are three major characteristic of Big Data. According to Gartner,

> *"Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."* *("Gartner IT Glossary, n.d.")*

MapReduce Architecture provides a sophisticated programming model for Cloud Computing framework. Now days, MapReduce introduced as a very dominant architecture in distributed computing with a simplified solution to handle large data set. MapReduce framework was first introduced by Google [4] in 2004 and Hadoop is the implementation of MapReduce framework simplifies distributed programming. Hadoop is reliable and scalable open source software for distributed computing. Parallel data mining techniques using MapReduce are popular since 2005. To improve the performance issues of MapReduce researchers introduced different frameworks. NIMBLE [5] designed a better parallel programming paradigm for data mining. TWISTER [6] has improved the performance of MapReduce by reducing the cycles. These frameworks are not widely used due to the unavailability of these frameworks.

This paper presents the performance comparative study of MapReduce based Frequent Itemset Algorithms. Sections are organized in following way: Section 1 detailed introduction, Section 2 describes Related Work. In Section 3, parallel and distributed algorithms have been discussed. In section 4, we have discussed major challenging issues and open problems.

2. RELATED WORK

2.1. Problem Definition

Given $I = \{I_1, I_2, I_3, \ldots, I_n\}$ represents a finite set of items. And transaction database D over set of items $X \subseteq I$. A transaction can be represented as $T = \{tid, X\}$. The vertical database D' composed of items and set of transactions contains those items over D. Mathematically,

$$D' = \{(I_j, C_{ij} = \{tid \mid I_j \in X, (tid, X) \in D\})\}$$

Here, C_{ij} cover of I_j. The support of an Itemset Y in transaction database D given a number of transactions contains that itemset. Formally,

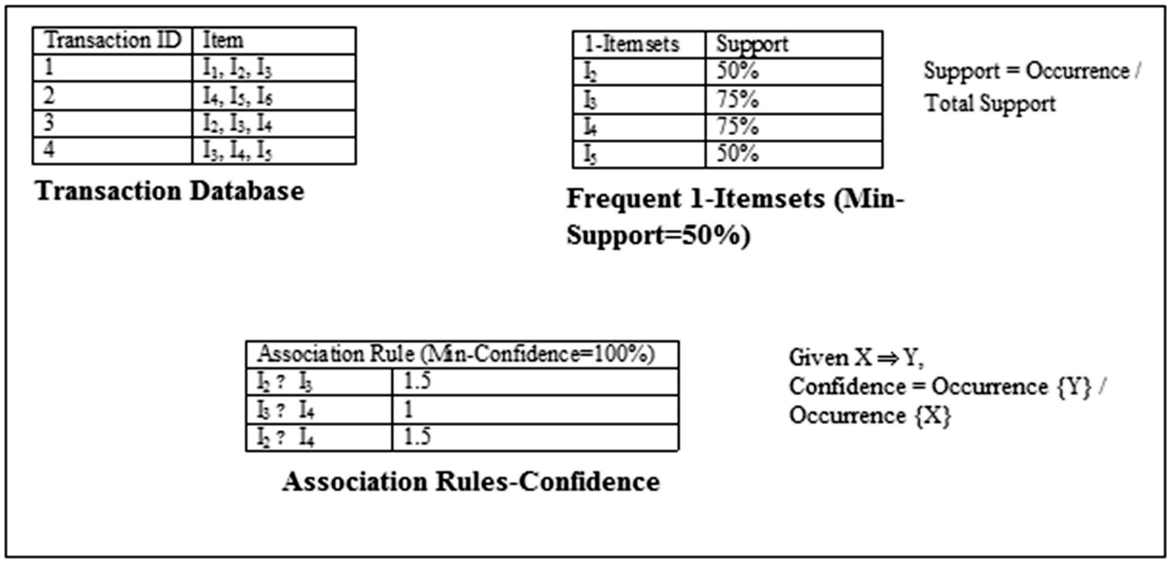
$$\text{Support}\ (Y) = \{tid \mid Y \subseteq X, (tid, Y) \in D\}$$

$$\text{Support}\ (Y) = \left| \bigcap_{i_j \in Y} \cap C_{i_j} \right|$$

In other words, occurrence probability of an itemset in transactional database D is called support. Confidence in transaction database D is the percentage of X over Y i.e. if transaction contains X itemset then it contains Y also. Confidence can also called as conditional probability that transaction contain Y if contains X. It is represented as $X \Rightarrow Y$. Formally,

$$\text{Confidence(X)} = \sigma(X \cup Y)/\sigma(X)$$

Intuitively, itemsets with support greater than threshold σ are said to be frequent. It has a monotonic property based on the principal that if itemset is not frequent than none of its superset will be frequent. Frequent Itemset Mining often presented as collection of if-then association rules. Association rule is formulated as $X \rightarrow Y$, if X appears in a transaction Y likely to appear. Association rules are categorized under four main categories: Boolean Association rules, Quantitative Association Rules, Multidimensional and Multilevel Association Rules [7].



Transaction ID	Item
1	I_1, I_2, I_3
2	I_4, I_5, I_6
3	I_2, I_3, I_4
4	I_3, I_4, I_5

Transaction Database

1-Itemsets	Support
I_2	50%
I_3	75%
I_4	75%
I_5	50%

Frequent 1-Itemsets (Min-Support=50%)

Support = Occurrence / Total Support

Association Rule (Min-Confidence=100%)	
I_2 ? I_3	1.5
I_3 ? I_4	1
I_2 ? I_4	1.5

Association Rules-Confidence

Given $X \Rightarrow Y$,
Confidence = Occurrence {Y} / Occurrence {X}

Figure 1: Transaction Database, Frequent Itemsets and Association Rules

2.2. Association Rule Mining

All ARM algorithms discussed here are composed of following characteristics:

Design Approach: Most algorithms use bottom-up approach for candidate generation. Frequent Itemsets are extended step by step with iteration. Some have proposed top-down and hybrid approaches. In top-down approach, un-necessary pattern generation can be avoided. For long frequent Itemsets, top-down approach might be preferred.

Candidate Generation: FIM algorithms require to generate candidate set to mine frequent subsets. Complete Candidate Generation, a preeminent approach generates all candidates for next step frequent subset. Sometimes, heuristic approaches are preferred for fast processing.

Layout: Two types of layouts are mainly used in FIM: Horizontal and Vertical. Horizontal layout stores transaction ids along with the items of the transaction. As compared, Vertical layout stores the itemset along with the transaction id containing the item.

Interestingness Measure: There are two Interestingness measures to evaluate the of association rules. Descriptive measure takes independency between itemsets. Most algorithms are based on descriptive measures; rules defined are simple and easy to use. Probabilistic measure considers the indetermination of confidence and imposes minimum threshold on it. Probabilistic approach requires prior statistical knowledge to design rules.

Database: In Database, data can be categorized under certain and uncertain. Certain database includes transactional, unstructured and semi-structured data. Uncertain database includes data like GPS, Whether etc. In uncertain database, Regressions, Poisson distribution etc. techniques are applied to mine frequent itemsets. As here, strong emphasis imposed on certain database, Literature not given much attention to Uncertain FIM algorithms.

Frequent Itemsets: FIM distinguishes frequent Itemsets depend on the completeness of patterns. In recent years, research have been dedicated to user-friendly concepts of Maximal frequent, closed frequent, Constrained frequent, Near-Match frequent, and top-k frequent Itemsets [7] are various kinds of frequent Itemsets generated by FIM algorithms.

2.2.1. ARM Algorithms

Apriori [2] is a traditional algorithm uses horizontal database for candidate generation. It uses BFS property to compute the support of items. K-length frequent itemset used to compute K + 1 *candidate itemset*. It starts with first iteration over D and computes the support. In the next step, frequent items from the iteration are used to compute *candidate itemset* of length 2. Minimum support threshold σ applied in iteration reduces the search space. In all iterations, Apriori scans the database to compute support count. The process made the job very time consuming. AprioriTid [8] algorithm reduces the time of support count calculation by replacing the transaction Identifier with the items in the transaction. Apriori Hybrid [9] combined approaches of AprioriTid and Apriori algorithm. Apriori Hybrid uses basic Apriori Algorithm in the initial phase of frequent itemset mining and AprioriTid in later phase.

SEAR Algorithm [10] modified version of Apriori Algorithm. Same steps of Apriori like candidate generation and pruning is used for frequent itemset mining. SEAR has modified the data structure of Apriori from set to prefix tree. Leaf nodes store the candidates for next iteration of frequent itemset mining. Both frequent itemsets and candidates are stored in prefix-tree. Prefix-tree gives better processing and memory utilization as compared to set.

The partitioning Algorithm [11] scans the database two times for frequent Itemsets. I first phase, it divide the database in multiple partitions without overlapping and find their frequent itemsets locally. In second phase, it finds the frequent itemsets from whole database based on support. Combined results give the frequent itemsets and reduce the search space.

SPEAR Algorithm [12] is combined approach of SEAR and Partitioning Algorithm. It scans the database two times. Firstly, it scans the database and generates global frequent items. Then database is partitioned without overlapping. Each partition generates frequent itemsets using SEAR. SPEAR computes the total frequent itemsets by counting the SEAR's active candidates. SPINC [10] is a modified version of SPEAR, use incremental partitioned approach. Here, incremental means partitions not compute the results by own but share the intermediate results to reduce the duplicate computation overhead.

Dynamic Itemset counting algorithm [13], DHP [14] and Perfect DHP [15] are Apriori based algorithms. DIC generates the candidate Itemset dynamically as transaction read and DHP is heuristic based algorithm. It performs Hashing to filter out the unnecessary itemsets and generating candidate itemsets for next phase. Perfect DHP reduces the size of the database that not contains frequent itemset. DHP is a very efficient technique for generating large itemsets. DHP is effective as it not only reduce the itemsets in transaction but also the number of transactions from the database.

IHP [16], unlike DHP Transaction Identifiers (Tid) of transaction contains the item to be hashed in Hash Table named Transaction Hash Table (THT). Firstly, transactions are hashed in the database. Secondly, THT generated between hash and Itemsets. The count of the item against that hash in database stored. 1-Itemsets are generated by pruning the infrequent Itemsets whose total count is less than threshold. THT of frequent 1-Itemset can be used recursively to find frequent k-Itemsets.

Dynamic Programming Algorithm [17] improves the performance of Apriori for frequent candidate itemset for 1-itemset and 2-itemset. Dynamic Programming reduces the redundancy and save the previous results. Special data structures are used for efficient storage.

Eclat [18] uses vertical database for fast computation and DFS in the prefix tree to find the frequent itemset candidates. Comparing with Apriori, Eclat iterates the complete database only once to compute the support of items. A prefix tree is generated; depth first search applied at each step for candidate generation. Frequent itemsets are prefix for candidate generation in tree.

FP-Growth [19] is a famous tree association Mining Algorithm. It scans the database two times. Firstly, find the support of the items in database. Secondly, arranges the items in Frequent Itemset header table in decreasing order based on their support. Frequent Itemset header table contains two fields, item and node. Node-link points to the first node of FP- Tree. Depth first strategy reduces the search space in FP-Growth and gives better memory utilization.

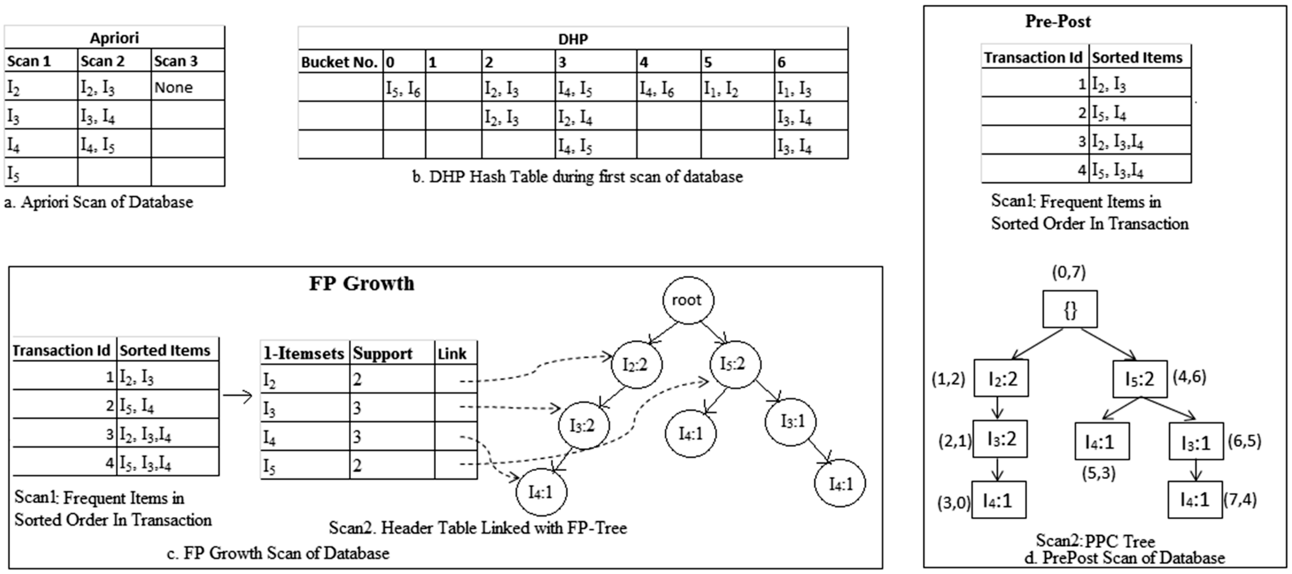
FIUT [20] algorithm enhances the FP-Growth algorithm using special frequent itemset Ultrametric tree (FUIT) structure for mining frequent Itemsets. Low I/O communication, improved dataset partitioning, compressed storage and avoid recursive traversing are major four advantages of FUIT. Compressed Storage achieved by avoiding conditional pattern base. Frequent Itemsets are generated from leaf nodes without traversing tree recursively. Transaction clustering is done improve the database partitioning and substantially reduces the search space.

Table 1
Algorithm characteristics of traditional FIM techniques
K-size of longest frequent Itemset

S.No.	*Approach*	*Database Layout*	*Database Structure*	*Number of Database Scans*
1	Apriori	Horizontal	Hash Tree	K
2	Apriori Tid	Horizontal	Hash Tree	K
3	Apriori Hybrid	Horizontal	Hash Tree	K
4	SEAR	Horizontal	Prefix Tree	K
5	DIC	Horizontal	Hash Tree	<=K
6	DHP	Horizontal	Hash Tree	K
7	IHP	Horizontal	Hash Tree	K
8	Dynamic Programming	Horizontal	None	K
9	Eclat	Vertical	Hash Tree	>3
10	FP Growth	Vertical	Prefix Tree	2
11	Partition	Vertical	None	2
12	SPEAR	Horizontal	Prefix Tree	2
13	SPINC	Horizontal	Prefix Tree	2
14	H-Mine	Horizontal	Linked List	2
15	Context Based	Horizontal	None	K
16	Pre-Post	Vertical	N-List	2
17	FIN	Vertical	Node Set	2
18	GUHA	Horizontal	Statistical	2
19	CARMA	Vertical	None	K
20	PPV	Vertical	Node-List	K
21	FIUT	Vertical	Ultrametric Tree	2
22	CBAR	Horizontal	None	1 + Contrast with Cluster Partial Table

H-Mine Algorithm [21] a scalable for frequent itemset mining for data set fit into memory. Give high performance when combined with FP Growth for large dataset. Linked Queue data structure is used to maintain the links. Header table in H-Mine store item, support and link where link addresses to the linked queue.

Context based Association Mining Algorithm [22] is based on the context variable. It is based on the theory association between objects can be different based the context; context can be any state, entity or event. Contextual situation generates positive and negative associations.



Apriori

Scan 1	Scan 2	Scan 3
I_2	I_2, I_3	None
I_3	I_3, I_4	
I_4	I_4, I_5	
I_5		

a. Apriori Scan of Database

DHP

Bucket No.	0	1	2	3	4	5	6
	I_5, I_6		I_2, I_3	I_4, I_5	I_4, I_6	I_1, I_2	I_1, I_3
			I_2, I_3	I_2, I_4			I_3, I_4
				I_4, I_5			I_3, I_4

b. DHP Hash Table during first scan of database

FP Growth

Transaction Id	Sorted Items
1	I_2, I_3
2	I_5, I_4
3	I_2, I_3, I_4
4	I_5, I_3, I_4

Scan1: Frequent Items in Sorted Order In Transaction

1-Itemsets	Support	Link
I_2	2	--
I_3	3	--
I_4	3	--
I_5	2	--

Scan2. Header Table Linked with FP-Tree

c. FP Growth Scan of Database

Pre-Post

Transaction Id	Sorted Items
1	I_2, I_3
2	I_5, I_4
3	I_2, I_3, I_4
4	I_5, I_3, I_4

Scan1: Frequent Items in Sorted Order In Transaction

Scan2: PPC Tree

d. PrePost Scan of Database

Figure 2: (a) Apriori Scan- Requires 2 scans to find the frequent Itemset of Size of 2. (b) DHP algorithm generates a Hash Table; support is applied on the bucket size. Here, if threshold is 2 then 0, 1, 4 and 5 buckets will automatically discarded during next scanning. (c) FP- Growth sorted the items in transaction and generate FP-Tree (d) Pre-Post's first scans is similar to FP-Growth and first scanning, generates PPC tree to create a N-List Data Structure

PrePost Algorithm [23] and PPV fast Vertical Data Mining Method [24] are based on PPC tree. PrePost employs N-List data structure while PPV is based on Node-List. In PPC-tree each node stores Item Count, Pre-Order and Post-Order and. PrePost algorithm requires two database scans to construct PPC-tree and generate N-List. Apriori or any other algorithm can be used for K-frequent Itemset based on N-List. FIN [25] employs vertical database with NodeSet data structure. NodeSet use Pre-Order to construct the tree. PrePost and PPV used PPC encoding which made mining process very time-consuming. In Contrast to both, FIN save almost half of the memory uses.

GUHA Method [26] combines logical and statistical approaches to mine frequent itemsets. It uses 4ft-Miner Procedure for mine the patterns. CARMA [27] is an Online Association Mining technique to compute long frequent itemsets. The algorithm maintains set of large itemsets with corresponding support. It provides the feature to change the threshold at any time during first scan.

CBAR [28] cluster based association mining algorithm for discover large frequent Itemsets. It require only single database scan followed by contrasts between partial cluster tables. Database scan generate frequent 1-Itemsets and cluster the database by decreasing lengths. Cluster-Table (k) store the transaction of length k. Frequent Itemset of length k can be finding with the reference to Cluster-Table (k). CBAR not only increasing the efficiency of mining but also ensures correctness of results.

PCAR [29] Pruning Classification Association Mining Algorithm combines minimum frequency of items with minimum frequency of Itemsets. Firstly, infrequent items deleted from the database based on the min-support. Then, frequent items are combined to generate frequent Itemsets of length k.

Comparison of above discussed techniques is given in Table 1.

2.2.2. Parallel and Distributed Association Rule Mining

Many researchers have contributed towards parallelization of Association Rule Mining in large databases. Parallelization of tasks becomes necessity to deal with huge databases. Shared and Distributed memory architectures are emerged. Shared memory systems allow multiple programs to simultaneously access the same memory. In Distributed memory architecture, memory not shared among processors and they only share data among each other. Each processor has access to its own memory. Because of the scalability of distributed systems, they are becoming more common and easy to use. MPI, most common programming model for distributed computing. But, its efficiency can be observed only on low level programming languages i.e. C and FORTRAN.

MapReduce has simplified distributed processing. Recent research in distributed computing has taken MapReduce as parallel programming paradigm. MapReduce provides high tolerance, good scalability and availability.

Various parallel and distributed techniques in ARM are discussed below:

Count Distribution is parallelization of Apriori Algorithm. In CD [30], transactions are distributed among local processors. Each processor scans the local database and a hash tree build from all candidate itemsets. In each local partition, processor calculates the support of candidate Itemsets and share among all the remote processors for global support count. Once the global frequent Itemsets have been determined; next candidate itemsets are generated locally each iteration. CD not able to parallelize the computation because each it communicates with remote processors to find the global frequent Itemsets. Large number of Itemsets and processors are major drawbacks of CD. As in case of large number large Itemsets fails to store the hash tree in memory and if processors becomes large communication between other processors will also increase.

Fast Distribution [31] based on Count Distribution reduces the number of candidates in Hash tree. The itemsets which are not frequent locally are removed from the hash tree. It reduces the communication between processors. Each site requests count for candidates assigned to it. Remote site share their local support count and broadcasts the global support count.

Data Distribution [30] overcomes the memory problem of CD by generating disjoint candidate Itemsets at each processor. However, complete database scan is required in all iterations to determine the global support. Data Distribution resolves the memory problem of CD but increases communication overhead in all iterations and comparatively poor than CD.

Candidate Distribution [30] distributes the disjoint candidate itemsets and transactions between processors. Heuristic Distribution is done in such a way that no synchronization and in between communication is required between processors to find frequent itemsets. Algorithm divides $l_k - 1$ frequent itemsets among partitions to determine l_k candidate itemsets in heuristically defined pass l.

Table 2
Characteristic approach of various Parallel and Distributed frequent Itemset Mining Techniques

S. No	*Algorithm*	*Characteristics*
1	Count Distribution	Apriori Based
2	Fast Distribution	CD- based reduces candidates by removing local infrequent itemsets
3	Data Distribution	Candidate Itemset Partitioning; Scan complete database in all iteration
4	PEAR	Parallel SEAR used candidate prefix tree
5	PPAR	Parallel Partition based algorithm
6	IDD	Ring based broadcast and Candidate Itemset Partitioning
7	Candidate distribution	Heuristically distributes candidates and database among processors
8	Hybrid Distribution	Combine CD and DD
9	NPA	All processors stores the copy of candidate Itemsets similar to CD

S. No	*Algorithm*	*Characteristics*
10	SPA	Candidate itemset partitioning; similar to DD
11	HPA	Candidate Itemset partitioned based on hash function
12	F-FPDM	Parallel FP-Growth based on FP-forest data structure
13	MLFPT	Multiple frequent pattern trees generated on distributed environment locally
14	ODAM	Apriori Based Parallel algorithm with elimination of infrequent 1-itemsets
15	PD-CLUB	Bit- Map based
16	FP-Growth Heuristic	Tree-Partitioning algorithm builds FP-Tree to identify heuristically defined N frequent items
17	Parallel FP-Growth	FP-Growth Based Algorithm
18	APM	All processors generate itemsets dynamically and independent support count
19	Clustering FIM	Both Sequential and parallel Itemset Mining
20	IMRApriori	Eliminate partial infrequent Itemsets from Apriori (MapReduce)
21	BigFIM	MapReduce Apriori
22	MRApriori	MapReduce Apriori
23	BPFP	MapReduce based FP-Growth with load balancing
24	SPC	K-frequent itemsets generated in k passes (MapReduce)
25	FPC	Based on support candidates from different fixed consecutive database passes are combined
26	DPC	Candidates are dynamically combined depend on work load
27	R-Apriori	Uses Spark architecture to reduce I/O bottlenecks
28	RFP-Growth	FP-Growth based algorithm eliminates the intra-property frequent itemsets
29	CARM	HD-Mine and FD-Mine based
30	SEARUM	A cloud based service based on PFP-Growth algorithm
31	FiDoop	Modified the data structure of FP-Growth from FP-Tree to FP-Ultrametric Tree for better performance
32	FiDoop-HD	FiDoop with multiple cache files depend on itemset length

PEAR algorithm [32] based on the parallelization of SEAR algorithm generates candidate prefix tree. Each processor generates local support counts. Global support count is determined after combining local support counts. Infrequent itemsets are eliminated and process continues.

Intelligent Data Distribution [33] overcomes the problems of Data Distribution. In IDD, ring based broadcast of local databases is done. IDD algorithm suffers from the problem of high communication and comparatively smaller number of candidates.

Hybrid Distribution [33] addresses the above mentioned problems by combining the Count Distribution and Intelligent Data Distribution. HD algorithm partitions the candidate into large sections and a number of processors are assigned to each section.

NPA and SPA algorithms proposed in [34] are similar to CD and DD respectively. HPA is similar to IDD to reduce the computation of DD but the approach taken has been different.

F-FPDM [35] algorithm uses FP-forest data structure. Algorithm is based on parallelization of FP-Growth algorithm using depth first search strategy. Databases are divided to processors by core processor. Each processor creates FP-Forest on portion of the database. Core processor performs the synchronization and merging of FP-Forests of all other processor nodes. Tree of FP-Forest is dynamically assigned to processor nodes as a sub- task.

MLFPT (Multiple Local Frequent Pattern Tree) algorithm [36] has taken FP-Growth as base algorithm requires only two full database scans. No candidate itemsets are generated. Workload is equally distributed among all processors. Firstly, parallel frequent pattern trees are generated. After only, actual mining is done for these data structures.

ODAM (Optimized Data Association Mining) [37] has taken Parallel Apriori as base algorithm. Algorithm starts with eliminating the *1*-infrequent itemsets from the database partition and store in main memory. The idea behind is to reduce the transaction size and find the identical transactions. Also, initial dataset contains both frequent and infrequent datasets and size is too large to store in memory. Finally, it moves all main memory partition to temporary file and continues the process for all partitions.

PD-CLUB [38] is a parallel bitmap-based algorithm to find frequent itemsets by differential mining in cluster refinement. It creates the database clusters and applies differential techniques to eliminated common patterns and mines the database.

FP-Growth based Heuristic algorithm [39] a parallel mining technique based on tree-partitioning. One FP-Tree builds in memory and independent partitions developed in parallel. Load balancing is achieved through equal partitioning of tree between threads. N frequent items are identified where N is heuristically identified. The transactions are partitioned and passed to threads. 2^N chunks are generated and divided to N-1 threads. Process is recursively applied to build FP-Growth tree.

Parallel FP Growth [40] generates global FP-Tree from accumulation of local FP-Conditional Patterns results. Initially, Transactional database is partially divided among all nodes. Nodes share their support count to create global support count. All nodes creates local conditional pattern base based on F-List. After the generation of Local Conditional Patterns hash function is used to determine node going to process conditional patterns.

Adaptive parallel mining algorithm [41] proposed for shared memory systems. All the processors generate itemsets dynamically and support count independent from each other without any synchronization. Adaptive interval configuration and virtual partition pruning technique reduces the database scanning and number of candidates.

Clustering based FIM [42] proposed both sequential and parallel Itemset Mining. MapReduce framework is applied to parallelize the mining. Instead of *1*-itemset or *k*-itemset, representative examples in the cluster can be used for searching. Firstly, transactions are divided in K-Clusters using K-Medoids algorithm. Two list accepted and excluded are computed. Itemsets in clusters are sorted by their decreasing lengths and occurrences. Random representative is chosen form each cluster. If candidate is found frequent in local then itemset will be accepted otherwise global test is performed check itemset is frequent or not.

IMRApriori (Improved MapReduce Apriori Algorithm) [43] offers better performance by pruning partial infrequent Itemsets as partial frequent itemsets in large amount can overload the mappers. When few mappers output an itemset as frequent Itemset called as Partial frequent Itemsets (INS-Itemset). The property of maximum support count for partial frequent itemset with size D_i is $(s \times D_i) - 1$ has defined. Reducers of phase 1 apply the above property to remove INS-Itemsets. There are several approaches on MapReduce Apriori like BigFIM, MRAprori and MapReduce Apriori [51, 52, 53] are proposed by many researchers to increase the efficiency of existing Apriori in distributed environment.

BPFP (Balanced Parallel FP- Growth) [44] algorithm parallelize the FP-Growth using MapReduce approach. BPFP implements load balanced feature to improve the performance of PFP. BPFP generates output in two steps. First step generates F-List, a key-value pair output of mappers are summed at reducers. Sorted list of frequent itemsets in descending order is called F-List. Secondly, F-List divided into Q balanced groups again in two sub-steps. To construct conditional pattern base, load calculated based on the amount of work on FP-Growth. After, calculated load divided among different units. Now, FP-Growth is implemented on group-dependent transactions.

SPC, FPC and DPC [45] algorithms are based on MapReduce has taken Apriori as base algorithm. SPC (Single Pass Counting) generates *k*-frequent Itemsets in *k*-passes of MapReduce phase. Mappers scan the database and key-value pairs are generated. Reducers perform candidate generation and support counting functions. FPC (Fixed Pass Combined-Counting) algorithm combines the candidates from different fixed consecutive database passes depend on their support counts. DPC (Dynamic Passes Counting) depend on the work load of units dynamically combines candidates from different successive passes.

R-Apriori [46] MapReduce based algorithm used Spark to overcome I/O bottlenecks in MapReduce. In phase 1, mappers produce key-value pair output for itemsets. Reducer counts the itemsets and prunes the itemsets based on their support counts. Bloom filter is used to store support counts of phase 1. Bloom filter provide high speed compared to hash tree thereby increasing the performance of algorithm.

RFP-Growth [47] and CARM [50] algorithms improves the efficiency of FP-Growth. RFP-Growth eliminates the intra-property frequent itemsets. Itemsets have same prefix or property are called intra-property itemsets. Major three phases of algorithm are: pre-processing, FP-Growth and reverse elimination to remove intra-property frequent itemsets. CARM contains two main algorithms HD-Mine and FD-Mine provides high workability on cloud computing environment. The mining of conditional FP-Tree done on different nodes based on availability.

SEARUM [48], a cloud based service for Association Rule Mining in distributed computing. It has implemented PFP-Growth [] algorithm to uncover frequent itemsets from network traffic data. A series of MapReduce jobs performed from network data acquisition to association rule aggregation. Each job may receive the input from one or more preceding jobs.

FiDoop [49], a parallel frequent Itemset Mining on MapReduce Programming Model. FiDoop incorporates Frequent Itemset Ultrametric tree (FIU-Tree) as it provides better storage, low I/O overhead, recursive traversing and natural partitioning of data set [52]. Three MapReduce jobs are executed gradually to mine the dataset. FIU- Tree is generated in first two phases of MapReduce. First phase scans the complete database and creates frequent 1-Itemsets. In second phase, again database scanning is done to generate frequent k-Itemsets and construction of k-FIU tree. Third phase, generate short frequent itemsets independent of large frequent Itemsets on MapReduce.

FiDoop give high performance on low-dimensional dataset. To efficiently handle high-dimensional dataset some modifications have been done existing FiDoop called FiDoop-HD. FiDoop-HD stores the second phase output of MapReduce Job into multiple cache files depending on Itemset lengths. FiDoop-HD proves to better than FiDoop because of in-built balancing and Itemsets decomposition of previous stages saved in new files.

3. CHALLENGING ISSUES

Above discussion reveal that researchers are continuously contributing towards increasing the performance, availability, parallelization and scalability of frequent Itemset Mining in Big Data. Despite, more attention needs to be given to the issues related to ease and flexibility. Some of the open issues are discussed below:

Database Scan: Most of the algorithms scans the database number of times to generate frequent Itemsets. Complete database scanning of exponentially large database not only reduces the performance but also a time consuming job.

Dynamic load balancing: All present parallel and distributed algorithms employ static load balancing scheme with the assumption of data distributed among various node in homogeneous environment. But Cloud Computing environment encompass heterogeneous data decomposition with transient loads. Hadoop MapReduce also featured implicit data distribution among nodes. Dynamic load balancing becomes very crucial job in such environment.

High Dimensionality: All present algorithms preforms well only on few thousand data dimensions or items. Itemset size implicitly escalates the complexity. Although dimensions of itemsets not directly proportional to Complexity of the algorithm. But enumeration of maximal Itemset can be the definite solution to the problem.

Data Location: Today geographically dispersed organizations store large amount of data in distributed environment. Although many researchers proposed different Hadoop MapReduce based algorithms for frequent Itemset Mining in distributed environment. But performance, cost, reusability and global database pruning are major issues.

Data Skew: Data skew a major problem in frequent itemset mining has adverse effect on load balancing. In MapReduce, data implicitly distributed among various mappers for parallel computation. Nodes usually contain equal sized data blocks. However, frequent itemsets generated by mapper may be highly skewed. Means, some Mappers may generate large frequent itemsets then other. MapReduce phase cannot obtain result without completion of Reducers job. Highly skewed mappers may require large computation time. Consequently, it effects the parallelization and performance.

Interestingness of Pattern and Rule Generation: Current research focuses on the length of frequent itemsets generated from database. However, the rule generation, usability and interestingness measure of the itemset equally important for today's business. On the assumption of few itemsets, rule generation found to be a cheapest task. But, actual complexity to generate rules is $0(r.2^l)$, where l is the maximum length of itemset and r denotes the number of frequent itemsets.

4. CONCLUSION

Social media, E-Commerce, Internet Banking, GPS, Telecommunication and industry globalization give explosive growth in data around the world. This huge amount of data introduced as Big Data. Parallelization is required to analyze and extract information from such a massive amount of data. MapReduce lucrative algorithm as parallel programming model processes Big Data on large clusters. The emphasis of this paper is to reveal both early and recent literature on frequent itemset mining techniques. In this paper, we have provided the survey of research in traditional and advanced frequent itemset mining techniques as well as characteristic measures and comparison of approaches. Further, some challenging issues and open problems have been discussed.

REFERENCES

[1] R. Agrawal, T. Imieli´nski, and A. Swami, "Mining association rules between sets of items in large databases," in SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, pages. 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for Mining Association Rules in large databases. In Proc. VLDB, pages. 487–499, 1994.

[3] Improving Decision Making in the World of Big Data http://www.forbes.com/sites/christopherfrank/2012/03/25/improvingdecision-making-in-the-world-of-big-data/.

[4] Dean Jeffery, Ghemawat Sanjay, "MapReduce: Simplified Data Processing on Large Clusters", "Google Publications", 2004.

[5] 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.

[6] 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.

[7] Han J, Kamber M, Data mining: concepts and techniques, 2nd edn. Morgan Kaufmann, 2006.

[8] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, Proceedings 20th International Conference on Very Large Data Bases, pages. 487–499. Morgan Kaufmann, 1994.

[9] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages. 307–328. MIT Press, 1996.

[10] Mueller A. Fast sequential and parallel algorithms for association rule mining: A comparison. Technical Report CS-TR-3515, Department of Computing Science, University of Maryland, College Park, MD, 1995.

[11] A. Savasere, E. Omiecinski, and S. Navathe, “An Efficient Algorithm for Mining Association Rules in Large Databases”, Proc. 21st Very Large Data Bases Conference, 1995.

[12] Mohammed J. Zaki, Rensselaer polytechnic Institute. Association Mining: A Survey. IEEE Concurrency, 1999.

[13] S. Brin, R. Motwani, J. Ullman, and S. Tsur, “Dynamic Itemset Counting and Implication Rules for Market Basket Data,” ACM SIGMOD conference. Management of Data, May 1997.

[14] J.S. Park, M. Chen, and P.S. Yu, “An Effective Hash Based Algorithm for Mining Association Rules”, ACM SIGMOD International conference. Management of Data, May 1995.

[15] OZEL, S. A. AND GUVENIR, H. A. An algorithm for mining association rules using perfect hashing and database pruning. In 10th Turkish Symposium on Artificial Intelligence and Neural Networks, Gazimagusa, T.R.N.C., A. Acan, I. Aybay, and M. Salamah, Eds. Springer, Berlin, Germany, pages. 257–264, 2001.

[16] J.D. Holt, S.M. Chung, Mining association rules using inverted hashing and pruning, Information Processing Letter, pages. 211–220, 2002.

[17] D. Bhalodiya, An efficient way to find frequent pattern with dynamic programming approach, Nirma University conference on Engineerin, NUiCONE, pages. 28-30, Nov, 2013.

[18] 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.

[19] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, 2013 Journal of Data Mining and Knowledge Discovery (8) – 1 53 -87, 2004.

[20] Y.-J. Tsay, T.-J. Hsu, and J.-R. Yu, “FIUT: A new method for mining frequent itemsets,” Inf. Sci., vol. 179, no. 11, pp. 1724–1737, 2009.

[21] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, “H-Mine: Fast and Space-Preserving Frequent Pattern Mining in Large Databases,” IIE Trans. Inst. of Industrial Engineers, vol. 39, no. 6, pp. 593-605, June 2007.

[22] Shaheen, M; Shahbaz, M; and Guergachi, A; Context Based Positive and Negative Spatio Temporal Association Rule Mining, Elsevier Knowledge-Based Systems, pp. 261-273, Jan 2013.

[23] Deng Z., Wang Z., Jiang J.J. (2012). A new algorithm for fast mining frequent itemsets using N-lists. SCIENCE CHINA Information Sciences, 55(9), 2008-2030

[24] Deng Z.H., Wang Z.H. A new fast vertical method for mining frequent itemsets. International Journal of Computational Intelligence Systems, 3(6), 733-744, 2010.

[25] Deng, Z. H., & Lv, S. Fast mining frequent itemsets using Nodesets. Expert Systems with Applications, 41(10), 4505–4512, 2014.

[26] Petr Hajek b,*,2, Martin Holeˇna b,2, Jan Raucha,1,The GUHA method and its meaning for data mining, Journal of Computer and System Sciences, 34–48, Elsevier, 2010.

[27] C. Hidber. Online association rule mining. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, volume 28(2) of SIGMOD Record, pages. 145–156. ACM Press, 1999.

[28] Y.J. Tsay, J.Y. Chiang, CBAR: an efficient method for mining association rules, Knowledge-Based Systems 18, 99–105, Jan 2013.

[29] Hui Zhang, Qingying Qiu, Zhaoxia Wang, PCAR: An Efficient Approach for Mining Association Rules, Fuzzy Systems and Knowledge Discovery, pages. 605-609, 2008.

[30] R. Agrawal and J.C. Shafer, Parallel Mining of Association Rules, IEEE Trans. Knowledge and Data Eng., vol. 8, no. 6, pages. 962-969, Dec. 1996.

[31] D. Cheung et. al., "A Fast Distributed Algorithm far Mining Association Ruler," Proc. 4th Int'i Conf. Parallel and Distributed Information Systems. IEEE Computer Soc. Pres, LOS Alamitoi, Calif., pages. 31-42, 1996.

[32] MUELLER, A. Fast sequential and parallel algorithms for association rule mining: a comparison. Tech. rep. CS-TR-3515, Department of Computer Science, University of MaryLand, College Park, MD, 1995.

[33] E.H. Han. G. Karypir,andV. Kumar, "Scalable Parallel Data Mining for Association Rules." Proc. ACM Conf. Management of Data, ACM Pres, New York, pp. 277-288, 1997.

[34] T. Shintani and M. Kitsuregawa, ªHash Based Parallel Algorithms for Mining Association Rules,º Proc. Conf. Paralellel and Distributed Information Systems, 1996.

[35] E. Han, G. Karypis, V. Kumar. Scalable Parallel Data Mining for Association Rules. In TKDE 12(2), 2000.

[36] Osmar R. Za¨ıane, Mohammad El-Hajj, and Paul Lu. Fast parallel association rule mining without candidacy generation. In ICDM, 2001.

[37] Sujni Paul, (2010) "An Optimized Distributed Association rule mining algorithm in Parallel and distributed data mining with XML data for improved response time", International Journal of Computer Science and Information Technology, Volume 2, Number 2, April 2010.

[38] 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.

[39] D. Chen, C. Lai, W. Hu, W.G. Chen, Y. Zhang, and W. Zheng, "Tree Partition Based Parallel Frequent Pattern Mining on Shared Memory Systems," Proc. IEEE Parallel and Distributed Processing Symposium, 2006.

[40] Iko Pramudiono and Masaru Kitsuregawa. Parallel FP-Growth on PC cluster. In PAKDD, 2003.

[41] Kan, D. C., "An Adaptive Algorithm for Mining Association Rules on Shared Memory Parallel Machines", Distributed and Parallel Databases, Vol. 9, pages. 99– 132, 2001.

[42] 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.

[43] Farzanyar, Z., & Cercone, N. (2013, August). Efficient Mining of Frequent itemsets in Social Network Data based on MapReduce Framework. In Proceedings of the 2013 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pages. 1183-1188, 2013.

[44] 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.

[45] 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.

[46] Rathee, S., Kaul, M., Kashyap, A.: R-Apriori: an efficient apriori based algorithm on spark. In: PIKM'15, Melbourne, VIC, Australia. ACM, 2015.

[47] Li, X. An algorithm for mining frequent itemsets from library big data. Journal of Software, 9(9), 2361–2365, 2014

[48] D. Apiletti, E. Baralis, T. Cerquitelli, S. Chiusano and L. Grimaudo, "SEARUM: a cloud-based SErvice for Association RUle Mining [J]", 2013 12th Ieee International Conference on Trust, Security and Privacy in Computing and Communications (Trustcom 2013), IEEE, pages.1283-90, 2013.

[49] Xun, Y., Zhang, J., Qin, X. FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015.

[50] S. Natarajan, S Sehar, A Noval Algorithm for Distributed Data Mining in HDFS, "ICoAC",pp. 93-99, 2013.

[51] S. Moens, E. Aksehirli and B. Goethals, "Frequent Itemset Mining for Big Data," in Proceedings IEEE International Conference on Big Data, pages. 111–118, 2013.

[52] A. Pradeepa, and A. S. Thanamani, PARALLELIZED COMPRISING FOR APRIORI ALGORITHM USING MAPREDUCE FRAMEWORK, International Journal of Advanced Research in Computer and Communication Engineering, vol. 2(11), pages. 4365-4368, 2013.

[53] O. Yaha, O. Hegazy and E. Ezat, "An Efficient Implementation of Apriori Algorithm Based on HadoopMapreduce Model," in International Journal of Reviews in Computing, vol. 12, pages. 59–67, 2012.

[54] Margaret H. Dunham and Yongqiao Xiao, Southern Methodist University, Dallas, Texas and Le Gruenwald, Zahid Hossain, University of Oklahoma, Norman UK, " A survey of Association Rules"

[55] Thabet Slimani," Efficient Analysis of Pattern and Association Rule Mining Approaches "IJITCS Vol. 6, No. 3, February 2014.

[56] J.S. Park, M. Chen, P.S. Yu, An effective hash-based algorithm for mining association rules, in: ACM SIGMOD, 1995.

[57] D. Cheung and Y. Xiaa, "Effect af Data Skewness in Parallel Mining of Association Rules," Proc. ParifirAsIa Conf. Knowledge Dismvery and Data Mining. Lecture Notes in Computer Science, Vol. 1394, Springer- Verlag. New York. 1998, pages. 48-60.

[58] Agarwal R, Imielinski. T., Swami, Database Mining: a performance prospective, IEEE Transaction on Knowledge and Data Engineering 5(6), pages. 914-925, 1993.

[59] Yanbin Ye ; Acxiom Corp., Little Rock, AR ; Chia-Chu Chiang, "A Parallel Apriori Algorithm for Frequent Itemsets Mining", Software Engineering Research, Management and Applications, pages. 87 – 94, 2006.

[60] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In Proc. WIDM, pages. 9–15. ACM, 2001.

[61] 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.