Parallelization of Association Rule Mining: Survey

Shivani Sharma* and Durga Toshniwal†
Dept. of Computer Science & Engineering
Indian Institute of Technology, Roorkee
Uttarakhand, India. 247667
*Email: shivani.vce@gmail.com
†*Email: durgatoshniwal@gmail.com*

Abstract—**In todays big data era, all modern applications are generating and collecting large amount of data. As a result, data mining is encountering new challenges and opportunities to make algorithms such that, this voluminous data can be effectively and efficiently transformed into actionable knowledge . Traditional algorithms were designed to run sequentially over a single machine. But, as the volume of data increases computational cost associated with its processing also increases. This causes problems in analysing data on a single sequential machine and instead of assisting in data analysis, the processor serve more like a bottleneck. Parallel and distributed approaches improve the performance in terms of computational cost as well as scalability but experience some limitations during load balancing, data partitioning, job assignment, monitoring etc. MapReduce, a parallel programming model is a new concept which provides seemingly unlimited computing power, cheap storage as well as, can overcome above limitations. This makes it a topic of upcoming research interest. A detailed literature review of some existing methods is given along with their pros and cons.**

1. Introduction

With the advancement in technology, all modern applications are generating and collecting large amount of data. Some typical examples include social networking, wireless sensors network, computer networks etc. As a result, data mining require more effective and efficient algorithms to transform this huge data into actionable knowledge. Data mining can be defined as a process of extracting hidden patterns and predictive information from large volume of data. Broadly this hidden knowledge can be extracted using methods like, Association Rule Mining, clustering, sequence analysis, classification or forecasting. Association rule mining is one of the important technique used for extracting interesting patterns and correlations among items in large dataset [21]. There are two traditional approaches followed called candidate-generation based (Apriori algorithm) and Candidate-less (FP-growth algorithm).

The rule mining task can be divided into two computationally intensive subtasks i.e frequent itemsets generation and association rule generation. It require the algorithms to be very efficient and scalable. Traditional algorithms were designed to run sequentially over a single machine. But, as the volume of data increases computational cost associated with its processing also increases. This causes problems in analyzing data on a single sequential machine and instead of assisting in data analysis, the processor serve more like a bottleneck. To deal with upcoming issues, due to large scale data, parallelization of mining algorithms becomes inevitable. Parallelization of association rule mining techniques can be done either by dividing and distributing data over multiple nodes and generating association rules locally followed by merging local results to obtain global association rules. This is called *Data-parallelization.* Second approach is, parallelizing algorithm itself called *Algorithm-parallelization.*

A detailed literature review of various existing methods is given along with their pros and cons. Section 2 includes some background of traditional association rule mining techniques. Section 3 includes the discussion of some existing research work done for parallelization of association rule mining. Section 4 explores some of the existing techniques for implementing association rule mining on MapReduce with their respective advantages and limitations. Section 5 draws conclusion.

2. Association Rule Mining

Association rule mining can be defined as a process of extracting correlations and associations among items in large dataset [21]. Association rule is the implication of the form $X- > Y$ where X and Y are different items in transactional or relational database. $X- > Y$ holds in dataset D, two properties: Support(s)- Total number of transactions containing both X and Y (X union Y) out of total present transactions. Equation1.
i.e.

$$support(X- > Y) = P(XUY). \quad (1)$$

and Confidence(c)- Percentage of transaction containing X, given that it already has Y Equation2.
i.e.

$$confidence(X- > Y) = P(Y|X). \quad (2)$$

Association rule mining aims to extract frequent itemsets with *support(s)* $>=$ *Min_sup* and Finally generating association rules from these frequent item set with and *confidence(c)*$>=$ *Min_conf* . Here, *Min_sup* and *Min_conf* are threshold values. Association rule mining techniques can be divided into two major catagories- Candidate-generation based techniques and Candidate-less techniques. Number of algorithms exist under each catagory but importantly Apriori algorithm is a candidate-generation based approach and FP-Growth is a candidate-less approach.

Apriori Algorithm [21] uses the prior-knowledge in terms of properties of frequent itemset. It is an iterative approach where k-frequent itemsets are extracted using (k-1)-frequent itemsets. The algorithm exhibits anti-monotonic property called *Apriori property* [21], states that " For a frequent itemset all its non-empty subsets must also be frequent." Apriori can be seen as two step process: *Frequent itemset generation* and *Association rule generation*. Frequent itemset generation can be further divided into two subtasks called *Joining* where, l_{k-1} candidate sets are joined with itself to generate candidate k-itemsets (C_k). and *Pruning* where, all those candidate frequent itemsets with support less than *Min_sup* get pruned off. The working of apriori can be seen in Figure 1.

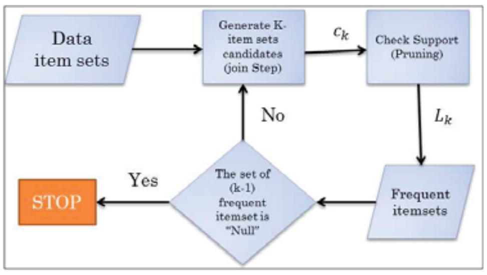
Data item sets
Generate K-item sets candidates (join Step)
c_k
Check Support (Pruning)
No
L_k
Yes
STOP
The set of (k-1) frequent itemset is "Null"
Frequent itemsets

Figure 1. Working Flow of Apriori Algorithm [21]

FP-Growth algorithm [20] is candidate-less approach therefore, do not generate candidate itemset in between. It uses tree as a data structure which is much compact and frequent itemsets are mined directly from the tree. FP-Tree is a composed and compressed representation of large dataset. This approach require only two database scans. First scan is required to obtain the support count of each item in dataset. Infrequent items with support less than *Min_sup* are removed and others are arranged in decreasing order of their support. Second scan over dataset is made to construct FP-Tree. Finally this tree is used to extract frequent itemsets directly using bottom-up strategy. Both traditional approaches can be compared over several parameters as shown in Table 2.

Traditional algorithms are designed to run sequentially over single machine [20], [21]. With increase in data volume, computational intensiveness increases such that it becomes impossible for single machine to perform efficiently [3].

3. Parallelization of Association Rule Mining

Association rule mining can be viewed as two step process [3] [21]- *frequent itemset generation* and *association rule generation*. Being computationally intensive tasks parallelization is next step required to be taken to reduce the work load over one sequential machine and distribute it among several nodes. Parallelization of association rule mining can be achieved in two major ways: *Data parallelization-* which divide and distribute data over multiple nodes and generate association rules locally followed by merging local results to obtain global association rules. Second is *Algorithm parallelization* where algorithmic tasks are distributed over number of nodes. Based on the literature review done, it can be concluded that most of the research work is being done on data parallelization.

3.1. *Parallelization of Apriori Algorithm*

Apriori is most widely used frequent itemset generation algorithm which, iteratively generate k candidate itemsets from present $(k-1)$ frequent itemsets. Several interesting and effective research work to parallelize the mining of itemsets can be seen in [11], [12]. Initially Apriori was implemented on multi-processors and further [1], [2], [13] implemented them with distributed architecture. Summary of parallel version of Apriori is defined in [8] as given below: *Count Distribution-* can be seen as a most direct form of parallelization for apriori algorithm. At each node global candidate itemset as well as frequent itemsets are stored. For respective local data present on each node, support count of candidate itemset is calculated using apriori algorithm and finally, these local results are exchanged among nodes respectively. *Candidate Distribution-* here, each node donot maintain global results but partitions the candidate itemsets with respect to the partition of dataset. Each node locally calculate support count of their own candidate itemsets. *Data Distribution-* combines the above two approches together by partitioning both datasets as well as candidate itemsets at the same time such that, each node can work independently.

Parallel implementation of finding frequent itemsets followed by sequential extraction of association rule depletes the performance gained so far when the number of generated frequent itemset is large. Hence, parallelization of second phase is equally important to enhance the overall performance in terms of scalability and efficiency. *SeaRum* [4] is one of the initial attempt in our knowledge for developing both phases of Apriori algorithm in distributed environment.

3.2. *Parallelization of FP-Growth Algorithm*

Multi-tree approach [10] was one of the initial works done to parallelize FP-Growth algorithm. Broadly, they used three steps to achieve parallelism which were followed in sequence by parallel processing flows. Initially, horizontal subset of data is analyzed. Secondly local FP-Tree is build

Comparison of Apriori and FP-Growth		
Parameters	Apriori	FP-Growth
Technique	Candidate-based	Candidate-less
Time	Execution time is more as need to produce candidate every time. (Slow)	It is much smaller than apriori. (Fast)
Memory usage	Due to large candidate generation require large space.	Uses tree data structure and donot generate candidate itemsets, require less memory.
No. of scans	Multiple scans for generating candidate sets	Only two scans are required.
Property	Use apriori property, Prune step, Join step	Conditional frequent pattern tree and base are constructed from database, saatisfying *min_sup* value.

TABLE 1. COMPARATIVE TABLE OF APRIORI AND FP-GROWTH [5]

in parallel and finally, on this local FP-Tree mining process is carried out. From every processing flow candidate patterns are obtained and then merged together. Further, enhancement were made in merging algorithms using cluster computing environment. Moreover, in [13] certain constraints were proposed for massive dataset, which should be followed in parallel itemset extraction to obtain good scalability. further improved itemset extraction with better hardware resource exploitation on multi-core processors. It proved to be a new innovation in the field which enhanced the performance of FP-Growth algorithm via improving the temporary locality for accessing the data at different levels of memory. In [12] initial effort was made to address cache-hint optimization using the above technique. Certain applications were developed [11], implementing the above for mining itemsets in parallel.

3.3. *Limitations of Parallelization of Association Rule Mining*

Implementing sequence of algorithms for association rule mining in distributed or parallel environment enhances the performance in terms of scalability and computational speed-up [20]. But, still persists various limitations while implementing sequential algorithms in parallel environment [3], [22] such as: *Partitioning and Distribution, Job assignment and Monitoring, Load Balancing, Communication Overhead.* All these activities require one central control which can moniter and maintain other distributed nodes, this may be prone to failure, costly as well as uneasy to handle for complex applications. Thus there is scope to leverage parallelization in a better fashion for further enhancing performance.

4. Association Rule Mining on MapReduce

Parallel programming model is a new concept which provide higher computation power and robustness to failure. MapReduce is one of the promising parallel programming model discussed in further Sections.

4.1. *MapReduce*

MapReduce [14] is a parallel programming model which support distributed computing, required for mining large-scale data. MapReduce composed of two procedure i.e **Map** which basically performs sorting and filtering of data and **Reduce** merges the map output to produce final result. Mapreduce is a framework which can process the parallelizable problems with massive data set by using a large number of machines. Machines collectively called as cluster which can take benefit of locality of data.

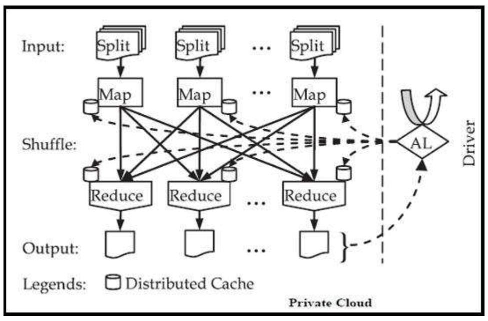

Figure 2. One Phase of MapReduce Computation [14]

Firstly *Mapping* is done by dividing datasets into subsets which are distributed as input to each processor where, code of respective mining algorithm run to obtain local frequent itemsets. Each processor produce (key/value list) as output so that next step can use it for further processing. Secondly *Reduce* step shuffle the output obtained and provide input to reduce processors. Reduce code is executed to merge the local result and obtaining global rules. Finally sorting is performed to obtain final result. All these steps combatively called phase of MapReduce as in Figure 2. One or more phases may be required to obtain desired pattern or actionable knowledge out of massive database.

Key benefits of MapReduce is that it automatically handles failures as well as hides the complexity of fault-tolerance from the programmer [8]. But support a particular kind of schema for input file hence, do not offer much space for improvements. However, the MapReduce framework [18] is not suitable for the frequent itemset mining algorithm like the Apriori algorithm which have intensive iterated computation.

4.2. *Parallelization of Apriori Algorithm on MapReduce*

Several versions of parallellized form of Apriori algorithm to handle issues like improper load balancing, uneven data distribution, high communication overhead have been developed and discussed in further Sections

In *K-MapReduce Apriori algorithm* [23] method extracts all frequent itemsets from the given dataset using k phases of MapReduce. Phase 'i' ran iteratively to obtain frequent i-itemset where *i= 1,2,3,4....k.* Improved performance in terms of scalability and computationally efficiency is obtained. but one MapReduce phase need to wait and schedule every time for next phase to start, which is pure burden to the mining task.

The waiting time issue is resolved [7], by using only a single phase to extract all frequent k-itemsets from the given data set. But, the performance of method is not much satisfactory .

In [9] *MRApriori* stratergy, introduced a new modified version of Apriori algorithm and requires only two phases of MapReduce to extract all possible frequent k-itemsets. The MRApriori performs outstandingly in terms of all desired deliverables in comparison to other approaches, as only two phases are required for extracting all possible frequent k-itemset from entire dataset. But in realistic scenerio, if number of partial frequent k-itemset generated is huge than then each node take insignificant amount of time to process hence, perform inefficiently .

An enhanced variant*IMRApriori* [3] of MRApriori reduced the processing time in case frequent k-itemset generated is huge [3]. *Efficient pruning technique* is used for removing number of partial k-itemsets which are infrequent, to achieve the computational speed-up. Significant amount of infrequent itemset get pruned of in phase one itself which improves the efficiency by removing the case of huge number of partial frequent itemsets.

4.3. *Parallelization of FP-Growth on MapReduce*

Initially, efforts were made to parallelize and speed up Apriori algorithm because of its simplicity. It has been shown that FP-Growth is much faster than Apriori hence, logically parallel implementation of FP-Growth will provide much faster speed up and low cost overhead. Initial parallel FP-Growth was implemented across multiple-thread and with shared memory in [10], [11]. But, this approach restricts the percentage of computation that can be parallelized as well as if dealing with huge dataset, requirement of huge memory space, high communication cost were another issue need to be handle. To deal with these limitations distributed variant of algorithm was designed, which run over the cluster of machines. [15] applied strategy to solve the issues like communication overhead, and achieved satisfactory performance over hundreds of node. But still issues like scalability, automatic fault-tolerance were hindering the desired performance.

In [2] *Parallel FP-Growth(PFP)* method parallelized the FP-Growth over distributed machines such that each machine executes independent mining tasks. Such partition remove the computational dependency which in turn reduces communication overhead between them. PFP divides the dataset and save them on different P Nodes. Each part is called 'Shard'. Parallel counting of support value corresponding to each item 'I' is performed on each part, using a MapReduce Pass and store these value in F-list. All 'I' items are divided into 'Q' groups (G-List), each given with unique group Id. Parallel FP-Growth uses another MapReduce Pass where 'Map' phase produces output as Key/Value pair and Reducer take output of mappers in form of group dependent transactions and again correspondingly group them into shard. Each machine again assigned with one or more shard to build a local FP-Tree and growth them recursively. Finally, discover frequent patterns. PFP is able to achieve 'Near-Linear' Speed-up and is fairly suited for query recommendation. But, it do not take load balancing into consideration, hence limits if data is huge. It also need multiple database scans as well as inter processor communication is high.

An improved variant of PFP called *Balanced PFP(BPFP)* [1] resolved the issue of load balancing when dataset is massive. The algorithm proposed additional estimate work load and dataset into several units on which association mining is performed to obtain frequent itemsets. Key Feature of BPFP is *Partitioning Mining Task into quit even subtasks*, which in turn produce higher level of parallelization. BPFP Performs well even with huge data volume.

Much similar to PFP a stratergy called *CPFP* [16] was introduced which improved in terms of inter process communication, a limitation of PFP. The algorithm implements *the separation strategy* which simply ask single visit to local database. Hence, reduces the inter processor and I/O overhead.

PIFP-Parallelized Incremental FP-Growth [17], a MapReduce Based strategy developed basically for large-scale data processing. The periodic updates of database and change in threshold value require data mining process begin from the beginning which is an overhead. Here, presents a scheme which adapt dynamic database (streaming data) as well as change in threshold value. The strategy realizes effective and efficient data mining without any repeated computation. The strategy divides the whole problem into two parts: firstly updates of the FP-Tree and secondly frequent itemset mining. Algorithm effectively reduce the duplicated work and allow to improve the speed-up, by adding more number of machines. But PIFP suffer from load balancing problem which effect the overall performance.

Summary Table				
Technique	**Platform**	**Algorithm improved**	**Achieved**	**Remark**
K-map Apriori [8]	MapReduce	Apriori	Scalability and works in only K phase	High waiting time between two phases
MR-Apriori [9]	MapReduce	Apriori	Scalability and works in only two phase	Perform insignificantly if generation of k-frequent itemset is huge, each node take insignificant amount of time
IMR-Apriori [3]	MapReduce	Apriori	Perform well and much scalable	–
PFP [2]	Mapreduce	FP-Growth	Near-Linear speed-up	Issue of load balancing, multiple scanning of dB and I/O overhead
BPFP [1]	MapReduce	FP-Growth	Near-Linear speed-up with load balancing	Additional computational overhead of load estimation
CPFP [16]	MapReduce	FP-Growth	Efficient performance, load balancing and reduced I/O overhead	Not compatable for dynamic database or change in threshold
PIFP [17]	MapReduce	Fp-Growth	Adopt incremental changes in database and threshold and reduce duplicacy	Load balancing issues still exists
SeaRum [4]	MapReduce	Apriori	Parallelization of association rule extraction phase and provide SaaS platform	–
PRAMA [6]	MapReduce	Fp-Growth/ Apriori	Near-Linear speed-up, High scalability, reduce duplicates, Extract rules Directly	Combines Random sampling and Parallelization
YAFIM [18]	Spark RDD	Apriori	Faster computation	Faster computation than Mapreduce
NIMBLE [19]	NIMBLE	FP-Growth/ Apriori	Portable, support rapid prototyping	Designed for fast and efficient implementation of MLDM algorithms

TABLE 2. SUMMARY TABLE FOR ABOVE DISCUSSED TECHNIQUES

4.4. *Other Approaches*

Beside MapReduce, many other research work using different approaches can be seen, which also ultimately improves the association rule mining process while using parallel environment. A cloud based service model*SeaRuM* [4] efficiently extracts association rules from huge frequent-items. SeaRuM run a number of distributed MapReduce jobs performing different tasks in cloud. The architecture contain following jobs Data acquisition, Data Preprocessing, Item Frequency Computation, Itemset Mining, Rule Extraction, Rule aggregation and Sorting.

A randomized approach [6], *PARMA* stands for -A Parallel Randomized Algorithm For Approximate Association Rule Mining. The approach combines the two different methods named *Random Sampling* and *Parallelization* for extracting association rules out of massively huge amount of dataset. The overall cost of association rule mining can be divided into two components: *scanning* and *Mining*.The scanning factor increases very rapidly and dominant mining and make the whole process unsellable and complex. Hence, PARMA combines both approaches in a novel fashion, it mines small random samples in parallel. Each sample is given as input to MapReduce function running in orthogonal manner. lastly, filtering and aggregation of association rule from each sample are collected. PARMA can compute association rules directly and is not limited to frequent itemset extraction. Significantly outperform and has near-linear *speed Up* and good *scalability*. Able to achieve near constant runtime because of scaling data and nodes together.

Another stratergy *YAFIM(Yet Another Frequent Itemset Mining)* [18], used different approach for using parallel Apriori algorithm over *Spark RDD framework* in-spite of MapReduce. Spark is a specially designed for iterative and interactive algorithms of data mining. It is basically in-memory based parallel computing model means all data is loaded in memory itself. Secondly it do not use fixed two state model as in MapReduce but provide DAG based data flow. These features allow speeding up the computation significantly for the iterative algorithms like Apriori. YAFIM outperforms in terms of fast computation in comparison to MapReduce.

A new method called *NIMBLE* [19], aims to achieve portability of programming code, for fast and efficient implementation of the parallel data mining algorithms. The tool kit allows to build Machine learning and Data Mining algorithms around reusable building blocks parallely such that they can be easily utilized by other programming model as well. This helps to achieve inter-portability. NIMBLE facilitates the processing of variety of data formats due to its built-in support as well as simplify the custom data-format implementations. Strategies for optimization and abstraction incorporate hand in hand to deliver high performance runtime. It can be seen as a infrastructure, providing limited effort parallelization and Support for rapid prototyping.

All discussed approaches in Section 4.1, 4.2 and 4.3 can be summarized in the form of Table 2

5. Conclusion

Association rule mining aims to find the correlations and associations from the data. Association rules can be extracted by using either of the two primary approaches i.e. candidate-generation based (Apriori algorithm) or candidate-less (FP-Growth). The process of association rule mining can be further divided into two subtasks *frequent itemset generation* and *association rule generation*. Frequent itemset generation is computationally very intensive. Hence, parallelization of traditional approaches becomes inevitable. Association rule mining techniques can be parallelized in two ways i.e. data parallelization or algorithmic parallelization. Based on the literature review done it can be concluded that most of the research work has been done using data parallelization approach but, not much research work is being done on algorithmic parallelization. Parallelization of association rule mining techniques over distributed architecture helps us to achieve scalability and computational speed-up. But, there are many limitations such as data partitioning problems, uneven data distribution, improper load balancing, data communication overhead etc. MapReduce, a parallel programming model is a new concept which seemingly provides high computational power hence, promises a great scope to evolve association rule mining techniques in parallel environment.

References

[1] Le Zhou, Zhiyong Zhong, Jin Chang, Junjie Li, Huang, J.Z. and Shengzhong Feng, ”Balanced parallel FP-Growth with MapReduce,” in IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT), pp.243-246, 2010.

[2] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang and Edward Chang, ”PFP: Parallel Fp- Growth For Query Recommendation,” in Proceedings of ACM conference on Recommender systems (RecSys), 2008.

[3] Farzanyar, Z. and Cercone, N., ”Efficient mining of frequent itemsets in social network data based on MapReduce framework,” in EEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp.1183,1188, 25-28 Aug. 2013.

[4] Apiletti, D., Baralis, E., Cerquitelli, T., Chiusano, S. and Grimaudo, L., ”SeaRum: A Cloud-Based Service for Association Rule Mining,” in 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp.1283,1290, 16-18 July 2013.

[5] http://www.slideshare.net/zafarjcp/data-mining-association-rules-basics.

[6] Riondato, Matteo, Justin A. DeBrabant, Rodrigo Fonseca and Eli Upfal, ”PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce,” In Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 85-94, 2012.

[7] Lingjuan Li and Min Zhang, ”The Strategy of Mining Association Rule Based on Cloud Computing,” in International Conference on Business Computing and Global Informatization (BCGIN), pp.475-478, 29-31 July 2011.

[8] Xin Yue Yang, Liu, Zhen and Yan Fu, ”MapReduce as a programming model for association rules algorithm on Hadoop,” in 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), pp.99-102, 23-25 June 2010.

[9] Othman Yahya, Osman Hegazy and Ehab Ezat,”An Efficient Implementation of Apriori Algorithm Based on Hadoop-Mapreduce Model,” in Proccedings of the International Journal of Reviews in Computing, Vol. 12, pp.59-67, 31st December 2012.

[10] Zaiane, O.R., El-Hajj, M. and Lu, P., ”Fast parallel association rule mining without candidacy generation,” in Proceedings IEEE International Conference on Data Mining ICDM, pp.665-668, 2001.

[11] 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., pp. 1275-1285, 2007.

[12] A. Ghoting, G. Buehrer, S. Parthasarathy, D. Kim; A. Nguyen, Y.-K. Chen, and P. Dubey, Cache-conscious frequent pattern mining on modern and emerging processors, in journal of the VLDB , vol. 16, pp. 77-96, 2007.

[13] M. El-Hajj and O. R. Zaane, Parallel bifold: Large-scale parallel pattern mining with constraints, in journal of Distributed and Parallel Databases, vol. 20, no. 3, pp. 225-243, 2006.

[14] J. Dean and S. Ghemawat, Mapreduce: simplified data processing on large clusters, in Community of ACM, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[15] Aouad, Lamine M., Nhien-An Le-Khac, and Tahar M. Kechadi. ”Distributed frequent itemsets mining in heterogeneous platforms.” in Journal of Engineering, Computing and Architecture Vol.1, no. 2, 2007.

[16] Wang Yong, Zhang Zhe and Wang Fang, ”A parallel algorithm of association rules based on cloud computing,” in 8th International ICST Conference on Communications and Networking in China (CHINACOM), pp.415-419, 14-16 Aug. 2013.

[17] Xiaoting Wei, Yunlong Ma, Feng Zhang, Min Liu and Weiming Shen, ”Incremental FP-Growth mining strategy for dynamic threshold value and database based on MapReduce,” in Proceedings of IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp.271-276, 21-23 May 2014.

[18] Hongjian Qiu, Rong Gu, Chunfeng Yuan and Yihua Huang, ”YAFIM: A Parallel Frequent Itemset Mining Algorithm with Spark,” in IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW), pp.1664-1671, 19-23 May 2014.

[19] Ghoting, Amol and Kambadur, Prabhanjan and Pednault, Edwin and Kannan, Ramakrishnan, ” NIMBLE: A Toolkit for the Implementation of Parallel Data Mining and Machine Learning Algorithms on Mapreduce,” in Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 334-342, 2011.

[20] J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation, in SIGMOD, pp. 1-12, 2000.

[21] Agrawal, Rakesh, and Ramakrishnan Srikant. ”Fast algorithms for mining association rules.” In Proccedings of 20th international conference on very large data bases, VLDB, vol.12-15, pp. 487-499, 1994.

[22] Geng, Xia, and Zhi Yang. ”Data mining in cloud computing.” In International Conference on Information Science and Computer Applications (ISCA 2013), 2013.

[23] X. Y. Yang, Z. Liu and Y. Fu, MapReduce as a Programming Model for Association Rules Algorithm on Hadoop, in Proceedings 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), vol. 99, no. 102, pp. 23-25, 2010.