A Comprehensive Survey on Scalable Frequent Itemsets Mining

Ashin M
M.tech Student, Computer Science and Engineering
NSS College of Engineering
Palakkad, Kerala, India
findashinappus@gmail.com

Sruthy Manmadhan
Assistant Professor, Computer Science and Engineering
NSS College of Engineering
Palakkad, Kerala, India
sruthym.88@gmail.com

***Abstract*—Data mining is a process of extracting useful and nontrivial information from databases. Frequent Itemset Mining is one of the best known and most popular technique in data mining. Frequent itemsets are those items which are occurred frequently in the transaction database. Frequent Itemset mining is widely used in many industries such as financial, retail and telecommunication. The main objective of these industries is faster processing of massive amount of dataset. Fast and scalable frequent itemset mining techniques are becoming more important. The basic concepts of the existing Frequent Itemset Mining algorithms are divided into two categories: Apriori and FP-growth. To improve the efficiency of these algorithms, several enhancement techniques to enhance algorithms and partitioning techniques are developed. This survey provides a brief description about the algorithm enhancement techniques divided into three: Apriori based FIM techniques, FP-growth based FIM techniques and other FIM techniques. Finally gives some of the partitioning strategies for the improvement of parallel Frequent Itemset Mining.**

***Index Terms*—Data mining, Frequent itemsets mining.**

I. Introduction

Data mining is the extraction of hidden predictive information from large datasets. It is one of the new powerful technology with a great potential to help many companies to focus on the most important information in their data warehouses. The overall important goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. The automated, prospective analyses reffered by data mining methods move beyond the analyses of past events by retrospective tools typical of decision support systems [1].

Data mining tools can efficiently answer the business questions that traditionally were too time consuming process to resolve. The real data mining process is the semi-automatic or automatic analysis of massive data to extract previously unknown one, interesting patterns such as groups of data records, unusual records, and dependencies.

The discovery of frequent itemsets is one of the very important topics in data mining. Frequent itemset discovery techniques help in generating qualitative knowledge which gives business insight and helps the decision makers. In the Big Data era the need for a customizable algorithm [2] to work with big data sets in a reasonable time becomes a necessity.

Since recent developments (in technology, science, businesses, etc.) gave rise to the production and storage of massive amounts of data, not surprisingly, the intelligent analysis of big data has become more important for both businesses and academics. Frequent itemsets mining (FIM) is a core problem in association rule mining (ARM), sequence mining, and the like, frequent itemset mining tries to extract information from databases based on frequently occurring events, i.e., a set of events, is interesting if it occurs frequently in the dataset, according to a user given minimum frequency threshold. Speeding up the process of Frequent itemset mining is a very critical issue, because FIM consumption accounts for a major portion of mining time due to its high computation and input/ output (I/O).

Over the past decade a variety of algorithms have been developed that address FIM issues through the refinement of search strategies, pruning techniques [3], data structures, and the use of alternative dataset organizations. While most algorithms focus on the explicit discovery of all inferences for a given dataset by incorporating domain knowledge. Section 2 provides a brief introduction about the basic algorithms of frequent itemsets mining. And section 3 discusses about various FIM techniques and finally section 4 concludes with overall summary.

II. Preliminaries

A. Apriori Algorithm

Apriori algorithm [4] is, the most typical and significant algorithm for mining frequent itemsets. Apriori is used to discover all frequent itemsets in a given database. It uses a level wise search, where n-itemsets are used to explore (n+1)-itemsets, to mine frequent itemsets from transactional database for Boolean association rules. An itemset that contains n items is an n-itemset. Reducing the search space to avoid finding of each Ln requires one full scan of the database. If an itemset I does not satisfy the minimum support threshold, min-sup, the Iist is not frequent, that is, P (I) ¡minimum-sup. If an item A is added to the itemset called to be I, then the resulting itemset not occur more frequently than in itemst I. Therefore, I $\cup$ A is not frequent , that is, P (I $\cup$ A) $<$ minimum-sup.

First, the set of frequent 1-itemsets is found. This set is denoted L_1. L_1 is used to find L_2, the frequent 2-itemsets,

which is used to find L_3, and so on, until no more frequent k-itemsets can be found. Finding of each Ln requires one full scan of the database. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used. Apriori property means "All non-empty subsets of a frequent itemset should be also frequent". This is made possible because of the anti-monotone property of the support measure the support for an itemset never exceeds the support for its subsets.

The memory requirements for handling the complete set of candidate itemsets blows up fast and renders Apriori based schemes very inefficient to use on single machines [5]. Secondly, current approaches tend to keep the output and runtime under control by increasing the minimum frequency threshold, automatically reducing the number of candidate and frequent itemsets.

B. FP-Growth Algorithm

Fp-growth is also called as projected database based data mining techniques [6], which generates frequent itemset without candidate generation. It uses tree based structure. The problem of Apriori algorithm was distributed with, by introducing a different data structure, called frequent pattern tree, or FP-tree, then based on this structure an FP-tree created pattern fragment growth method was established. It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support.

The biggest problem of FP-growth is the interdependency of data. The interdependency problem is that, for the parallelization of the algorithm some that still needs to be shared, which creates a bottleneck in the shared memory.

III. Improved FIM Techniques

There are many frequent itemset mining strategies are exist for efficient mining of data. We can improve the FIM technique either by enhancing the mining algorithm or by improving the partitioning strategies of transaction database. For the enhancement of algorithm there are several apriori or FP-growth based strategies and other FIM techniques except apriori and FP growth.

A. Apriori Based FIM Techniques

One of the most important challenges for data mining is quickly and correctly finding the relationship among data. The Apriori algorithm has been the most popular technique in finding frequent patterns. However, when applying this method, a database has to be scanned many times to calculate the counts of a huge number of candidate itemsets. Several FIM techniques are introduced by enhancing the classic apriori algorithm.

High-Dimension Oriented Apriori algorithm [7] (HDO Apriori) is an improved version of apriori for mining the association rules in the high-dimensional data. This algorithm can cut down the redundant generation of identical sub-itemsets from candidate itemsets, by means of pruning the candidate itemsets with the infrequent itemsets with lower dimension.

Another new apriori enhancement policy [8] introduced in 2010, which is based on the user interest and the importance of itemsets is put forward by the paper, incorporate item that user is interested in into the itemsets as a seed item, then scan the database, incorporate all other items which are in the same transaction into item sets and then construct user interest itemsets and reduce unnecessary itemsets through the design of the support functions algorithm not only considered the itemset's frequency, but also consider different importance between different itemsets. This new algorithm reduces the storage space, also improves the efficiency and accuracy of the algorithm.

After, a new policy Distributed Parallel Apriori [9] (DPA) algorithm is proposed as a solution to accelerate the mining process in Parallel and distributed computing . In this method, metadata are stored in the form of Transaction Identifiers (TIDs), such that only a single scan to the database is needed. The approach also takes the factor of itemset counts into consideration and thus generating a balanced workload among processors and reducing processor idle time.

DPA can thus effectively reduce the required scan iterations to a database and accelerate the calculation of itemsets. It also adopts a useful heuristic to partition the itemsets to processors.

An enhancement to the Apriori algorithm is introduced to reduce the time consuming for candidates itemset generation [10]. Firstly scan all transactions to generate L_1 which contains the items, their support count and Transaction ID where the items are found. And then use L_1 later as a helper to generate L_2, L_3 ... L_k. For generate C_2, it make a self-join $L_1 * L_1$ to construct 2-itemset C (x, y), where x and y are the items of C_2.

Before scanning all transaction records in the dataset to count the support of each candidate generated, use L_1 to get the transaction IDs of the minimum support count between x and y, and then scan for C_2 only in these specific transactions. Then the same thing for C_3, construct a 3-itemset as C (x, y, z), where x, y and z are the items of C_3 and use L_1 to get the transaction IDs of the minimum support count between x, y and z, thus scan for C_3 only in these specific transactions and repeat these procedure until no new frequent itemsets are identified.

Another strategy based on Prefixed-itemset-based data structure [11] for candidate itemset generation is developed. With the help of this structure, it is used to improve the efficiency of the classical Apriori algorithm. There are mainly three schemes,

- Prefixed-itemset-based storage
- Prefixed-itemset-based connecting step
- Prefixed-itemset-based pruning step

With the help of prefixed-itemset-based data storage, it should be managed to finish the connecting step and the pruning step of the Apriori algorithm much faster, besides that, it can store the candidate itemsets with smaller storage space.

In order to scale to truly big datasets one has to find a way to parallelize them. As a solution to this, a new algorithm Apriori SON algorithm [12] is developed, which makes use of an interesting property to partition the database of transactions into several chunks, which can be executed in parallel.

On each chunk, some worker thread will run whatever method to find all frequent itemsets limited to it. The property guarantees that if an itemset is frequent in database D, then it will be frequent in at least some chunck, so there will be no false negatives. After every worker has finished, all local itemsets are gathered by a master, and the real support of each one is counted (can be done in parallel). This second pass removes any false positives that might have been reported by a worker.

B. FP-Growth Based FIM Techniques

FP-Growth algorithm run much faster than the Apriori, as it is logical to parallelize the FP-Growth algorithm to enjoy even faster mining process.

A MapReduce approach of parallel FP-Growth algorithm [13] developed, which intelligently shards a large-scale mining task into independent computational tasks and maps them onto MapReduce jobs. PFP uses three MapReduce [14] phases to parallelize PF-Growth. PFP can achieve near-linear speedup with capability of restarting from computer failures. Five steps of PFP is given below.

- Sharding: Division and distribution of data.
- Parallel Counting: Count the support values of all items that appear in DB.
- Grouping Items: Dividing all the items on F-List into Q groups.
- Parallel FP-Growth
  - Mapper: Generating group-dependent transactions.
  - Reducer: FP-Growth on group-dependent shards.

A balanced parallel FP-Growth algorithm BPFP [15], is based on the PFP algorithm, which parallelizes FP-Growth in the MapReduce approach. BPFP adds into PFP load balance feature, which improves parallelization and thereby improves performance. BPFP uses two rounds of MapReduce to parallelize FP-Growth. In BPFP, the grouping step has a balanced strategy, which divides the entire mining task into relatively even subtasks to improve parallelization.

Later an improved FP-Growth(IFP) [16] algorithm is proposed, which firstly combine the same patterns based on the situation whether the support of the transaction is greater than the minimum support to mine the frequent patterns. It uses the depth-first method mining the frequent itemsets and saves a great deal of space. Thus the improved FP-Growth cuts down on the space and it improves the efficiency of mining process. Further more, combine the improved FP-Growth with the MapReduce computing model, named as MR-IFP (MapReduce-Improved FP), to improve the capability to deal with the mass amount of data.

Later an optimized schema by modifying head table and FP-tree [17] data structure is introduced. Enhancing the FP-growth algorithm by improving the data structure reduce the time complexity and the judgments of a single path reduce the cost of space-time, which results from the recursive operation.

An implementation of scalable parallel FP-Growth(S-FPG) [18] using the in memory parallel computing framework Apache Spark is introduced. Parallelising FP-growth in spark employs a unique search strategy using compact structures resulting in a high performance algorithm that requires only two database passes. In addition, there is no need to use k-frequent itemsets iteratively to generate (k+1)-frequent itemsets.

A tree-partition algorithm [19] for parallel mining of frequent patterns is developed based on FP-Growth algorithm, which is constituted of tree-building stage and mining stage. The main idea is to build only one FP-Tree in the memory, partition it into several independent parts and distribute them into different threads. A heuristic algorithm is devised to balance the workload. This algorithm successfully avoided the overhead introduced when building multi-trees and the tree partition algorithm can minimized the lock overhead and make the workload perfectly balanced.

C. Other FIM Techniques

Other than apriori or FP-growth based FIM techniques, there are several frequent itemset mining strategies are developed which provides much flexibility and efficiency in the mining of frequent itemset from the databases.

Mining frequent closed itemsets within sliding window instead of complete set of frequent itemset is very interesting since it needs a limited amount of memory and processing power. A new approach named TMoment [20] for mining all closed frequent itemset within sliding window over data streams. A novel data structure for storing transactions of the window and corresponding frequent closed itemsets. Moreover, the support of a new frequent closed itemset is efficiently computed and an old pattern is removed from the monitoring set when it is no longer frequent closed itemset. It uses a lower memory and requires smaller time. Therefore, it is suitable for high speed and unbounded transactional data streams.

Another new structure of efficient frequent pattern mining based on linear prefix tree [21] is proposed. The LP-growth is constructed the LP-tree with the help of the algorithm. The LP- growth is used to apply in the mining process. The main intention of this algorithm is to reduce the memory usage but the time was increased. LP-tree contributed for improving the performance of frequent pattern mining so it used only the less memory and it was accessed in many cases, such as applications. The algorithm is not only used in the general frequency pattern mining, it also used in the variety of pattern mining fields such as closed or maximal pattern mining.

Later FIUT [22], A new method for mining frequent itemsets is developed. This method proposes a productive technique, the frequent itemset ultrametric trees (FIUT), for mining incessant itemsets in a database. FIUT utilizes an extraordinary continuous things ultrametric tree (FIU-tree) structure to improve its proficiency in getting continuous itemsets.

FIUT has four major advantages. To start with, it minimizes I/O overhead by examining the database just twice. Second, the FIU-tree is an enhanced approach to segment a database, which results from grouping exchanges, and fundamentally diminishes the inquiry space. Third, just incessant things in every exchange are embedded into the FIU-tree for fully packed storage. At long last, all successive itemsets are produced by checking the leaves of each FIU-tree, without crossing the tree recursively, which altogether decreases processing time. FIUT was contrasted and FP development, an understood and broadly utilized algorithm and the reenactment results demonstrated that the FIUT beats the FP-development.

To improve data storage efficiency and to avoid building conditional pattern bases, FiDoop [23] incorporates the concept of FIU-tree rather than traditional FP trees. By parallelizing the serial algorithm FIUT [22] as three mapreduce jobs. It includes two major phases:

- The first phase of FIUT involving two rounds of scanning a database is implemented in the form of two MapReduce jobs. The first MapReduce job is responsible for the first round of scanning of transaction database to create frequent one-itemsets. Then the second MapReduce job scans the database again to generate k-itemsets by removing infrequent items in each transaction.
- The second phase of FIUT involving the construction of a k-FIU tree and the discovery of frequent k-itemsets is handled by a third MapReduce job.

The third MapReduce job plays an important role in parallel mining; its mappers independently decompose itemsets whereas its reducers construct small ultrametric trees to be separately mined.

D. Partition enhanced FIM Techniques

Before applying a mining algorithm to a distributed database, the data should be partitioned efficiently. Various default partitioning techniques are present for partitioning of the data in distributed database. Other than default partitioning concept, we can partition the database as per the requirement and to improve the efficiency of the mining process. Several partitioning strategies are developed for the partitioning of the database to improve the mining process.

A hierarchical partitioning approach [24] is developed for the mining frequent itemsets from large databases. It is based on a novel data structure called the Frequent Pattern List (FPL). One of the important features of the Frequent Pattern List is the ability to partition the dataset, and thus convert the database into a set of sub-databases of recent sizes, That is, it partitions the search space and solution space and thus reducing the database into sub-databases of manageable sizes. As a result, a divide-and-conquer approach can be developed to perform the desired data-mining tasks.

Since the subdatabases in hierarchical partitioning contain only frequent items, re-scanning the database to check for minimum support threshold is not required as in flat partitioning. Therefore, hierarchical partitioning can improve the speed in mining frequent itemsets in large databases.

To minimize the transferred data between mappers and reducers in the shuffle phase of MapReduce a new technique is proposed, that repartitions tuples [25] of the input datasets, and optimizes the distribution of key-values over mappers, and increases the data locality in reduce tasks. It finds the relationships between input tuples and intermediate keys and assigns input tuples to the appropriate fragments in such a way that these MapReduce jobs follows the modeled workload will take the full advantage of data locality in the reduce phase. For finding the workload, there is a monitoring component in the MapReduce framework for producing the required metadata.

Another component, which is executed offline, combines the information captured for all the MapReduce jobs of the workload and partitions the input data accordingly. Workload is modeled by means of a hypergraph, to which a min-cut k-way graph partitioning algorithm applied to assign the tuples to the input fragments.

Later a highly scalable, parallel frequent itemset mining (PFIM) [26] algorithm, namely Parallel Absolute Top Down (PATD) is developed. PATD algorithm renders the mining process of very large databases simple and compact. Its mining process is made up of the only one parallel job, which considerably reduces the mining runtime, the communication cost and the energy power consumption overhead, in a distributed computational platform. Based on a clever and efficient data partitioning strategy, namely Item Based Data Partitioning (IBDP). IBDP allows for an optimized data placement on MapReduce. It allows PATD algorithm to exhaustively and quickly mine very large databases. PATD algorithm mines each data partition independently, relying on an absolute minimum support instead of a relative one.

Another strategy, Partition Enhanced Mining Algorithm (PEMA) [27] is a solution to high response times, high communication costs and inability to adapt to the constantly changing databases. In Partition Enhanced Mining Algorithm, first the Association Rule Mining Coordinating Agent receives a request and then decides the appropriate data sites, after that partitioning strategy and finally mining agents to use.

The mining process is divided into two stages. In the first stage, the data agents horizontally segment the databases with small average transaction length into relatively smaller partitions based on the number of available sites and the available memory. On the other hand, databases with relatively large average transaction length were vertically partitioned. After this, Mobile Agent-Based Association Rule Mining-Agents, which are the mining agents, carry out the discovery of the local frequent itemsets. Then at the next stage, the local frequent itemsets were integrated from the one data site to another to get the global frequent itemsets. This reduces the response time and communication cost in the system.

As a solution to existing techniques that suffer high communication and mining overhead induced by redundant transactions transmitted among computing nodes. A new data partitioning approach called FiDoop-DP [28] using the MapReduce programming model is then introducec in 2017.

The main goal of FiDoop-DP is to boost the performance

of parallel Frequent Itemset Mining on Hadoop clusters. The heart of FiDoop-DP is the Voronoi diagram-based data partitioning technique, which exploits correlations among transactions. By incorporating the similarity metric(Jaccard Coefficient) and the Locality-Sensitive Hashing technique, FiDoop-DP places highly similar transactions into a data partition to improve locality without creating an huge number of redundant transactions.

CONCLUSION

Efficient data mining algorithms for mining frequent itemsets are crucial for mining association rules. The main aim is to optimize the process of finding frequent itemsets which should be efficient, scalable and can detect the important itemsets by enhancing algorithm and partitioning the transaction database. In this paper, initially discussed about the various existing classic algorithms for Frequent Itemsets Mining such as apriori and FP-growth and then reviewed many improved frequent itemset mining strategies.

In case of apriori based techniques, it requires high computational time and cost when compared to FP-Growth. It is logical to parallelize FP-Growth algorithm, thus it enhance the mining task better than apriori based strategies. Also by improving the partitioning strategy of the database, we can improve the parallel mining process by reducing the time consumption and minimizing the data transfer.

REFERENCES

[1] R. Agrawal and J. C. Shafer, Parallel mining of association rules, IEEE Trans. Knowl. Data Eng., vol. 8, no. 6, pp. 962-969, Dec. 1996.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo, "Fast discovery of association rules," Advances in Knowledge Discovery and Data Mining, pages. 307328. MIT Press, 1996.

[3] Mueller A, "Fast sequential and parallel algorithms for association rule mining: A comparison", Technical Report CS-TR-[10] 3515, 1995.

[4] Agrawal R, Srikant R, "Fast algorithms for mining association rules", In Proceedings of the 20th VLDB conference,pp.487- 499, 1994.

[5] Ashok Kumar D and Loraine Charlet Annie M. C, "A Review on Frequent Itemset Mining based on Partitioning Approach", International Journal of Computer Science And Technology, 2012.

[6] J. Han, H. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation", In: Proc. Conf. on the Management of Data (SIGMOD00, Dallas, TX). ACM Press, New York, NY, USA 2000.

[7] Lei Ji, Baowen Zhang and Jianhua Li, "A New Improvement on Apriori Algorithm", International Conference on Computational Intelligence and Security, 2006.

[8] Du Ping and Gao Yongping, "A New Improvement of Apriori Algorithm for Mining Association Rules", International Conference on Computer Application and System Modeling, 2010.

[9] Kun-Ming Yu, Jiayi Zhou, Tzung-Pei Hong and Jia-Ling Zhou, "A load-balanced distributed parallel mining algorithm", Journal on Expert Systems with Applications, 2010.

[10] Mohammed Al-Maolegi and Bassam Arkok, "An Improved AprioriI Algorithm For Associatiom Rules", International Journal on Natural Language Computing (IJNLC) Vol. 3, No.1, February 2014.

[11] Yu Shoujian and Zhou Yiyang, "A Prefixed-Itemset-Based Improvement For Apriori Algorithm", Journal on CoRR, 2016.

[12] Andrea Galloni and Daniel Bruzual, "Mining Frequent Itemsets in Apache Spark", Journal on Big Data and Social Networks,2016.

[13] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang,PFP: Parallel FP-growth for query recommendation, in Proc. ACM Conf. Recommend. Syst., Lausanne, Switzerland, 2008, pp. 107114.

[14] J. Dean and S. Ghemawat,MapReduce: Simplified data processing on large clusters, Commun. ACM, vol. 51, no. 1, pp. 107113, Jan. 2008.

[15] Le Zhou, Zhiyong Zhong, Jin Chang, Junjie Li, Joshua Zhexue Huang and Shengzhong Feng, "Balanced Parallel FP-Growth with MapReduce", IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT),2010.

[16] Sun Hong, Zhang Huaxuan, Chen Shiping and Hu Chunyan, "The Study of Improved FP-Growth Algorithm in MapReduce", Proceedings of the The 1st International Workshop on Cloud Computing and Information Security, 2013.

[17] DENG Lingling and LOU Yuansheng, "Improvement and research of FP-growth algorithm based on distributed spark", International Conference on Cloud Computing and Big Data, 2015.

[18] Aissatou Diaby dite Gassama, Fode Camara and Samba Ndiaye, "S-FPG: A Parallel Version of FP-Growth Algorithm under Apache Spark", 2nd IEEE International Conference on Cloud Computing and Big Data Analysis, 2017.

[19] D. Chen, Tree partition based parallel frequent pattern mining on shared memory systems, in Proc. 20th IEEE Int. Parallel Distrib. Process. Symp. (IPDPS), Rhodes Island, Greece, pp. 18 2006.

[20] Fatemeh Noria, Mahmood Deypirb and Mohamad Hadi Sadreddinia, "A sliding window based algorithm for frequent closed itemset mining over data streams", The Journal of Systems and Software, 2012.

[21] Gwangbum Pyun, Unil Yun and Keun Ho Ryu, "Efficient frequent pattern mining based on Linear Prefix tree", Journal on Knowledge-Based Systems, vol. 55, pp. 125-139, January, 2014.

[22] 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. 17241737, 2009.

[23] Yaling Xun, Jifu Zhang, and Xiao Qin, "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce," IEEE Transactions on system, man, and cybernetics systems,2016.

[24] Fan-ChenTseng, "Mining frequent itemsets in large databases: The hierarchical partitioning approach", Expert Systems with Applications, vol. 40, Issue. 5, pp. 1654-1661, April 2013.

[25] Miguel Liroz-Gistau, Reza Akbarinia, Divyakant Agrawal, Esther Pacitti and Patrick Valduriez, "Data Partitioning for Minimizing Transferred Data in MapReduce", International Conference on Data Management in Cloud, Grid and P2P Systems, 2013.

[26] Saber Salah, Reza Akbarinia, and Florent Masseglia, "Data Partitioning for Fast Mining of Frequent Itemsets in Massively Distributed Environments", 26th International Conference on Database and Expert Systems Applications, vol. 9261,pp. 303-318, 2015.

[27] A.O.Ogundea, Folorunso and A.S.Sodiya, "A partition enhanced mining algorithm for distributed association rule mining systems", Egyptian Informatics Journal vol. 16, Issue 3, pp. 297-307, November 2015.

[28] Yaling Xun, Jifu Zhang, Xujun Zhao and Xiao Qin, "FiDoop-DP: Data Partitioning in Frequent Itemset Mining On Hadoop Clusters", IEEE Transactions on Parallel and Distributed Systems, vol. 28, Issue. 1, pp. 101 - 114, 2017.