

Efficient Association Rule Mining For Discovery of Frequent Item Sets on Big Data Sets

Abstract— Knowledge discovery and decision making demands the ability to process and extract insights from “Big” Data in a scalable and efficient manner. Data mining is the function of refined analysis to large quantity of data in order to determine new knowledge in the outline of patterns, trends, and relations. With the introduction of World Wide Web, the sum of data stored and available electronically has grown-up enormously and the method of knowledge innovation from this data has turn out to be very significant for the business and scientific research communities. Frequent Itemset Mining is one of the most extensively acknowledged procedures to dig out knowledge from data. The combinatorial outburst of Frequent Itemset Mining methods becomes even additional challenging when they are pertained to Big Data. Providentially, recent developments in the area of parallel programming already offer outstanding tools to deal with this problem. Nevertheless, these tools move toward their own technical disputes, e.g. unbiased data sharing and inter-communication costs. In our paper, we examine the applicability of Frequent Itemset Mining method on the Map Reduce platform. We introduce new method for pulling out large datasets: Big-Frequent-Itemset-Mining is optimized to run on actually large datasets. Our approach is similar to FP-growth i.e top down but using different data structure based on algebraic topology. In our research we demonstrate the scalability of our techniques.

Index Terms— Data mining, frequent item sets, association rules, Big data Sets, Frequent Pattern Mining, Map Reduce.

I. INTRODUCTION

Since current progress in technology, science, user habits, businesses, etc amplified production and storage of gigantic quantity of data, not astonishingly, the clever analysis of big data has develop into more significant for equally businesses and academics. Previously Frequent Itemset Mining has been vital ingredient of data analysis and data mining. Frequent Item Set Mining attempt to pull out information from databases based on frequently taking place measures, i.e., an event, or a set of events, is attractive if it take place frequently in the data, according to a user specified minimum frequency threshold. Numerous methods have been made-up to extract databases for frequent events [5], [7], [31]. These procedures work well in practice on archetypal datasets, but they are not appropriate for really Big Data. Applying frequent Itemset mining to large databases is difficult. Predominantly very large databases do not fit into main memory. In these conditions, one elucidation is to exercise level wise breadth first search based algorithms, like reputable Apriori algorithm [5], where frequency accumulation is realized by reading the dataset over and over again for each size of candidate item sets. Unfortunately, the memory requirements for managing the total set of candidate item sets propel up fast and turn

into Apriori based schemes very incompetent to exploit on single machines. Recent approaches tend to keep the output and runtime under control by escalating the minimum frequency threshold, involuntarily dropping the number of candidate and frequent item sets. Conversely, research in recommendation systems has revealed that item sets with lower frequencies are more fascinating [22]. Hence, we still observe obvious need for methods that can contract with low frequency thresholds in Big Data. Parallel programming is becoming a necessity to deal with the massive amounts of data, which is produced and inspired more and more every day. Parallel programming architectures, and therefore the algorithms, can be clustered into two major subcategories: shared memory and distributed (share nothing). Considering shared memory systems, all processing units can in tandem use a shared memory area. Additionally, distributed systems are tranquil of processors that contain their individual internal memories and converse with each other by passing messages [6]. It is easier to adapt algorithms to shared memory parallelism in general, but they are typically not scalable enough [6]. The Map Reduce framework anticipated by Google [10], which make simpler the programming for distributed data dispensation, and the Hadoop execution by Apache Foundation [2], which provides the framework unreservedly accessible for one and all, distributed programming is fetching more and more accepted. Besides, current viable and non-viable systems and services advance the usability and accessibility for anyone. In this paper, we establish new algorithm that exploit the Map Reduce framework to compact with Big Data. Big-Frequent-Itemset-Mining is optimized to compact with strictly Big Data by using a hybrid algorithm on Map Reduce.

II. RELATED WORK

Literature in Data mining make use of parallel methods previously since its very early days [4], and many novel parallel mining methods as well as plans that parallelize existing sequential mining techniques exist. However, the number of algorithms that are adapted to the Map Reduce framework is rather limited. In this section we will give an overview of the data mining algorithms.

A. Overview of association analysis

Among the various data mining techniques, association analysis is a popular and well researched method for discovering interesting associations and/or relationships among variables in large databases. For a pattern to be interesting, it should be logical and actionable. Association

rule learning basically consists of revealing relationships among attribute values that occur frequently together in a dataset, and then representing them in the form of association rules. Association rules do not indicate causality, but instead suggest strong co occurrence relationships that can be further investigated as associated factors. Two important metrics in association analysis are support and confidence.

Support indicates how often a rule is applicable to a specific data set and can be used to eliminate uninteresting rules, such as those that occur simply by chance [10] [38]. Confidence measures the reliability of the inference made by a rule, for instance, $X \rightarrow Y$, measured as how frequently items or attributes in Y appear in transactions or patients that contain X . Minimum support and confidence thresholds are selected for assessing the association rules extracted from the data. An Itemset is frequent if its support is greater than or equal to that minimum support value. One important issue with mining association rules in large data sets is the fact that it can be computationally expensive depending on the algorithm used. A brute-force approach for discovering patterns from data would consist of computing the support and confidence for every possible rule. As the number of rules that can be obtained from a data set grows exponentially with the number of items in that set, this brute-force approach becomes prohibitively expensive. This approach also results in wasted transactions, as many of the rules would be discarded for falling below the minimum support and confidence levels selected.

B. Apriori and FP-Growth Algorithms

Many algorithms for generating association rules have been presented over time, such as the Apriori and FP-Growth algorithms [7] [40] [41]. A common strategy that these algorithms implement, in terms of performance improvement, is to decompose the problem into two subtasks:

- a) Frequent item set generation, accomplished by reducing either the number of:
 - i. candidate item sets based on the support measure, as in the Apriori algorithm, or
 - ii. Comparisons, as in the FP-Growth algorithm;
- b) rule generation, which first excludes rules that have empty antecedents or consequents and then checks that, after splitting item set Y into two non-empty subsets (X and $Y - X$), rule $X \rightarrow Y - X$ satisfies the confidence threshold. $Y - X$ in this case is known as the rule consequent. Rule generation does not require any additional passes over the dataset.

C. Data Mining Algorithms on Map Reduce

Lin et al. propose three algorithms that are adaptations of Apriori on Map Reduce [21]. These algorithms all distribute the dataset to mappers and do the counting step in parallel. Single Pass Counting (SPC) utilizes a Map Reduce phase for each candidate generation and frequency counting steps.

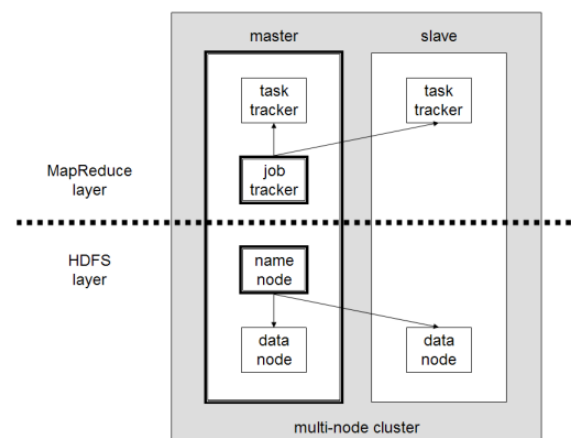
Fixed Passes Combined-Counting starts to generate candidates with n different lengths after p phases and counts their frequencies in one database scan, where n and p are given as parameters. Dynamic Passes Counting is similar to FPC, however n and p is determined dynamically at each phase by the number of generated candidates. The PApriori algorithm by Li et al. [20] works very similar to SPC, although they differ on minor implementation details. MRApriori [16] iteratively switches between vertical and horizontal database layouts to mine all frequent item sets. At each iteration the database is partitioned and distributed across mappers for frequency counting. Parallel FP-Growth [18] is a parallel version of the well-known FP-Growth [17]. PFP groups the items and distributes their conditional databases to the mappers. Each mapper builds its corresponding FP-tree and mines it independently. The PARMA algorithm by Riondato et al. [27] finds approximate collections of frequent item sets.

D. Hadoop Framework

Hadoop is a software framework for distributed processing of large datasets across large clusters of computers. HDFS was designed in the project NUTCH]. It acts as the storage mechanism. Input data are split into different chunks and stored in HDFS. The default chunk size is 64MB.

Hadoop framework consists on two main layers

- Distributed file system (HDFS)
- Execution engine (Map Reduce)



The following diagram shows Hadoop architecture.

Fig 2.1 Hadoop architecture

HDFS has block oriented architecture. Each block has fixed size and is stored in the hadoop cluster. These different blocks are called as data node and they contains the actual data. The data nodes are stored in different machines at different clusters. The data is processed in the same cluster were it is stored. HDFS has the master slave architecture. Each hadoop cluster contains a single name node, that is the master node and slave nodes are the data nodes. The primary communication mechanism between the name node and data node is called heartbeat. In every three seconds the heartbeat is sent to the name node from the data node. Heart beat

contain the block report and list of blocks in the data node. If the heartbeat is not received the name node will create another replica of the data node. The following diagram describes architecture of HDFS.

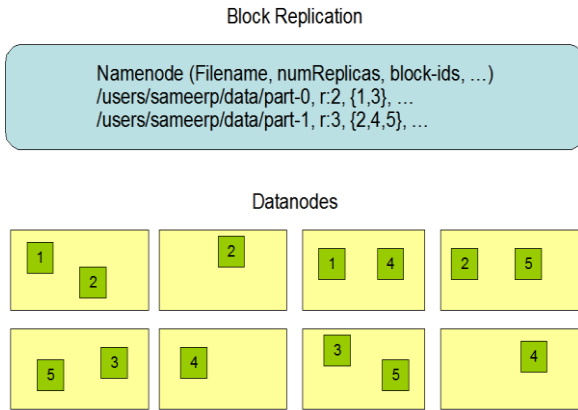


Fig 2.2 HDFS architecture

E. Map Reduce

The Map Reduce framework helps us to retrieve the data efficiently from large spectrum of data. The Map Reduce framework was introduced by Google in 2004. Map Reduce process the data as the key value pair. There are mainly four phases to perform the Map Reduce operation. First phase, mapper phase will collect the data from the HDFS which are stored in the different clusters. Output from the mapper phase is the intermediate results. These results are then given to other phase for passing to the reducer. Second phase is shuffle phase; here intermediate results are shuffled so that the results of the different mappers are brought together. Third phase is sort phase. In this the shuffled intermediate results are sorted together based on the key value, so the same key valued contents are brought together. By doing sorting the contents can be easily passed to the reducer for processing. Last phase is reducer phase. In this the sorted contents are processed to get the significant data. The jobs in the hadoop cluster are performed by the task tracker. When a job is scheduled, the job tracker will assign the job to the task tracker. It will proceed to the job execution and the output is produced. Thus the large range of data is processed into useful contents. The following diagram explains layers in Map- Reduce.

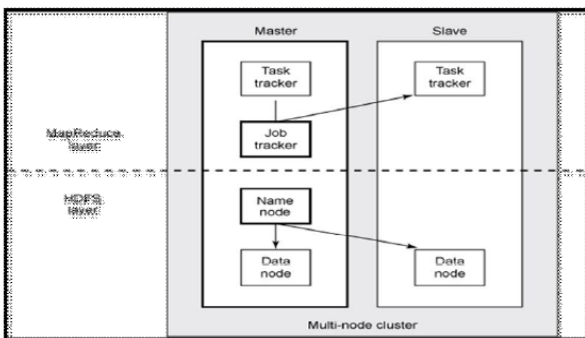


Fig 2.3 Layers in Map Reduce frame work

The functionality of Map Reduce can be explained by following diagram.

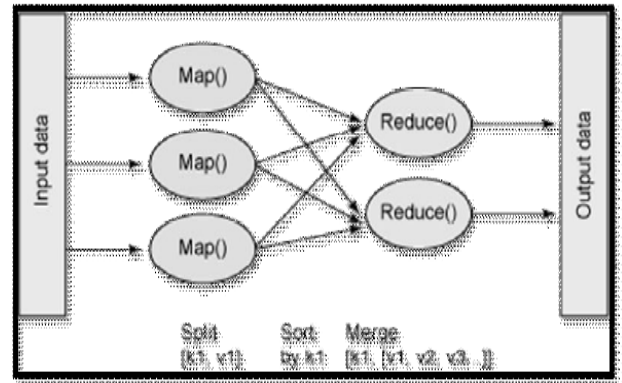


Fig 2.4 Functionality of Map Reduce

III. FREQUENT ITEM SET MINING ON MAPREDUCE

We propose two new methods for mining frequent item sets in parallel on the Map Reduce framework where frequency thresholds can be set low. We introduce a second method which is hybrid method that first uses an Apriori based method to extract frequent item sets of length k and later on switches to Eclat when the projected databases fit in memory. First, mining for k -FIs can already be infeasible. Indeed, in the worst case, one mapper needs the complete dataset to construct all 2-FIs pairs. Considering Big Data, the tid-list of even a single item may not fit into memory. Secondly, most of the mappers require the whole dataset in memory in order to mine the sub-trees.

The following are various steps involved in our new algorithm

1. **Generating k -FIs:** Big Frequency Item Set Mining covers the problem of large tid-lists by generating k -FIs using the breadth-first method. This can be achieved by adapting the Word Counting problem for documents, i.e., each mapper receives part of the database (a document) and reports the items/item sets (the words) for which we want to know the support (the count). A reducer combines all local frequencies and reports only the globally frequent items/item sets. These frequent item sets can be redistributed to all mappers to act as candidates for the next step of breadth-first search. These steps can be repeated k times, to obtain the set of k -FIs.

2. **Finding Potential Extensions:** After computing the prefixes, the next step is computing the possible extensions, i.e., obtaining tid-lists for $(k+1)$ -FIs. This can be done similar to Word Counting, however; now instead of local support counts we report the local tid-lists. A reducer combines the local tid-lists from all mappers to a single global tid-list and assigns complete prefix groups to different mappers.

3. **Subtree Mining:** Finally, the mappers work on individual prefix groups. A prefix group defines a conditional database that completely fits into memory. The mining part then utilizes diffsets to mine the conditional database for frequent item sets using depth-first search. we

can continue the iterative process until we reach a set of k-FIs that are small enough.

In our methods frequent item sets are mined in step 3 by the mappers and then communicated to the reducer. To reduce network traffic, we encoded the mined item sets using a compressed trie string representation for each batch of patterns.

The new algorithm can be explained for mining frequent item sets from big Data sets is an improvement to the original FPTree algorithm and described as follows.

Algorithm BigFPTree ():

Input: Big Data Set of Transactions
Support

Output: Frequent Patterns

Begin

ConstructHeaderTable ()

FindOneFrequentItemSets ()

ProcessTransactionSets ()

End

Algorithm ConstructHeaderTable ()

Begin

Scan Dataset

Count support of each item

Construct Header Table (TH) by using Items and their support
(Header Table Consists 3 fields name, support and link)

Link refers to all nodes of an item on Tree

End

Algorithm FindOneFrequentItemSets ()

Begin

While (TH Not empty)

Do

Remove Items with support less than min support from TH
Sort the TH based on support in descending order

Done

End

Algorithm ProcessTransactionSets ()

Begin

Scan the Data set to create Transaction Item Sets (TIS)

Remove the non frequent Items from TIS

Sort TIS by order of Items in TH

Construct Tree (T) with Ordered Item Sets

Add new nodes to link field of TH

ProcessItem ()

End

Algorithm ProcessItem ()

Begin

Add Item Q into Base Item (BI)

In TH, Q.link contains all Nodes in Tree T whose Item is Q

Read all Items from Node N_i , $i = 1$ to k to root of T

Create SubHeaderTable (SHT) with items and support

While (SHT Not empty)

Do

Remove items from SHT with support less than min support
Sort SHT on support in descending order

Done

Read all Items from N_i to root

Remove non local frequent Item Sets

Sort Item Sets by SHT

Construct new Sub Tree with sorted Item Sets

Add s to support keep all new nodes in link of SHT
End

The following flowchart explains the process of finding frequent Item Sets on Big Data Sets

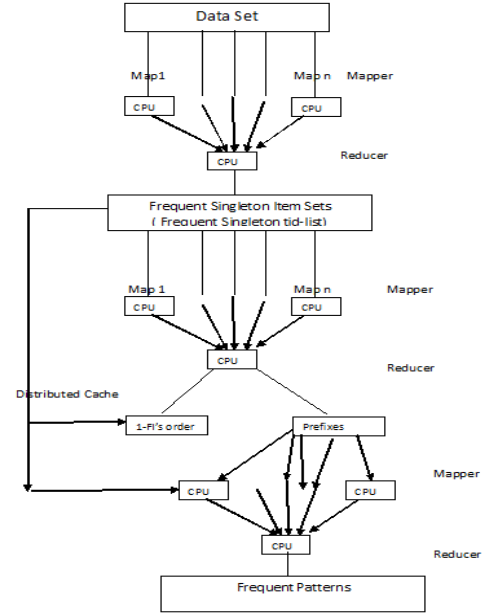


Fig 3.1 Process for finding frequent Item Sets on Big Data Sets

IV. EXPERIMENT RESULTS AND ANALYSIS

We analyze the running time of algorithms on both synthetic and real datasets. Our experiments suggest independence of the assignment method: using longer prefixes results in a better balancing of the load. Generating the longer prefixes requires additional initial computation. However, for large databases this computation is negligible compared to the entire mining process. Here we consider two data sets of different sizes and evaluated generation of association rules based on different parameters.

The following diagram shows the relation between support and confidence in generating association rules.

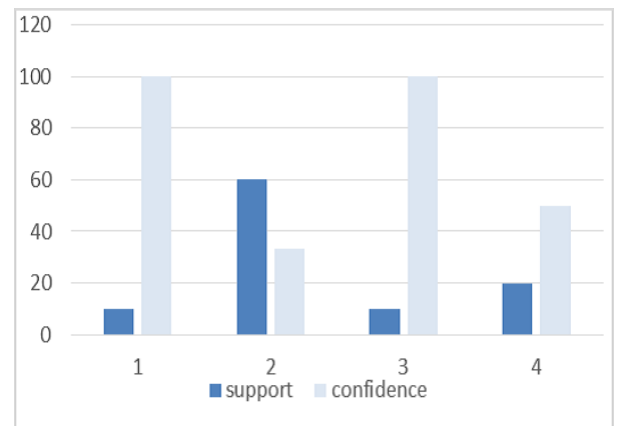


Fig 4.1 Support and Confidence

The following diagram shows the Mapper and Reducer execution.

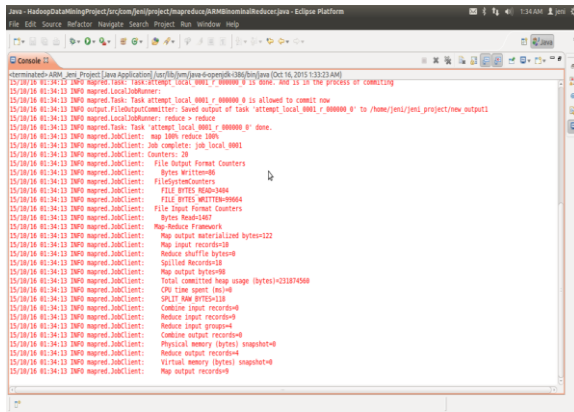


Fig4.2 Mapper and Reducer execution

The Figures 4.3 and 4.4 shows how execution time varies for different no of association rules for data set1 and data set2

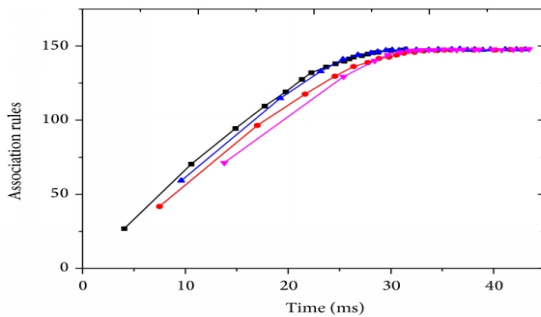


Fig 4.3 No of Association Rules and time taken for execution from dataset1

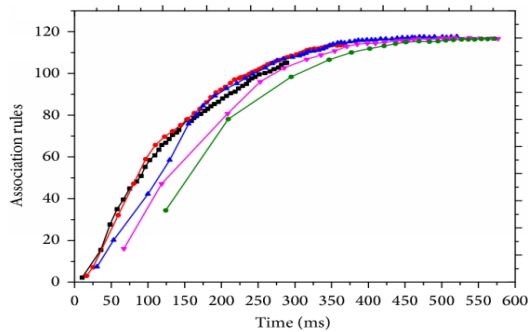


Fig 4.4 No of Association Rules and time taken for execution from dataset2

The Figures 4.5 and 4.6 No of association Rules generated from data set1 and data set2

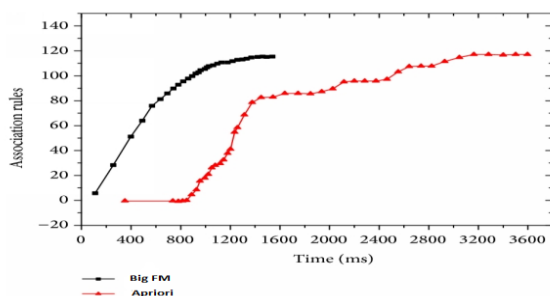


Fig 4.5 No of Association Rules mined from data set1

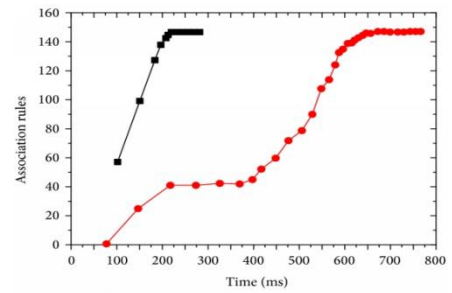


Fig 4.6 No of Association Rules mined from data set2
The following diagrams shows efficiency of BigFM and Apriori algorithms on data set1 and data set2 with different min support and min confidence values.

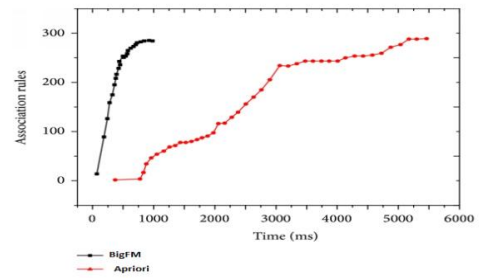


Fig 4.7 Mining efficiencies of BigFM and Apriori algorithm with different min_sup and min_conf in dataset1

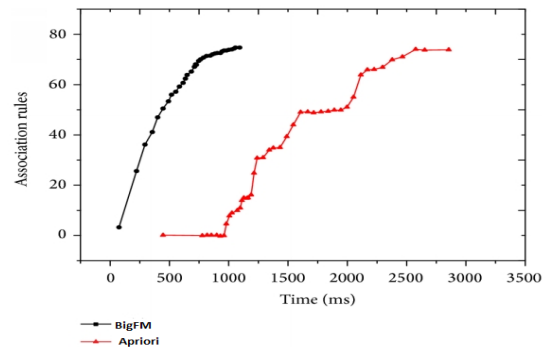


Fig 4.8 Mining efficiencies of BigFM and Apriori algorithm with different min_sup and min_conf in dataset 2

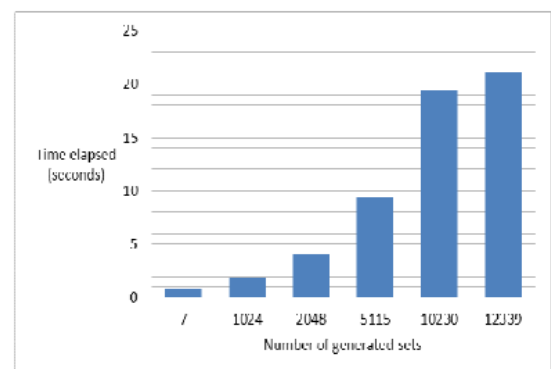


Fig 4.9 Relationship between No of Generated Item Sets and time

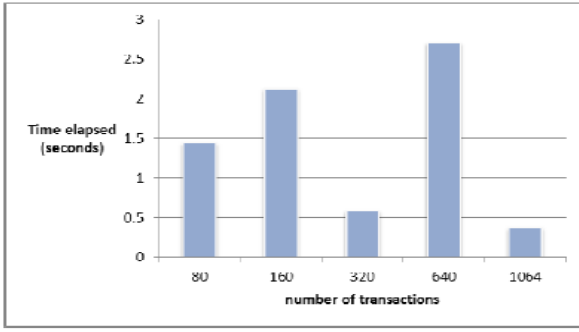


Fig 4.10 Relationship between number of transactions and time in BigFM

The following table shows results of BigFM and Apriori on Big Data sets

Table 4.1 Results of BigFM Vs Apriori on Big Data Sets

Status	#new Transactions	# Transactions	Big FM	Apriori
First run	10	10	0.51	0.17
Frequent run	10	20	0.53	0.42
Frequent run	50	70	0.91	1.88
Frequent run	200	270	1.13	3.61
Frequent run	400	670	1.30	5.00
Frequent run	600	1270	2.67	6.98
Frequent run	1200	2470	5.28	14.98
Average Frequent run times			1.97	5.482
This is Decrease of				63.95%

The following diagrams describes the efficiency in terms of No of Different Items and Total Frequent Item Sets

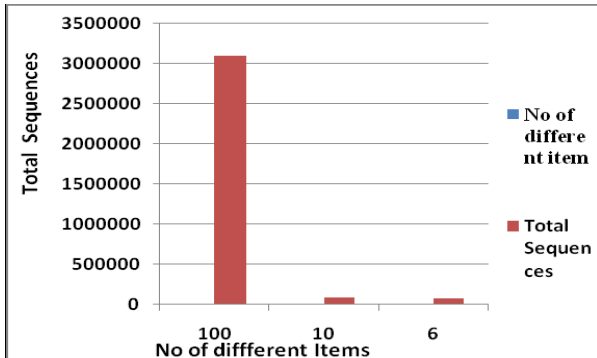


Fig 4.11 No of Different Items Vs Total Frequent Item Sets

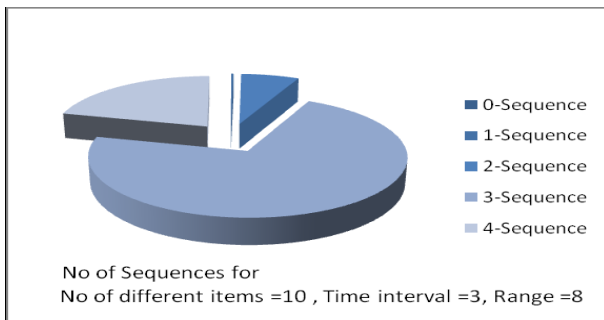


Fig 4.12 No of Different Item Sets for No of Different Items

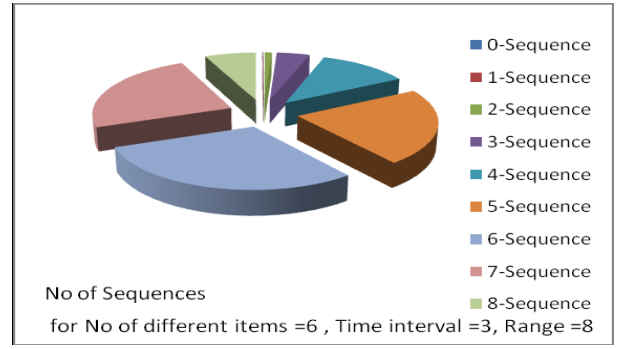


Fig 4.13 No of Different Item Sets for No of Different Items

V. CONCLUSION AND FUTURE WORK

The concept of association rule generation or mining can be done effectively in distributed systems that can use parallel executions as in Hadoop environment. This is because it can be scaled up to large volumes of data with less execution time and cost with good accuracy. The proposed algorithm in this paper also considers the type of input data and can be applied to any data formats. By dividing the input data into many splits and processing them using many nodes, the execution is made easy. The management issues such as data transfer between the nodes, storage of data, failure of any node and other issues within the cluster are all handled by Hadoop automatically. Thus the proposed system is more efficient in terms of scalability and robustness. The proposed association rule mining algorithm also has the same features and so it is efficient. Also by making use of the key-value pair approach, the processing is made much easier compared to that of the existing binomial approach. But still the proposed algorithm is not the best in performance when comes to really large datasets. So in the future the Fuzzy based association rule mining can be done in Hadoop to handle data larger than the one in this paper. Further the input data can be classified based on the calculated support and confidence values by using a suitable classification algorithm. In future this work can be extended to implement feature selection first using information gain or mutual information before implanting Association Rule Mining.

REFERENCES

1. Cheikh Tidiane Dieng, Tao-Yuan Jen, Dominique Laurent, and Nicolas Spyrtos. Mining frequent conjunctive queries using functional and inclusion dependencies. *VLDB J.*, 22(2):125-150, 2013.
2. Bart Goethals, Dominique Laurent, Wim Le Page, and Cheikh Tidiane Dieng. Mining frequent conjunctive queries in relational databases through dependency discovery. *Knowl. Inf. Syst.*, 33(3):655-684, 2012.
3. Cheikh Tidiane Dieng, Tao-Yuan Jen, and Dominique Laurent. An efficient computation of frequent queries in a star schema. In *Database and Expert Systems Applications, 21th International Conference, DEXA 2010, Bilbao, Spain, August 30 - September 3, 2010, Proceedings, Part II*, pages 225-239, 2010.
4. Kun He, Yiwei Sun, David Bindel, John E. Hopcroft, and Yixuan Li. Detecting overlapping communities from local spectral subspaces. In *2015 IEEE International Conference on Data Mining (ICDM 2015), Atlantic City, NJ, USA, pages 769-774, 2015*.
5. Ayad Ibrahim, Hai Jin, Ali A. Yassin, and Deqing Zou. Towards privacy preserving mining over distributed cloud databases. In *Proceedings of the 2nd International Conference on Cloud and Green*

- Computing (CGC 2012)*, Xiangtan, Hunan, China, pages 130{136. IEEE Computer Society, 2012.
6. Leila Ismail and Liren Zhang. Modeling and performance analysis to predict the behavior of a divisible load application in a cloud computing environment. *Algorithms*, 5(2):289{303, 2012.
7. Fan Jiang and Carson Kai-Sang Leung. Stream mining of frequent patterns from delayed batches of uncertain data. In *Proceedings of the 15th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2013)*, Prague, Czech Republic, pages 209{221. Springer-Verlag New York, Inc., 2013.
8. Alfredo Cuzzocrea, Ladjel Bellatreche, and Il-Yeol Song. Data warehousing and olap over big data: Current challenges and future research directions. In *Proceedings of the 16th International Workshop on Data Warehousing and OLAP (DOLAP 2013)*, San Francisco, California, USA, pages 67{70. ACM, 2013.
9. Malu Castellanos, Chetan Gupta, Song Wang, and Umeshwar Dayal. Leveraging web streams for contractual situational awareness in operational BI. In *Proceedings of the 2010 International Conference on Extending Database Technology/International Conference on Database Theory (EDBT/ICDT 2010) Workshops*, Lausanne, Switzerland, pages 7:18. ACM, 2010.
10. Alfredo Cuzzocrea, Carson Kai-Sang Leung, and Richard Kyle MacKinnon. Mining constrained frequent itemsets from distributed uncertain data. *Future Generation Computer Systems*, 37:117{126, 2014.
11. Alfredo Cuzzocrea, Domenico Saccia, and Jefrerey D. Ullman. Big data: A research agenda. In *Proceedings of the 17th International Database Engineering & Applications Symposium (IDEAS 2013)*, Barcelona, Spain, pages 198{203. ACM, 2013.
12. Alfredo Cuzzocrea. CAMS: OLAPing multidimensional data streams efficiently. In *Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2009)*, Linz, Austria, pages 48-62. Springer verlag, 2009.
13. Yifan Chen, Xiang Zhao, Xuemin Lin, and Yang Wang. Towards frequent subgraph mining on single large uncertain graphs. In *2015 IEEE International Conference on Data Mining (ICDM 2015)*, Atlantic City, NJ, USA, pages 41{50, 2015.
14. Jeferey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107{113, 2008.
15. [dson Dela Cruz, Carson Kai-Sang Leung, and Fan Jiang. Mining 'following' patterns from big sparse social networks. In *Proceedings of the International Symposium on Foundations and Applications of Big Data Analytics (FAB 2016)*, San Francisco, CA, USA, pages 923{930. ACM, 2016.
16. Mohammad El-Hajj and Osmar R. Zaiane. Parallel bifold: Largescale parallel pattern mining with constraints. *Distributed and Parallel Databases*, 20(3):225{243, 2006.
17. Mohammad El-Hajj and Osmar R. Parallel leap: Large-scale maximal pattern mining in a distributed environment. In *Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS 2006)*, Minneapolis, USA, pages 135{142. IEEE, 2006.
18. Fan Jiang, Carson Kai-Sang Leung, Dacheng Liu, and Aaron M. Peddle. Discovery of really popular friends from social networks. In *Proceedings of the 4th IEEE International Conference on Big Data and Cloud Computing (BDCloud 2014)*, Sydney, Australia, pages 342{349, 2014.
19. Dongme Sun, Shaohua Teng, Wei Zhang, Haibin Zhu, "An Algorithm to Improve the Effectiveness of Apriori", In Proc. Int'l Conf. on 6th IEEE Int'l Conf. on Cognitive Informatics (ICCI'07), 2007.
20. Mannila, H. and Toivonen, H., "Discovering generalized episodes using minimal occurrences", In Proc. of ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), Pages 146-151, 1996.
21. Anandhavalli M, Suraj Kumar Sudhanshu, Ayush Kumar and Ghose M.K. (2009) Optimized Association Rule Mining using Genetic Algorithm, *Advances in Information Mining*, ISSN:0975-3265, Volume 1, Issue 2, 2009, pp-01-04.
22. Markus Hegland. The Apriori Algorithm – a Tutorial, CMA, Australian National University, WSPC/Lecture Notes Series, 22-27, March 30, 2005.
23. Bart Goethals, Dominique Laurent, Wim Le Page, and Cheikh Tidiane Dieng. Mining frequent conjunctive queries in relational databases through dependency discovery. *Knowl. Inf. Syst.*, 33(3):655-684, 2012.
24. Agrawal, A.; Choudhary, A. 2011. Identifying HotSpots in Lung Cancer Data Using Association Rule Mining. 11th IEEE International Conference on Data Mining Workshops. Pp. 995 – 1002
25. 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.
26. M. Boley and H. Grosskreutz. Approximating the number of frequent sets in dense data. *Knowl. Inf. Syst.*, pages 65–89, 2009.
27. L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, and P. Luo. Distributed data mining: a survey. *Information Technology and Management*, pages 403–409, 2012.
28. Apache hadoop. <http://hadoop.apache.org/>, 2013.
29. Apache mahout. <http://mahout.apache.org/>, 2013.
30. T. De Bie. An information theoretic framework for data mining. In Proc. ACM SIGKDD, pages 564–572, 2011.
31. [10] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In Proc. OSDI. USENIX Association, 2004.
32. 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.
33. K. Finley. 5 ways to tell which programming languages are most popular. ReadWrite. <http://readwrite.com/2012/06/05/5-ways-to-tell-which-programming-languages-are-most-popular>, 2012.
34. F. Geerts, B. Goethals, and J. V. D. Bussche. Tight upper bounds on the number of candidate patterns. *ACM Trans. Database Syst.*, pages 333–363, 2005.
35. 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.
36. B. Goethals. Survey on frequent pattern mining. Univ. of Helsinki, 2003.
37. S. Hammoud. MapReduce Network Enabled Algorithms for Classification Based on Association Rules. Thesis, 2011.
38. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, pages 1–12, 2000.
39. H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang. Pfp: Parallel fp-growth for query recommendation. In Proc. RecSys, pages 107–114, 2008.
40. 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.
41. N. Li, L. Zeng, Q. He, and Z. Shi. Parallel implementation of Apriori algorithm based on MapReduce. In Proc. SNPD, pages 236–241, 2012.
42. 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.
43. 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.