INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

OPEN ACCESS
Received: 06-07-2020
Accepted: 05-09-2020
Published: 22-09-2020

Editor: Dr. Natarajan Gajendran

Citation: Senthilkumar A, Hari Prasad D (2020) An efficient FP-Growth based association rule mining algorithm using Hadoop MapReduce. Indian Journal of Science and Technology 13(34): 3561-3571. https://doi.org/10.17485/IJST/v13i34.1078

*Corresponding author.
senthilcsscholar@gmail.com

Funding: None

Competing Interests: None

ISSN
Print: 0974-6846
Electronic: 0974-5645

An efficient FP-Growth based association rule mining algorithm using Hadoop MapReduce

A Senthilkumar[1]*, **D Hari Prasad**[2]

1 Research Scholar, Sri Ramakrishna College of Arts and Science, Coimbatore, 641 006
2 Professor and Head, Department of Computer Application, Sri Ramakrishna College of Arts and Science, Coimbatore, 641 006

Abstract

Objectives: To achieve improved performance of FP-Growth based Association Rule Mining algorithm for massive data by effective utilization of storage, execution capability and improved partition technique within the Hadoop MapReduce framework. **Methodology**: The proposed methodology has four main phases: In the first phase, the item sets for finding the frequent pattern are encoded and thus minimizes the expensive operation for large data set. In the second phase, improved hash partitioning reduces the network overhead and improves the communication speed within the MapReduce phase for each item set. The effective usage of network bandwidth and storage is obtained by the impact of compression in the third phase. The use of combiner in final phase for frequent item set mining minimizes the overhead of reduce phase by finding the pattern in each partition and minimizes the overall execution time of the FP-Growth algorithm. **Findings**: FP-Growth based association rule mining algorithm is designed for parallel execution on distributed cluster of servers. Changes to the MapReduce implementation of FP-Growth with the impact of encoding. Improved hash partitioning, compression and configuration results in a significant performance gain with better improvement in execution time. **Novelty/Improvements**: According to the experimental results, the changes in storage and processing level within the MapReduce framework improves the overall performance of the parallel frequent item set mining in Hadoop cluster.

Keywords: Association rule mining; Hadoop; MapReduce; FP-Growth

1 Introduction

Association Rule Mining (ARM) describes the associated relationship among different items for finding the frequent itemsets [1,2]. FP-Growth is the widely used algorithm for identifying all frequent patterns and it is the fastest approach to mine frequent item set. It uses Divide and Conquer strategy to mine the patterns. In this technique the frequent items are transformed into a frequent pattern where the itemsets are extracted directly from the FP-Tree. The drawback of the traditional approach is when the dataset is very large, the efficiency of FP-Growth algorithm reduces, uses excessive memory and also suffers performance degradation and does not support scaling to greater extend. To overcome this problem association rule mining algorithm are designed for parallel execution on distributed network. In this research work Hadoop MapReduce is used for an efficient FP-Growth based association rule mining algorithm [3].

1.1 Big Data

Big data deals with large amount of structured, unstructured and semi structured data. It is often characterized by 3vs, the extreme volume of data, the wide variety of data types and the velocity in which the data can be processed [4–6]. Hadoop is a Big data framework used for processing large amount of data. MapReduce and Hadoop Distributed File System(HDFS) are the two main components of Hadoop. In MapReduce, large amount of data usually in terabytes is processed in parallel on large clusters with high reliability and fault tolerance using low cost commodity hardware. HDFS is a reliable storage system used for storing very large files running on a cluster of low commodity hardware.

MapReduce is the heart of the hadoop and it has two main phases Map phase and the Reduce Phase [7]. In Map Phase block of data is read and processed to produce key value pairs as an intermediate output. These intermediate results which are key value pairs are taken as input to reduce phase where it aggregates all intermediate results and final output is produced. In between the Map phase and the Reduce phase there are other intermediate phases such as combiner, partitioner, shuffle and sort. Combiner is a mini reducer which performs a local reduce on the mapper results before they are distributed to the reducer. Once the combiner functionality is executed, it is then passed on to the reducer for further work. Partitioner is used when there are more than one reducer and decides which reducer is responsible for the particular key. Combiner takes the output of Mapper or combiner results and sends it to the responsible reducer based on the key. The default partitioning function is the hash partitioning function where hashing is done on the key. The process of transferring data from the mappers to reducers is shuffling and within this process sorting is also performed. The overall MapReduce job flow execution is shown in Figure 1 [8].

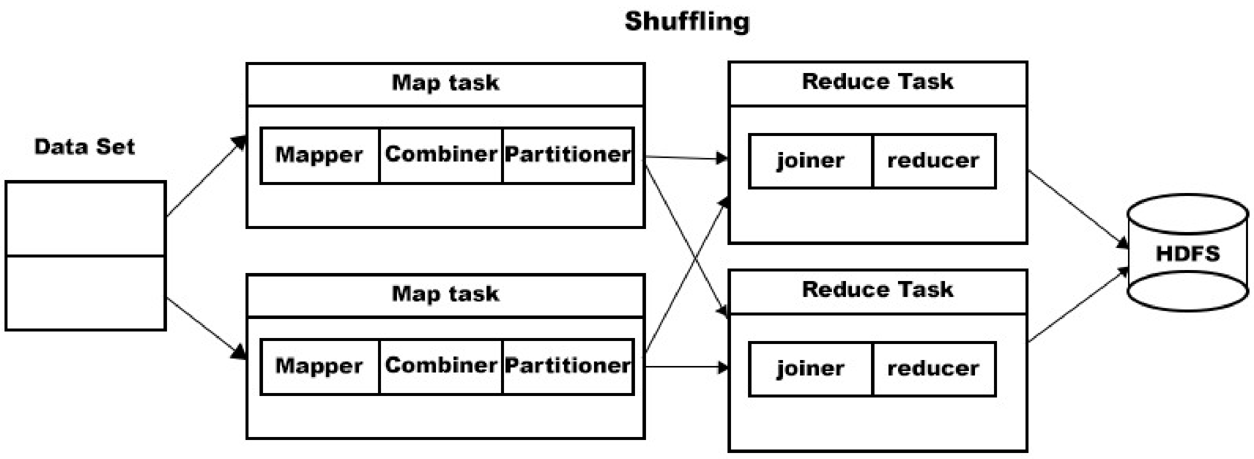

Fig 1. MapReduce Job execution flow

1.2 Motivation

The main motivation behind the research is to improve the efficiency of Association Rule Mining algorithm by generating rule for a massively large dataset in Hadoop Environment. This research work tries to improve the efficiency of Association Rule Mining in MapReduce framework with the following features.

- Dictionary Encoding - Strings are encoded as Integer. String comparison is much more expensive operation when compared to integer comparison which massively increase the execution performance.
- Hashing - When compared to String Hashing Integer Hashing will be faster and improves the computational speed.
- Compression - The effective usage of storage and network bandwidth usage is obtained by the impact of compression, which also makes the copy phase more efficient.

- Combiner - The larger the size of the input data, combiner has highly significant impact on the execution time. It minimizes the execution time of the reducer.

1.3 Background study

PrePost algorithm [9] based on the concept of N-lists is used for association rule mining which presents a data structure named N-list, for storing the information related to Association rule mining. PrePost scans the database twice to construct a tree which generates the N-list of frequent 1-itemsets. In the mining process, the database does not require rescanning, only need to intersect the merger N-list. It is a scalable Hadoop based method for frequent itemset mining that has no intermediate data, and small network communication

To boost the performance of parallel Frequent Itemset Mining(FIM) on Hadoop clusters FiDoop is used in [10] where all relevant transaction are moved into a data partition and thus the number of redundant transactions are reduced. FiDoop uses Voronoi diagram-based data partitioning technique which incorporate Locality Sensitive Hashing technique. This algorithm places highly similar transactions into a data partition to improve locality without creating an excessive number of redundant transactions. The drawback of FiDoop is when data-placement mechanism in HDFS on heterogeneous clusters is used which degrades the performance.

In [11], authors proposed Kavosh: An effective Map-Reduce based association rule mining method. When dealing with large volume of data in a map reduce environment the issues faced are data locality and skewness. Kavosh proposed a Kavosh format where the input data is converted in a custom format. Data are distributed over nodes and does not require inputs from other nodes. Each node performs the task independently as it follows a unified format and hence exchange of data between the nodes is reduced which compresses input data to facilitate data management. Another advantage is the lack of process skewness because it is possible to allocate a predefined amount of data to each node. This is not suitable for high-dimensional data since it requires many nodes.

In [12] the authors proposed a big data analytics framework that uses Hadoop-based parallel computing to achieve high-efficiency mining of itemsets with multiple item supports. Hadoop MapReduce architecture is employed to determine the support for each item. Next, in the analytics phase, sub-transaction blocks are generated. To facilitate decision makers, the concept of classification of item, which classifies items of higher homogeneity into the same class, by which the items inherit class support as their item support is used. The algorithm is implemented on the distributed computing framework which enables individual analysis of the sub-transaction DBs. The analysis results from each reducer were further aggregated to generate association rules for the entire dataset. In [13] the authors used Hybrid Frequent Itemset Mining (HFIM) technique in Spark to optimize the execution time. HFIM uses vertical and horizontal layout of the dataset to find the association. It overcomes the challenges that are there in Apriori Algorithm i.e. it needs huge memory and resource for computing.

Using Hadoop's distributed and parallel MapReduce environment, an architecture is presented in [14] to mine positive as well as negative association rules in big data using frequent itemset mining and the Apriori algorithm. The results of a few optimization parameters which are based on the number of rules generated as well as the run-time efficiency in Hadoop's MapReduce environment is analyzed and presented. According to their findings, higher amount of parallelization, which means larger block sizes, will increase the run-time efficiency of the Hadoop implementation of the Apriori algorithm.

2 Problem Identification

A number of research efforts have explored to address the problem of frequent itemset mining in parallel and distributed environments. The existing problems of frequent itemset mining in large-scale datasets for generating association rules have large research gaps. Traditional Association mining algorithm when dealing with large dataset uses excessive memory and also suffers performance degradation due to scanning the database for multiple time and it is difficult to scale. There were also many research efforts for efficient execution of Association Rule Mining algorithm in Big data. The existing methods suffer from heterogeneous cluster [10] and are not suitable for high dimensional dataset. To improve the performance of frequent patter mining, this study proposes an effective algorithm which mines association rules from frequent pattern by using MapReduce concept with multiple item support.

3 Proposed Methodology

Traditional Association rule mining algorithm lacks performance and support, when it comes to deal with large volume of data and also it is impossible to generate rules at a faster pace. These algorithms do not have mechanisms like Load balancing, data distribution etc. Big data is used to analyze the huge volume of data and get the association between the items and also checks if there are any trend in the data [15,16]

In the proposed method, FP-Growth algorithm is used in Hadoop MapReduce environment for the better efficiency of association rule mining. FP-Growth is the widely used algorithm for identifying all frequent patterns and it is the fastest approach to mine frequent itemset. The Proposed method extracts market basket analysis rules using FP-Growth for frequent pattern mining. To improve the performance of the parallel frequent itemset mining in Hadoop cluster, dictionary encoding, compression, combiner and partitioning is used and it is explained below.

Dictionary encoding also known as substitution coder can encode String, char and varchar columns. It operates by searching for matches between the text to be compressed and a set of strings contained in a data structure maintained by the encoder. For dictionary encoding the dictionary is sorted and 8 bytes of each unique values are placed into dictionary data. The length of each item in the dictionary is put into the length stream. The Data stream consists of the sequence of references to the dictionary elements. When the encoder finds a match, it substitutes a reference to the string's position in the data structure.

File compression brings two major benefits: it reduces the space needed to store files, and it speeds up data transfer across the network or to or from disk. Mining frequent itemsets over big transactional data using Apriori on Hadoop framework can be achieved by utilizing compressed bitmaps of hadoop framework. In Hadoop there are different compression formats such as Bzip2.bz2 – Snappy.snappy – Gzip .gz – LZO .lzo and so on. More Efficient fast compression algorithms offer low compression ratio. Higher compression algorithms are usually slower and higher in CPU overheads. GZIP format balances between Compression & Performance and it is used in the proposed work. In the proposed method, BZip2 compression is enabled for all MapReduce jobs for better execution time.

Combiner function is applied to reduce communication cost of transferring intermediate outputs of mappers to reducers. Output pairs of mapper are locally sorted and grouped on same key and feed to the combiner to make local sum. The intermediate output pairs of combiners are shuffled and exchanged between machines to group all the pairs with the same key to a single reducer.

The role of partitioning in MapReduce is to control the distribution of map output keys and the number of partitions is equal to the number of reducers. Partitioner assigns and controls which Key will be processed by which Reducer and data mapped to a single partition is processed by a single reducer. The default partitioning method uses hash function. With the existing scenario when the frequent itemset mining is executed in hadoop environment, the main focus will be on balancing the data such that it can be equally partitioned and distributed among cluster nodes. To better improve the performance of the rule mining, the data can be balanced in a cluster such that the data which have strong correlation can be partitioned to the same partition. This can be performed by using LSH(Locality Sensitive Hashing) partitioning. In LSH the data items which have high probability are partitioned into the same buckets. The purpose of using LSH in MapReduce is to maximize the items with high correlation are mapped to the same partition in reduced phase and thus improves the overall performance of the MapReduce jobs while also reducing the memory foot print as it eliminates the redundancy of same items copied to multiple reducers. The overall flow diagram is shown in Figure 2 [10].

For the given database the steps for implementing the proposed algorithm,"An Efficient FP-Growth based Association rule mining algorithm using Hadoop MapReduce" is shown below:

Step 1: Read Dataset

(i) The given database is encoded using dictionary encoding. The String is encoded and converted into a integer which results in reduction of file size

(ii) The encoded data is sent as input for MapReduce Job.

Step 2: Parallel Counting and Sorting of items

(iii) The support values of all the items are calculated to find the popularity of all items and the support values are sorted.

(iv)Combiners are used to achieve maximum time, memory usage and computation cost savings.

(v) During the Map Reduce job, the map output which is copied over the network is compressed using Bzip2 Codec which decreases network IO.

Step 3 : FP-Growth Processing

(vi) In this step the actual FP-Growth implementation is done and calculates the FP-Tree using the MapReduce framework.

(vii) The partitioning is done with Modified version of LSH to handle Integer data and LSH ensures reduced network IO and memory usage as it maximises similar items are processed within the same partition. And here also the map output is compressed using Bzip2 Codec before sent over the network to the Reducer machine

Step 4: Aggregation

The last Phase is the Aggregation phase which aggregates the output of previous FP-Growth algorithm's output and produces the final result of Association Rules

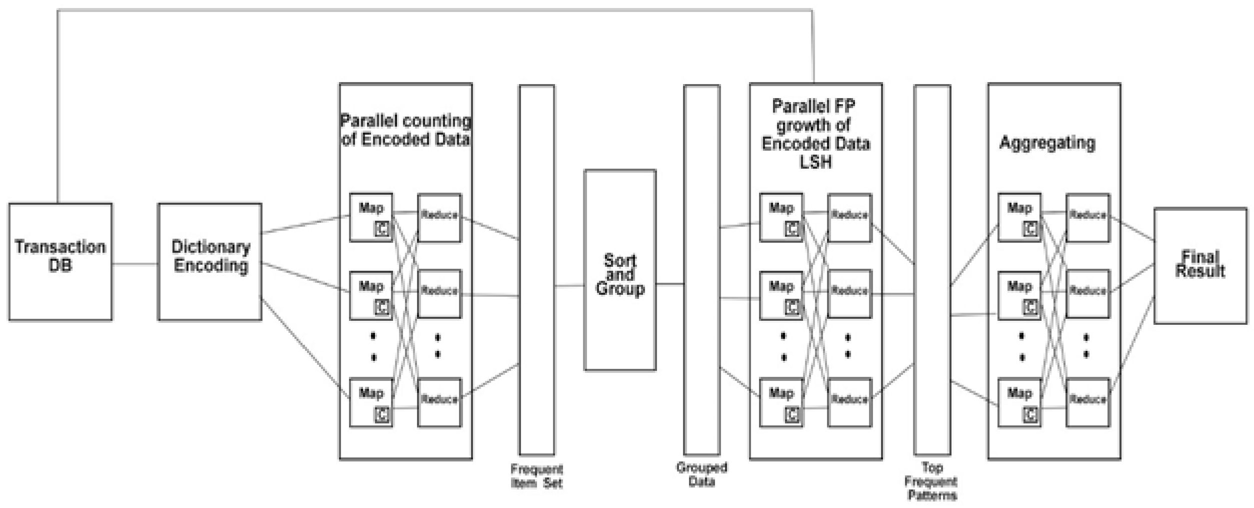

Fig 2. Flowchart of the proposed method

4 Discussion

The proposed method is implemented using Ubuntu 16.04 with Hadoop installed in each node. The dataset used is the IBM Quest Market-Basket Synthetic generator which is used to evaluate the performance of the model. The data in the dataset includes the parameter number of transactions, average number of items per transaction, total number of items and average transaction length. All these parameters can be controlled as command line arguments for the IBM Quest data generator. The configuration environment is the Intel Xeon 5670 6 core dual processors with 12 CPU cores with 32 GB Ram, Cisco catalyst 2948G -GE-TX and GBPS network switch. The model is executed in a three node cluster.

The algorithm behavior is evaluated in terms of input data properties, execution metrics and the cluster size. The various input data properties taken for evaluation are the file size, average items per transaction, correlation and support of each item. The execution metrics are evaluated in terms of execution time and shuffle byte size. For evaluating the performance in the cluster environment the number of node taken is three. The dataset used for evaluation is shown in Table 1. In this paper the performance metrics used is execution time and it is evaluated based on support, dimensions and number of nodes. The proposed method is compared with the existing parallel FP-Growth algorithm Kavosh [11].

Table 1. Dataset description

Parameters	Average Length	No. of Items	Avg.Size/Transaction
T10I4D	10	4000	17.5B
T40I10D	40	10000	31.5B
T60I10D	60	10000	43.6B
T85I10D	85	10000	63.7B

In the proposed work to perform Association rule mining using FP-Growth algorithm, three different tasks are submitted as three different jobs to Hadoop MapReduce. Job 1 performs parallel counting, Job 2 performs Parallel FP-Growth and job 3 performs aggregation.

The execution time for job 1 and job 2 for the existing and proposed method for different support threshold values are shown in Tables 2 and 3 respectively and the corresponding graph is depicted in Figures 3 and 4. As the support value increases the execution time is reduced. When compared to the existing method, the execution time is minimized in the proposed method.

Table 2. Execution Time of Job1 for different support threshold values

Support threshold values	Execution Time(Sec) for Proposed Method	Execution Time(Sec) for Existing Method
0.0005	305	528
0.001	311	491
0.0015	297	470
0.002	285	469
0.0025	280	470

Table 3. Execution time of Job2 for different support threshold values

Support threshold values	Execution Time(Sec) for Proposed Method	Execution Time(Sec) for Existing Method
0.0005	131	203
0.001	121	197
0.0015	121	194
0.002	120	199
0.0025	120	196

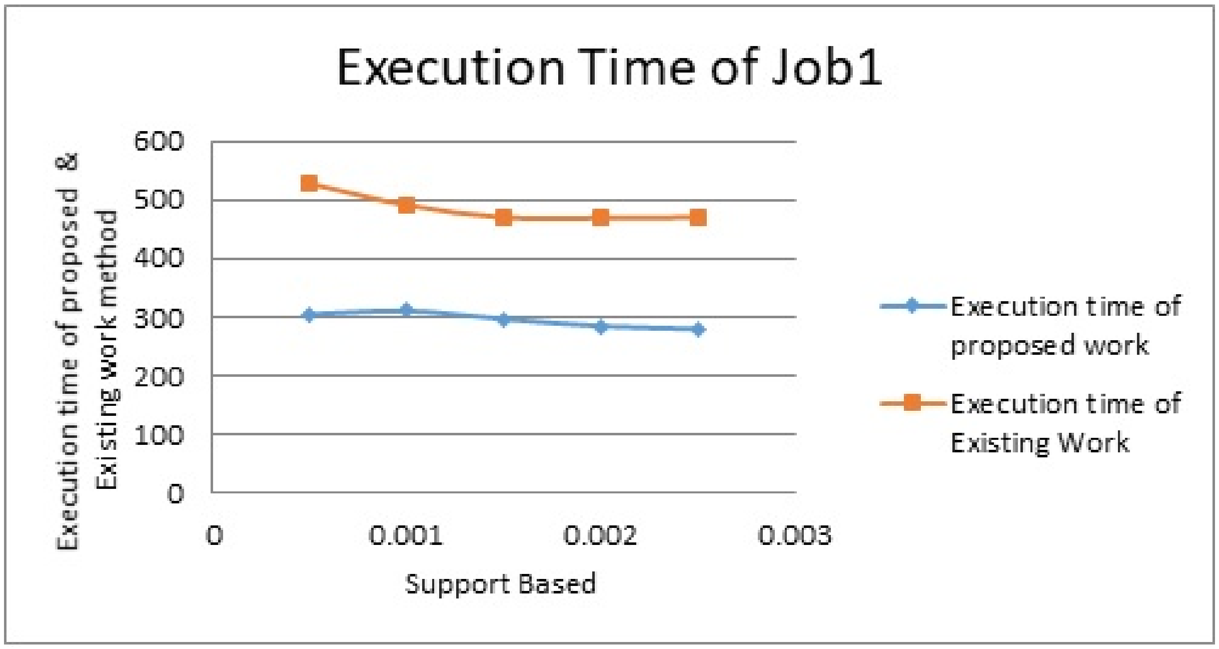

Fig 3. Execution time of Job 1 for different support values

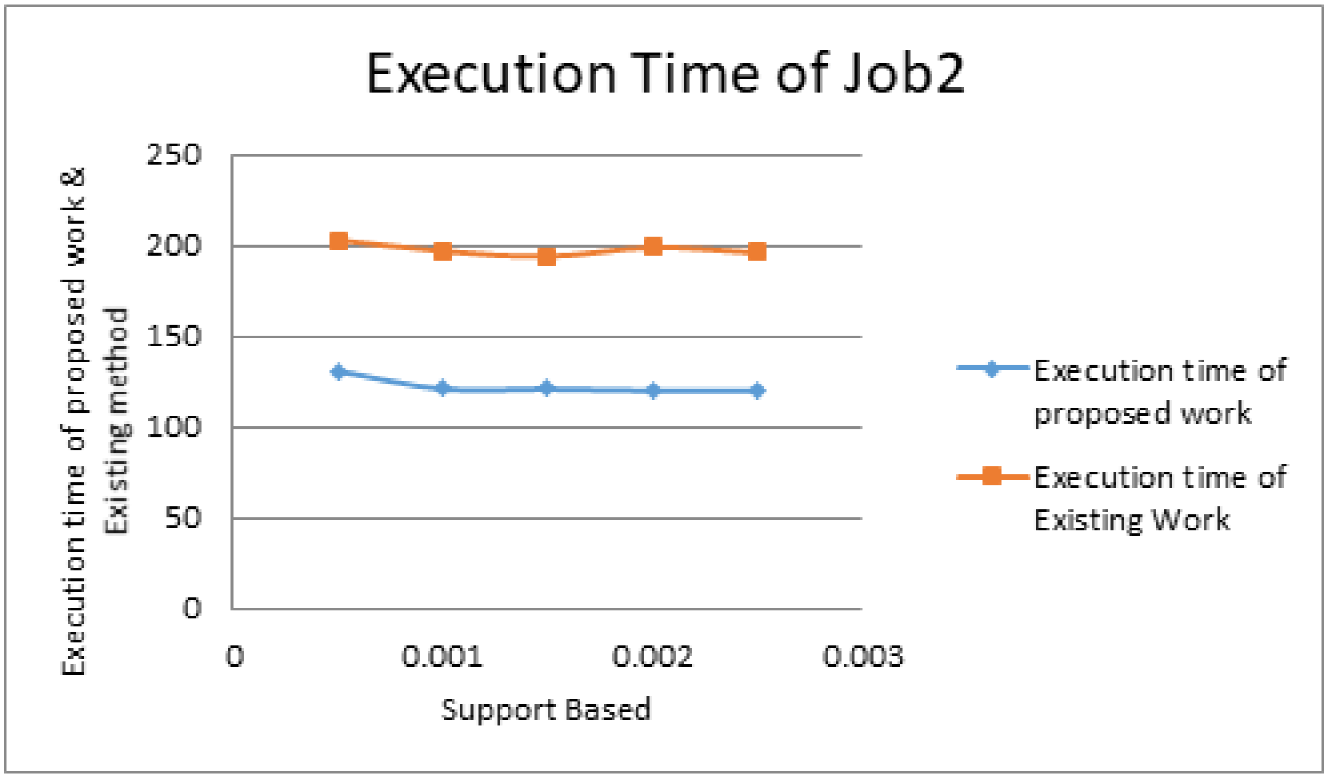

Fig 4. Execution time of Job 2 for different support values

The execution time for job 1 and job 2 for the existing and proposed method for different dimensions where dimension represents average length per transaction and is shown in Tables 4 and 5 respectively and the corresponding graph is illustrated in Figures 5 and 6. As the dimension increases there is an increase in the execution time. When compared to the existing method, the execution time is minimized in the proposed method.

Table 4. Execution Time of Job1 for different dimensions

Dimension Based	Execution time of proposed work	Execution time of Existing Work
10	19	43
40	107	169
60	148	243
85	584	934

Table 5. Execution time of Job 2 for different dimensions

Dimension Based	Execution time of proposed work	Execution time of Existing Work
10	13	26
40	57	67
60	73	107
85	195	382

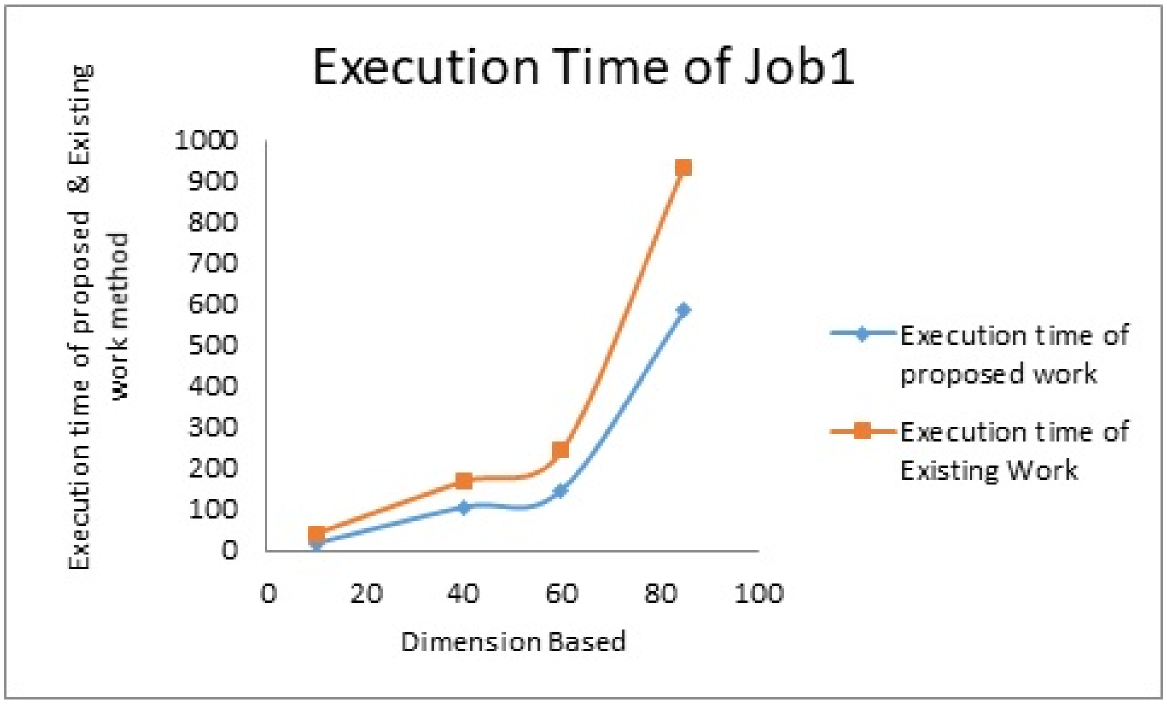

Fig 5. Execution time of Job 1 for different dimensions

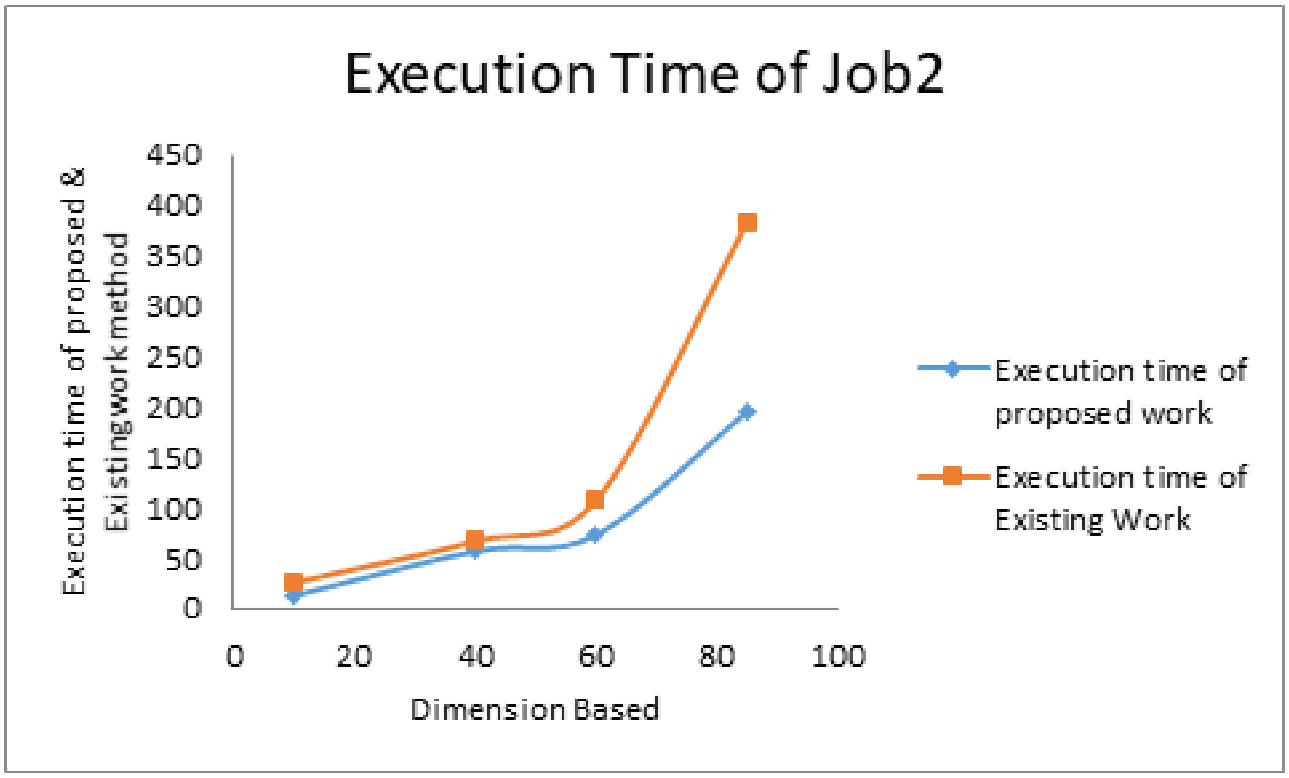

Fig 6. Execution time of Job 2 for different dimensions

The execution time for job 1 and job 2 for the existing and proposed method for different node size is shown in Tables 6 and 7 respectively and the corresponding graph is illustrated in Figures 7 and 8. As the node size increases there is a decrease in the execution time. When compared to the existing method, the execution time is minimized in the proposed method.

Table 6. Execution time of Job1 for different node size

Num of Nodes	Execution time of proposed work	Execution time of Existing Work
1	179	162
2	74	121
3	70	108

Table 7. Execution time of Job2 for different node size

Num of Nodes	Execution time of proposed work	Execution time of Existing Work
1	179	162
2	74	121
3	70	108

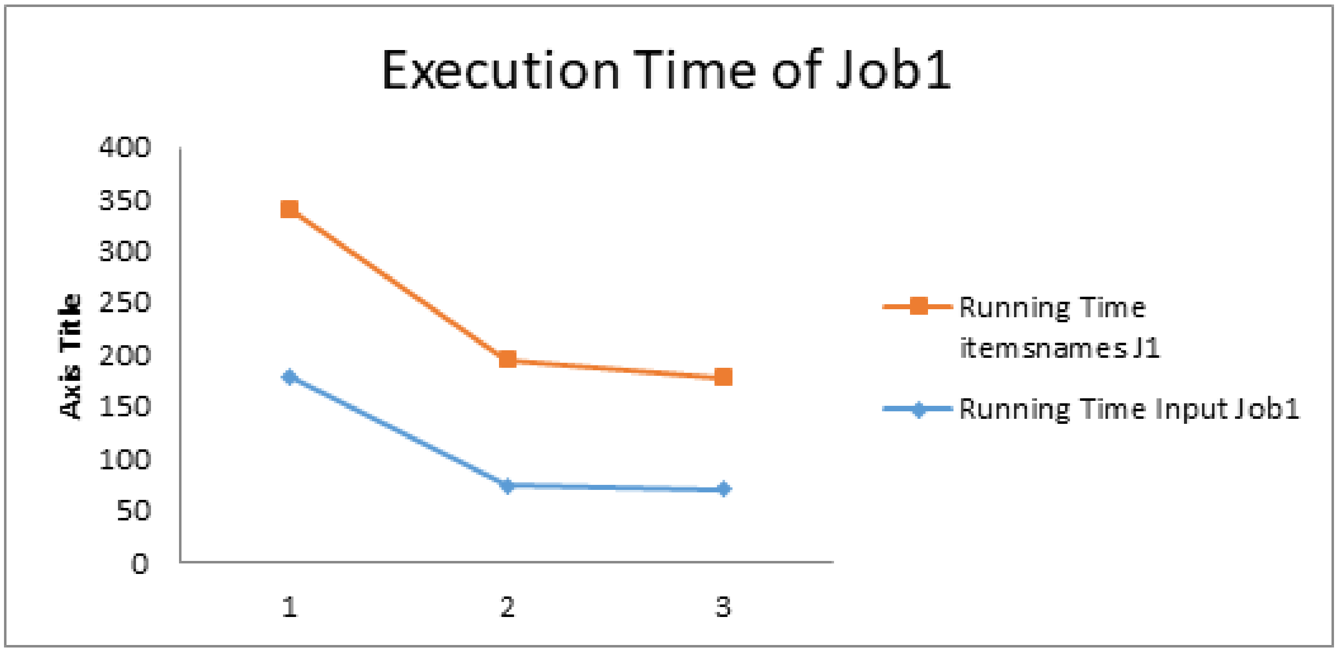

Fig 7. Execution time of Job 2 for different node size

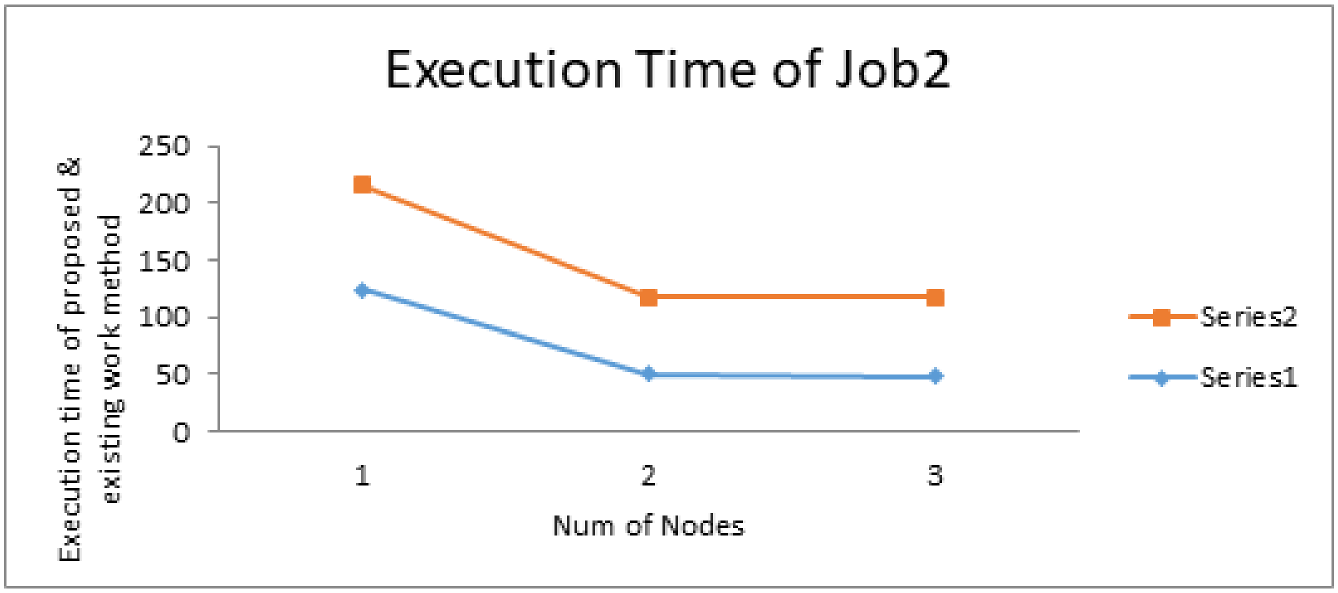

Fig 8. Execution time of Job 2 for different node size

5 Summary and Conclusion

Association Rule Mining is used to find frequent patterns but for a large dataset, traditional association rule mining algorithm is not suitable as it requires more scanning of datasets to find the frequent item set. To overcome this problem association rule mining algorithms are designed for parallel execution on distributed network. When used in big data perspective, its performance can be improved. In this research work Hadoop Mapreduce is used for an efficient FP-Growth based association rule mining algorithm. To improve the performance of the parallel frequent itemset mining in Hadoop cluster dictionary encoding with compression, combiner and partitioning is used. The Association Rule Mining Algorithm is evaluated in terms of execution time in different dimensions and number nodes in cluster. The proposed algorithm shows better performance when compared to the existing methods. This work can be extended, where HDFS remains the storage and the execution framework will be Spark.

References

1) Lin JCW, Gan W, Fournier-Viger P, Yang L, Liu Q, Frnda J, et al. High utility-itemset mining and privacy-preserving utility mining. *Perspectives in Science*. 2016;7:74–80. Available from: https://dx.doi.org/10.1016/j.pisc.2015.11.013.
2) Kumbhare AT, and VSC. An Overview of Association Rule Mining Algorithms. *International Journal of Computer Science and Information Technologies*. 2014;5(1):927–930.
3) Shirke D, Varshney D. Parallel Mining of Frequent Itemsets in Hadoop Cluster Having Heterogeneous Nodes. *International Journal of Advance Research in Computer Science and Management Studies*. 2017;5(7):129–136. Available from: https://doi.org/10.1109/CBD.2013.22.
4) Siddiqa A. A survey of big data management: Taxonomy and state-of-the-art. *J Netw Comput Appl*. 2016;71:151–166. Available from: https://doi.org/10.1016/j.jnca.2016.04.008.
5) Fournier-Viger P, Lin JCW, Kiran RU, Koh YS, Thomas R. A survey of sequential pattern mining. *Data Science and Pattern Recognition*. 2017;1:54–77. Available from: https://dl.acm.org/doi/10.1145/3314107.
6) Choi TM, Chan HK, Yue X. Recent Development in Big Data Analytics for Business Operations and Risk Management. *IEEE Transactions on Cybernetics*. 2017;47(1):81–92. Available from: https://dx.doi.org/10.1109/tcyb.2015.2507599.
7) Khezr SN, Navimipour NJ. MapReduce and Its Applications, Challenges, and Architecture: a Comprehensive Review and Directions for Future Research. *Journal of Grid Computing*. 2017;15(3):295–321. Available from: https://dx.doi.org/10.1007/s10723-017-9408-0.
8) Gu R, Yang X, Yan J, Sun Y, Wang B, Yuan C, et al. Shadoop Improving MapReduce Performance by Optimizing Job Execution Mechanism in Hadoop Clusters. *Journal of Parallel and Distributed Computing*. 2014;74(3):2166–2179. Available from: https://doi.org/10.1016/j.jpdc.2013.10.003.
9) Rochd Y, and IH. Performance Improvement of PrePost Algorithm Based on Hadoop for Big Data. *International Journal of Intelligent Engineering and Systems*. 2018;11(5):226–235. Available from: https://dx.doi.org/10.22266/ijies2018.1031.21.
10) Xun Y, Zhang J, Qin X, Zhao X. FiDoop-DP: Data Partitioning in Frequent Itemset Mining on Hadoop Clusters. *IEEE Transactions on Parallel and Distributed Systems*. 2017;28:101–114. Available from: https://dx.doi.org/10.1109/tpds.2016.2560176.

11) Barkhordari M, Niamanesh M. Kavosh: an effective Map-Reduce-based association rule mining method. *Journal of Big Data*. 2018;5(1). Available from: https://dx.doi.org/10.1186/s40537-018-0129-4.
12) Wang CS, Chang JY. MISFP-Growth: Hadoop-Based Frequent Pattern Mining with Multiple Item Support. *Applied Sciences*. 2019;9(10). Available from: https://dx.doi.org/10.3390/app9102075.
13) Sethi KK, Ramesh D. HFIM: a Spark-based hybrid frequent itemset mining algorithm for big data processing. *The Journal of Supercomputing*. 2017;73(8):3652–3668. Available from: https://dx.doi.org/10.1007/s11227-017-1963-4.
14) Bagui S, Dhar PC. Positive and negative association rule mining in Hadoop's MapReduce environment. *Journal of Big Data*. 2019;6(1). Available from: https://dx.doi.org/10.1186/s40537-019-0238-8.
15) Xia D, Lu X, Li H, Wang W, Li Y, Zhang Z. A MapReduce-Based Parallel Frequent Pattern Growth Algorithm for Spatiotemporal Association Analysis of Mobile Trajectory Big Data. *Complexity*. 2018;2018:1–16. Available from: https://dx.doi.org/10.1155/2018/2818251.
16) Al-Hamodi AGA, Lu S, Al-Salhi EAY. An Enhanced Frequent Pattern Growth Based on MapReduce for Mining Association Rules. *International Journal of Data Mining & Knowledge Management Process*. 2016;6(2):19–28. Available from: https://dx.doi.org/10.5121/ijdkp.2016.6202.