A Parallel Apriori Algorithm and FP- Growth Based on SPARK

Priyanka Gupta[1], *Dr.Vinaya Sawant*[2]

[1]Dwarkadas J Sanghvi College of Engineering, Mumbai
[2]Dwarkadas J Sanghvi College of Engineering, Mumbai

Abstract. Frequent Itemset Mining is an important data mining task in real-world applications. Distributed parallel Apriori and FP-Growth algorithm is the most important algorithm that works on data mining for finding the frequent itemsets. Originally, Map-Reduce mining algorithm-based frequent itemsets on Hadoop were resolved. For handling the big data, Hadoop comes into the picture but the implementation of Hadoop does not reach the expectations for the parallel algorithm of distributed data mining because of its high I/O results in the transactional disk. According to research, Spark has an in-memory computation technique that gives faster results than Hadoop. It was mainly acceptable for parallel algorithms for handling the data. The algorithm working on multiple datasets for finding the frequent itemset to get accurate results for computation time. In this paper, we propose on parallel apriori and FP-growth algorithm to finding the frequent itemset on multiple datasets to get the mining itemsets using the Apache SPARK framework. Our experiment results depend on the support value to get accurate results.

Keywords:-Data Mining, Association Rule Mining, Apriori Algorithm, FP- Growth Algorithm, RDD, Apache Spark.

1 INTRODUCTION

The Association Rule Mining (ARM) methodology is used to extract relevant associations from a transactional dataset. Even though association rules are produced from them, the development of frequent itemsets is the most computationally intensive element of the association rule mining process. For the regularity of the used itemsets, various adaptions have been evolved by various experts for the procedures to speed the range of the items. These algorithms were unable to process large data sets due to their sequential computing structure. For larger datasets, parallel and distributed types of frequent itemsets mining algorithms have been evolved to build the frequent itemsets. To handle the massive amount of data, the Hadoop platform is designed. During execution, it moves data processing to the nodes rather than sending data to the nodes. HDFS (Hadoop Distributed File System) and MapReduce, Hadoop's two primary components, manage data storage and computing, respectively. MapReduce is a scalable and fault-tolerant parallel programming technology. HDFS is a file system that separates enormous amounts of data into blocks and replicates them across all nodes in a cluster. Throughout the cluster, an application is stated as a MapReduce job, that runs on separates tasks which are divided into various tasks and run on different blocks of data simultaneously.

1.1 Apache Spark Framework

Apache SPARK is a cluster computing framework that is free and open source. It was first developed in 2009 at UC Berkeley's AMPLab. In Java, it provides a high level of APIs. Due to the in-memory computing technique, Spark provides the implementation 10 times faster than the Hadoop framework. It provides batch and real-time iterative processing of data. Spark provides a programming interface for "Resilient Distributed Dataset", a type of data structure. In Apache Spark, RDD is an abstraction. Spark is platform-independent it can run on any platform.

1.2 Resilient Distributed Dataset (RDD)

In Apache Spark, RDD is the basic abstraction of data structure. The main feature of RDD is that it maintains a partitioned, immutable collection of elements that can be managed simultaneously. RDD can improve the execution and makes Spark acceptable for interacting with datasets. Dataset will divide into logical patterns which can be measured on the individual node. There are two types of RDD operations: Transformation and Action. Transformation is used to create a new RDD from the existing whereas action is used to perform the computation and return the result to the main program.

2 LITERATURE REVIEW

Sanjay Rathee et.al [1] have proposed an adaptive-miner algorithm for efficient use of distributed association rule mining algorithm using spark. They make use of a reduced approach when several frequent itemsets are large else basic Apriori is used. For each iteration the execution plan with the minimum cost. The approach they have used for every iteration will give fast and efficient results.

Ravi Ranjan et.al [2] they perform the Apriori and FP-growth algorithm on Hadoop-MapReduce on the various dataset. They stated that big data can be used to make accurate predictions and enhance the association rule mining algorithm. Also, they raised some research questions based on which technique is used and how they will execute on various datasets and more questions are there.

Samba N et.al [3] represented a parallel version of FP-Growth algorithm to work as efficiently for processing large datasets. They demonstrate the proposed algorithm can work efficiently to process large datasets.

Shaozang C et.al [4] used distributed FP-Growth algorithm to mining the frequent itemset in three steps: Step 1 is to calculate 1-itemsets, step 2 i.e for conditional pattern base, step 3 is for mining frequent itemset. Their implementation results state that distributed FP- growth algorithm works better than Parallel FP- Growth algorithm.

Zhaowei Q, et.al [5] present an approach for imputation algorithm based on Apriori they aimed at the issue of incomplete data. They used the distributed approach to decrease the execution time of datasets. They also stated that they need more approaches to find the solution of how the problems will be resolved.

Shaosong Y, et.al [6] have used the improved kind of parallel Apriori algorithm based on Spark to mine the frequent itemsets. They stated that using the IAP algorithm, they worked on the parallelization and verified by the various comparative datasets.

Ravi R, et.al [7] have used the Apriori algorithm on Hadoop MapReduce to get accurate results using various datasets to mine the frequent itemsets. They have worked on a single node to get the results. They also stated that they were used to imitate the performance of the distributed environment.

Feng G, et.al [8] have used the matrix-based pruning method to mine the frequent itemsets under the MapReduce framework. They have used the novel pruning technique to get accurate results. Also, implemented the algorithm under the Spark framework to stimulate the efficiency of the results.

Sanjay R, et.al [9] presented a new algorithm based on Apriori i.e R-apriori used to mine the frequent pattern from large datasets having various attributes. They have used the reduced approach to mine the frequent itemsets. They stated that the conventional Apriori algorithm takes too much time than the parallel apriori algorithm.

Hongjian Q, et.al [10] frequent itemset mining algorithm i.e Apriori algorithm used to determine the fast and efficient algorithm that can handle large datasets using spark. They proposed the YAFIM algorithm for the use of the fast and efficient result of frequent itemset mining. They used the concept of RDD of the algorithm. They also mentioned the application which can be used for the association rule mining algorithm.

3 PROPOSED SYSTEM

Basic Association Rule Mining algorithms are used the serial approach but they were not beneficial for the larger data size. As the size of datasets become larger, their accuracy began to degrade. As a result, to manage the larger datasets parallel algorithm comes into the picture. There are many cluster-based algorithms that are there to handle the larger datasets but they produced several problems such as data replication, synchronization, etc. Therefore, the parallel approach is replaced by the MapReduce approach. An algorithm like Apriori and FP-Growth is the best for finding the frequent itemsets. In Apriori, key-value pairs are produced easily. Using Map-Reduce approach, all of which display a significant performance enhancement over the traditional algorithm. For implementing the Apriori algorithm, MapReduce Approach is the best platform. But using Hadoop also has several limitations. For each iteration, results are saved to HDFS on the Hadoop platform and for the next iteration, input is taken from HDFS which shows the reduction in productivity because of input-output time. So, RDD (Resilient Distributed Dataset) in Spark helps to resolve these problems. The main point of using these features is that it stores the result at the end of the iteration in a local cache and it's available for subsequent iterations. Both Apriori and FP-Growth algorithms are implemented on the Spark platform with multiple datasets to mine the frequent itemsets and generates the association rules.

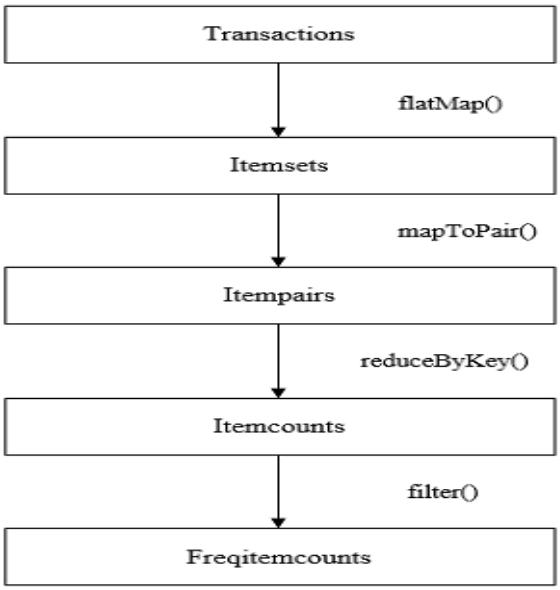

Fig 1:- RDD of Apriori Phase 1

Phase 1 generates the frequent items only.
It returns an RDD of objects that separates the transaction with the flatMap() transformation.
mapToPair() transformation generates the (item, 1) pairs for each item.
It adds all the values of the similar key together by the reduceByKey() function.
Lastly, the items are filtered out the pairs whose value is less than the minimal support with the filter() transformation and leaves those pairs which are frequent items then it took to evaluate them.
The action operation collect() returns the driver's frequently used objects for usage in the next phase.

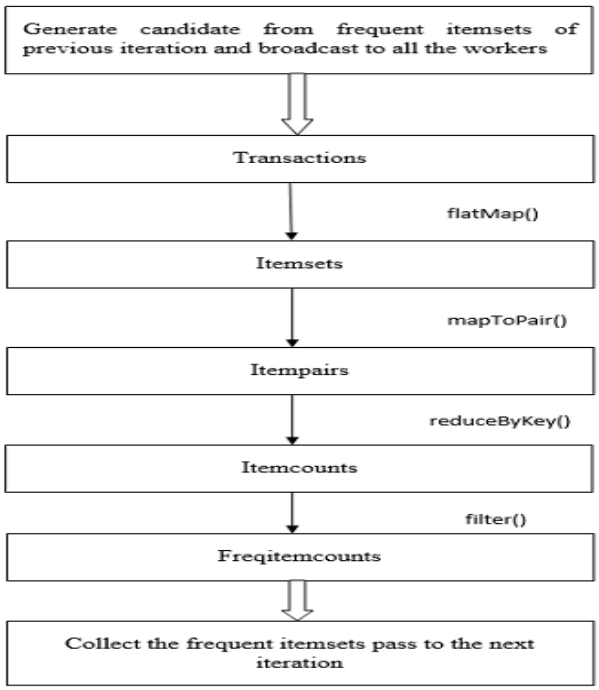

Fig 2:- RDD of Apriori Phase 2

The output of the previous iteration is the input of phase 2.
In previous iterations, (k-1) frequent are created i.e used for k^{th} iteration.
It returns a list of all candidate k-itemsets with the flatMap() transformation that the subset() operation has generated. Each itemset is converted into (itemset, 1) pairs, or key-value pairs, by the transformation mapToPair(). By summing all values with the same key, reduceByKey() reduces these pairs. Finally, the filter() method filters out infrequent itemsets, returning just frequent k- itemsets

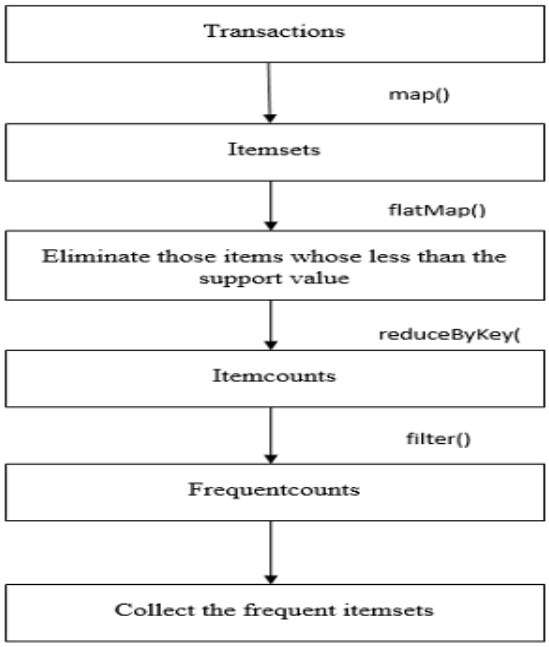

Fig 3:- RDD of FP-Growth

It involves the execution of two scans on the transaction database.
The first scan is to build up the support of every item and then select those items with the minimum support. It produces the frequent itemsets which store in frequency in descending order. The second scan builds the FP-Tree.
From the diagram, first, a transaction t is a scan from the database. It checks the prefix of t maps to the path in the FP- tree. The support count of the corresponding node is incremented in a tree.
New nodes are created if there is no overlapped path, and then it will eliminate whose items which is less than the support value by using reduceByKey() transformation.
Lastly, the items are filtered out the pairs which are frequent items using filter() transformation.
Then, action operation collect() returns the frequent items pairs.

4 EXPERIMENT RESULT

The experiment analysis was based on the support value provided by the user at the time of execution. It was performed on a single workstation. We have used four

datasets with the sample dataset to examine the results. We present the performance evaluation of both algorithms based on execution time.

4.1 Experimental Setup

The setup of implementation was built on a workstation with an Intel i7 processor, 16 GB RAM, 2 TB of the hard disk. Hadoop – 2.2.0 and python language is used for coding which is running on windows 10.

All source codes are written in python. On a single workstation, algorithms are tested.

4.2 Evaluation Measures

In this paper, different evaluation measure has been used to evaluate the quality of proposed methodology. The figure shows the implementation results of each dataset.

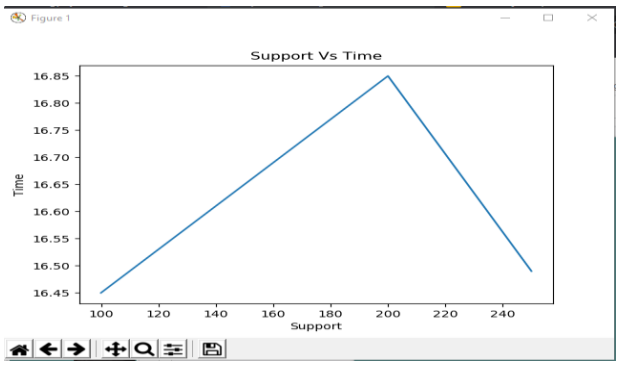

Fig 4:- Apriori Of Instacart Dataset

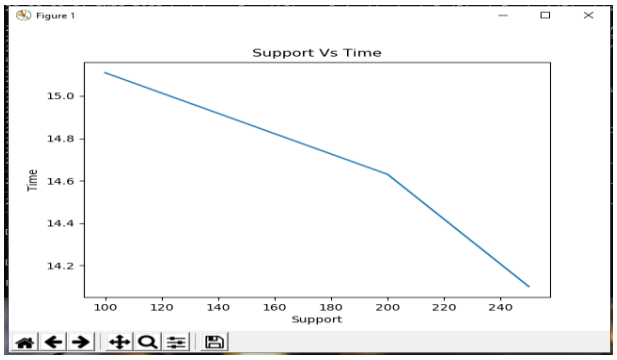

Fig 5:- Apriori Of Online Retail I Dataset

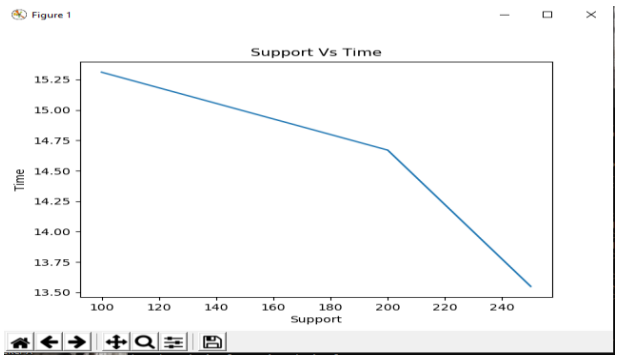

Fig 6:- Apriori Of Online Retail II Dataset

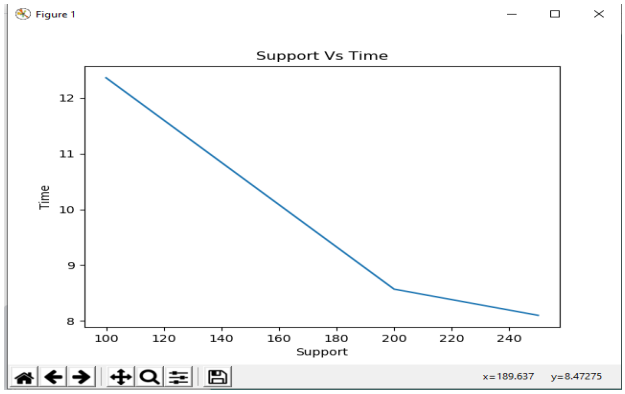

Fig 7:- Apriori Of Movie Rating Dataset

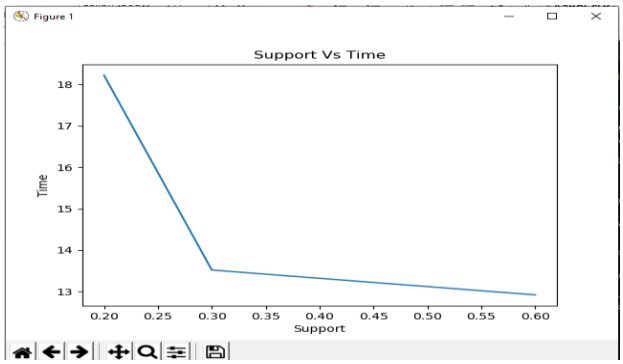

Fig 8:- FP-Growth of Instacart Dataset

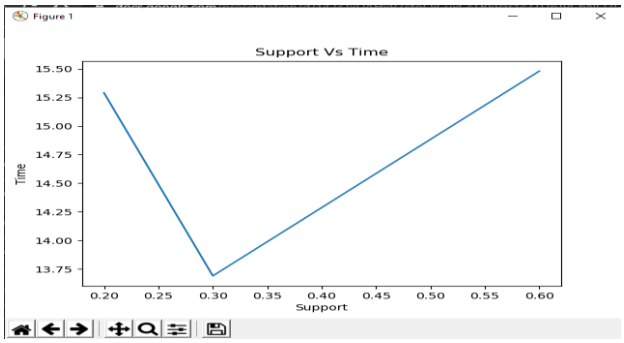

Fig 9:- FP-Growth of Online Retail I Dataset

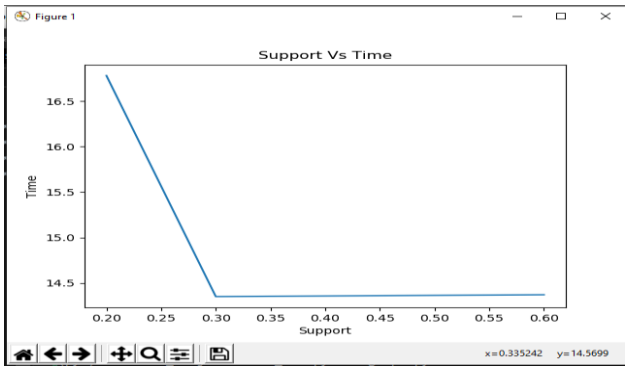

Fig 10:- FP-Growth of Online Retail II Dataset

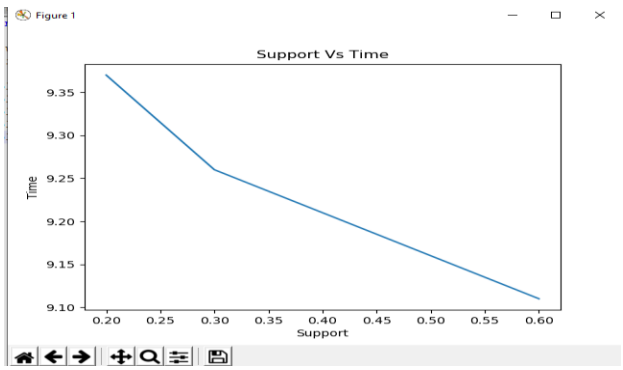

Fig 11:- FP-Growth Of Movie Rating Dataset

Acknowledgment

The authors would like to express their gratitude towards their guide, Dr. Vinaya Sawant, The Head of Information Technology Department, and the Principal, Dr. Hari Vasudevan for the opportunity provided to them to carry out implementation and research on this topic.

CONCLUSION

In the existing system, we performed the parallel Apriori algorithm based on MapReduce on the Hadoop platform, which results in the lack of implementation cost and communication cost. Using Hadoop, finding the association rules need a sufficient computation cost and memory. Thus, we execute the Parallel Apriori algorithm and FP-Growth algorithm on the SPARK platform with multiple datasets to get the results. Using, Apache Spark the performance of the results provides the fast and efficient than the Hadoop platform. During execution, this dynamic method makes very fast and efficient results for every iteration. Apache Spark architecture allows for highly parallel and distributed computing techniques. Due to its in-memory distributed computation, Apache Spark is well suited for both the algorithm for finding the frequent itemsets from the distributed data mining. Overall, both the algorithm performs admirably in the Apache Spark environment, although sequential Apriori performed poorly.

References

[1] S, Evaluation of Frequent Itemset Mining Platforms using Apriori and FP Growth algorithm, ICCM, vol. 2, pp. 1-6, (2019)

[2] Sanjay R, Arti K, Adaptive-Miner: an efficient distributed association rule mining algorithm on Spark, Springer, pp. 1-17, (2018)

[3] Singh S, Garg R, Performance optimization of MapReduce based apriori algorithm on Hadoop cluster, CEE, pp. 348-364, (2018)

[4] Diaby D, Fode C, S-FPG: A Parallel version of FP-growth algorithm under Apache Spark, IEEE, pp. 98-101, (2017)

[5] Xiujin S, Shazong C, DFPS: Distributed FP-Growth algorithm based on Spark, IEEE, pp. 1725-1731, (2017)

[6] Singh S, Garg R, Mishra P, Review of Apriori based algorithm on MapReduce framework, ICC, pp. 593-604, (2017)

[7] Zhaowei, Jianru Y, Association Rule based data imputations with Spark, CCIS, pp. 1725-1731, (2016)

[8] Xun Y, Zhang J Fidoop: Parallel mining of frequent itemsets using MapReduce, IEEE, pp. 313-325, (2016)

[9] Shaosong Y, Guoyan X, Zhijian W, The parallel Improved Apriori Algorithm research based on Spark, ICFCT, pp. 354-359, (2015)

[10] Ravi R, Performance Analysis of Apriori and FP Growth on different Mapreduce frameworks, DTU, pp. 1-7, (2015)

[11] Feng G, Yunlogo M, A Distributed frequent itemset mining algorithm based on Spark, IEEE, pp. 271-275, (2015)

[12] Sanjay R, Manohar K, R-Apriori: an efficient Apriori based algorithm on Spark, PIKM, pp. 1-8, (2015)

[13] Zhang F, Lin M, A distributed frequent itemset mining using Spark for big data analytics, CC, pp. 1493-1501, (2015)

[14] Singh S, Garg R, Performance analysis of Apriori algorithm with different data structure on Hadoop cluster, IJCA, pp. 45-51, (2015)

[15] Hongjian Q, Yihua H, YAFIM: A parallel frequent itemset mining algorithm with Spark, IEEE, pp. 1664-1671, (2014)

[16] Aavdh S, Ajeet K, Ashish M, An empirical analysis and comparison of Apriori and FP-Growth algorithm for frequent pattern mining, IEEE, pp. 1599-1602, (2014)

[17] Moens S, Aksehirli E, Frequent Itemset Mining for Big Data, IEEE, pp. 111-118, (2013)

[18] Frequent Itemset Mining Dataset Repository: http://fimi.ua.ac.be/data. Accessed 15 April 2021.

[19] Online Data Repository: https://www.kaggle.com/coldperformer/online-retail-data-v3.

[20] Online Data Repository: https://www.kaggle.com/rounakbanik/the-movies-dataset?select=ratings.csv.