JOURNAL OF MULTIDIMENSIONAL RESEARCH & REVIEW

Website: http://www.jmrr.org **Vol.2, Iss.2 Pages:01-08**

MAP REDUCE APRIORI ALGORITHM USING HADOOP FRAMEWORK IN BIG DATA

[1]A S Divyaa Lakshmi, [2] P Santhiya, [3]J Magelin Mary

[1,2]BSc Computer Science, [3]Assistant Professor
Holy Cross College, Trichy, Tamilnadu

Article Received: April 2021 Published: July 2021

Abstract

One of the most significant aspects of data mining is finding frequent itemets. Apriori algorithm is the most established algorithm for finding frequent item sets from a transactional dataset; however, it needs to scan the dataset many times and to generate many candidate item sets. Unfortunately, when the dataset size is huge, both memory use and computational cost can still be very expensive. In addition, single processor's memory and CPU resources are very limited, which make the algorithm performance inefficient. Parallel and distributed computing are effective strategies for accelerating algorithms performance. In this paper, we have implemented an efficient MapReduce Apriori algorithm (MRApriori) based on Hadoop-MapReduce model. The paper presents Map/Reduce algorithm of the legacy Apriori algorithm that has been popular to collect the item sets frequently occurred in order to compose Association Rule in Data Mining.

Keywords: Map/Reduce, apriori algorithm, Data Mining, Association Rule, Hadoop, Cloud Computing

INTRODUCTION

People started looking at and implementing Map/Reduce algorithm for most of applications, especially for computing Big Data that are greater than peta-bytes as cloud computing services are provided, for example, by Amazon AWS. Big Data has been generated in the areas of business application such as smart phone and social networking applications. Especially these days, the better computing power is more necessary in the area of Data Mining, which analyzes tera- or peta-bytes of data. Thus, the paper presents Apriori-Map/Reduce Algorithm that implements and executes Apriori algorithm on Map/Reduce framework.

The main aim of the Apriori Algorithm Implementation Using Map Reduce on Hadoop project is to use the apriori algorithm which is a data mining algorithm along with mapreduce. This is mainly used to find the frequent item sets for a application which consists of various transactions. Using this algorithm we will take the inputs from the data sets present in the application and the output is given as frequent item sets.

PROPOSED SYSTEM

The process of generating association rules is an important task in data mining. It is widely used in market basket analysis. The data used to generate association rules is usually distributed and complex. This can be effectively achieved using the Hadoop framework, because it can process large data sets with low cost and good performance. We propose an efficient algorithm MapReduce is an efficient, scalable and simplified programming model for large scale distributed data processing on a large cluster of commodity computers and also used in cloud computing [3].

BACKGROUND STUDY

1) APRIORI

Apriori is the most classical algorithm in data mining.it is used for mining frequent itemsets and relevant association rule. It is device to operate on database containing a lot of transactions, for instance, item brought by customer in a store, proposed by R. Agrawal and R. Srikant in 1994. The Apriori algorithm mines all the frequent itemsets in a transactional database, where each transaction ti contains a set of items called itemset. An itemset having k items is called a k-itemset and its length is k. An itemset X is frequent if its support, which is the fraction of transactions containing X in the database, is at least certain user specified minimum support min_sup. Let Lk denote the frequent itemsets of length k and Ck denote the candidate itemsets of length k. The Apriori algorithm joins Lk-1 to generate Ck, counts the supports of Ck, and determine the Lk in k-th database scanning. The algorithm terminates when no Ck or Lk is generated.

EXAMPLE OF APRIORI ALGORITHM

Table 1: Market Basket Transaction

TID	LIST OF ITEM_IDs
T100	11,12,15
T200	12,14
T300	12,13
T400	11,12,14
T500	11,13
T600	12,13
T700	11,13
T800	11,12,13,15
T900	11,12,13

Let us consider one database here, which consist in nine transactions where in each transaction what are the list of items that are purchased by customer, so the first transaction three items are purchased that is I1, I2, I5 the second is I2, I4…like this you have all the transactions.

Now the question is asked like find the all the frequent itemsets by using a apriori algorithm where the minimum support is given as 2 (Min.suppport count=2).

Table 2: Removal of candidate key that fails in Min-Sup count

C_1 L_1

Scan D for count of each candidate →

itemset	Sup.count
{11}	6
{12}	7
{13}	6
{14}	2
{15}	2

compare candidate support count with Minimum suppor count →

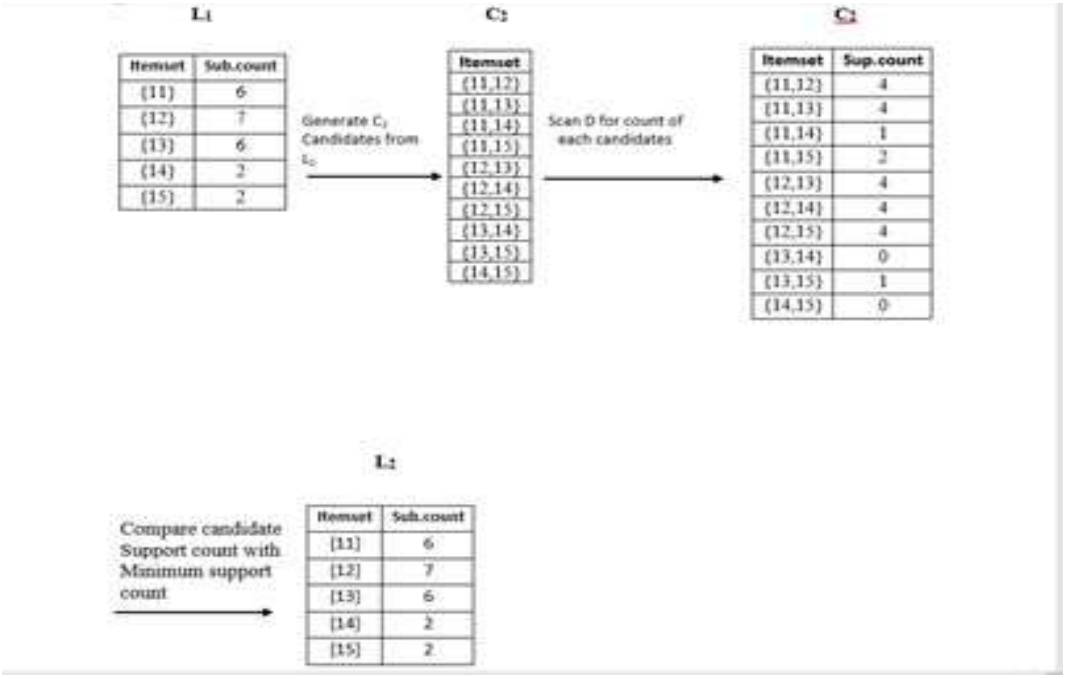
Itemset	Sub.count
{11}	6
{12}	7
{13}	6
{14}	2
{15}	2

This list is now called L1 containing the frequent data item sets. Scan D for count of each candidate. Candidate in the sense each product so, totally five product (i1, i2, i3, i4, i5).so scan the database that means you need to find count of each item, that count we call it as sup.count so far i1 how many transactions consisting of i1 you can check it the total count is i1-6 like this i2-7, i3-6, i4- 2, i5-2 this we call it as C1

Second step:

Compare candidate support count with minimum support count what is the minimum support count is given to so compare it that means compare in C1 support count of course here we call C1 as also one frequent item set. After finding support count in one frequent item sets (C1) now you need to compare with the minimum support count =2 all the items are satisfying that we need to write L1 so, it is called L1.

Table 3: Generation and Support of Candidate Lists



Generate the second candidate list using L1cross join L1. And note support counts. Remove candidate key that fail min-sup count. Generate the third candidate list L2. Now after L1 you need to find C2 – two frequent itemset. C2 means from L1 you need to make joints like (i1 i2), (i1 i3), (i1 i4), (i1 i5),(i2 i3), (i2 i4), (i2 i5),(i3 i4), (i3 i5),(i4 i5) this is two frequent itemsets so it is called C2. Now after getting C2 find again to scan the database count of each candidate next we need to write support count that is the count should be taken for each itemset that means how many transactions consisting of i1 as well as i2. {11,12}-4 like this. Next we need to write L2 again compare candidate support count with minimum support count remove the unsatisfied itemset and write remaining itemset it is called L2.

After finding L2 we need to find C3- three frequent item sets. C3 means from L2 you need to make joints like (i1 i2 i3), (i1 i2 i5), (i1 i2 i4), (i1 i3 i5),(i2 i3 i4), (i2 i3 i4), (i2 i4 i5)this is three frequent itemsets so it is called C3. Now after getting C3 find again to scan the database count of each candidate next we need to write support count that is the count should be taken for each itemset that means how many transactions consisting of i1 i2 i3. {11,12,13}-2 like this. Next we need to write L3 again compare candidate support count with minimum support count remove the unsatisfied itemset and write remaining itemset it is called L3.

2) MAPREDUCE

Mapreduce is a programming framework that allows us to perform distributed and parallel processing on large datasets in a distributed environment. It consists of two periods, map and reduce. The input data are split into small pieces and each small split of input data is delivered to a mapper for calculation, each mapper will shuffle its own intermediate output data (key, list (value)) pair to corresponding reducer. Once the reducer has collected all key-value pair, it will begin to run reduce function and calculate the output. Both map and reduce function here are specified and implemented by programmers.

HADOOP FRAMEWORK

Hadoop is an apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models the Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

BIG DATA

Big data is the science of analyzing high volumes of diverse data in near-real time (volume, velocity, variety, veracity). Massive amount of data generated are generated daily due to technological growth, digitalization and by a variety of sources, including business application transactions, web pages, videos, images, e-mails, and social media. Typically it involves using NoSQL technology and a distributed architecture to analyze the data. The analysis can be done in the public cloud or on private infrastructure. Cloud computing provides IT resources such as Infrastructure, Platform, Software, Database, and Storage as service. It provides many features like: on-demand self-service, resource pooling, rapid elasticity, flexible scaling and high availability. Big data represents content and cloud computing is an environment that can be used to perform tasks on big data.

EXAMPLE OF APRIORI-MAPREDUCE ALGORITHM

TABLE 5: Transaction Data at a Store

TRANSACTION ID	ITEMS
Transaction 1	Cracker, beer
Transaction 2	Chicken, pizza
Transaction 3	Coke, cracker, beer
Transaction 4	Coke, cracker
Transaction 5	Beer, chicken, coke
Transaction 6	Chicken, coke

Table 5 is the example transaction data at a store to explain how the proposed algorithm works. Suppose that minimum support is 2/6 that represents two items out of 6 transactions as 33%.

THE FIRST STEP

Assuming there are three Map nodes, two transaction data are distributed to three map nodes, that is, map node 1 has transaction data 1 and 2. Node 2 has transaction data 3 and 4. Node 3 has transaction data 5 and 6. Therefore, we can generate size 1 frequent items Cm1 with an item pair set <item, frequency> at the map node m.

Table 6: Map Node Vs Transaction Set Properties

Map node	**<item, frequency>**
C11	{<cracker, 1>, <beer, 1>, <chicken, 1>, <pizza, 1>}
C12	{<coke, 2>, <cracker, 2>, <beer, 1>}
C13	{<beer, 1>, <chicken, 2>, <coke, 2>}

From all Cm1, reduce nodes collect and compute C1 that is size 1 frequent item pairs and L1 that is size 1 frequent item pairs that meet minimum support. Thus, [pizza, 1] of C1 is eliminated in L1.
C1 = {[cracker, 3], [beer, 3], [chicken, 3], [pizza, 1], [coke, 4]}
L1 = {[cracker, 3], [beer, 3], [chicken, 3], [coke, 4]}

THE SECOND STEP

In the loop, L1 is mapped to each map node m for Lm1. L11= {cracker, beer}
L21 = {chicken}
L31 = {coke}
Then, size 2 frequent item pair sets can be generated by joining and sorting L1 to each item sets of the map node m as Cm2 = L1 join_sort Lm1 where the duplicated item sets are eliminated:
C12 = {<beer, cracker>, <chicken, cracker>, <beer, chicken>, <coke, cracker>, <beer, coke>}
C22 = {<chicken, cracker>, <beer, chicken>, <chicken, coke>} C32 = {<coke, cracker>, <beer, coke>, <chicken, coke>}

From all Cm2, reduce nodes collect C2 that is size 2 frequent item pairs.
C2 = {<beer, cracker>, <chicken, cracker >, <beer, chicken>, <coke, cracker>, <beer, coke>, <chicken, coke>};

C2 is mapped to each map node as Cm2 as follows: C12 = {<beer, cracker>, <chicken, cracker >} C22 = {<beer, chicken>, <coke, cracker>}
C32 = {<beer, coke>, <chicken, coke>}
Now, we can generate size 2 frequent items with an item pair set [item, frequency] at the node m that contains all transaction data as presented in Table 5

Map node	<item, frequency>
C12	{[<beer, cracker>,2], [<chicken, cracker>,0]}

C22	{[<beer, chicken>,1], [<coke, cracker>, 2]}
C32	{[<beer, coke>,2], [<chicken, coke>, 2]}

From all Cm2, the reduce nodes collect and compute C2 that is size 2 frequent item pairs and L2 that is size 2 frequent item pairs that meet minimum support.

C2= {[<beer, cracker>,2], [<chicken, cracker>,0], [<beer, Chicken> ,1], {[<coke, cracker>, 2]}, [<beer, coke>,2], [<chicken, coke>, 2]

Thus, the minimum support will be 2 so [<chicken, cracker >, 0] and [<beer, chicken >, 1] are eliminated from C2 for L2:
L2={[<beer,cracker>,2],[<coke,cracker>,2],[<beer,coke>,2][<chicken, coke>,2]}

THE THIRD STEP

In the loop, L2 is mapped to each map node m.
L12 = {<beer, cracker>, <coke, cracker>} L22 = {<beer, coke>}
L32 = {<chicken, coke>}

Then, size 3 frequent item pair sets can be generated by joining and sorting L2 to each item sets of the map node m as Cm3 = L2 join_sort Lm2 where the duplicated item sets are eliminated:
C13 = {<beer, coke, cracker>, <beer, chicken, coke>, <chicken, coke, cracker>} C23 = {<beer, coke, cracker>, <beer, chicken, coke>}
C33 = {<beer, chicken, coke>, <chicken, coke, cracker>}

As the item size k is or greater than 3, prune (Cmk) deletes non frequent item sets that violates apriori property that all subsets of a frequent item set must also be frequent:

C13 = {<beer, coke, cracker>} C23 = {<beer, coke, cracker>} C33 = {}
C13, C23, C33 eliminate <beer, chicken, coke> and C13, C33 removes <chicken, coke, cracker> as <beer, chicken> and <chicken, cracker> respectively are not a member of L2, that is, non-frequent items.
From all Cm3, reduce nodes collect C3 that is size 3 frequent item pairs and L3 that is size 3 frequent item pairs that meet minimum support:
C3= {<beer, coke, cracker>} L3= {}
Since L3 does not have any element, the algorithm ends. Therefore, we have frequent item sets L1 and L2 with size 1 and 2 respectively:
L1 = {[cracker, 3], [beer, 3], [chicken, 3], [coke, 4]}
L2 = {[<beer, cracker>, 2], [<coke, cracker>, 2], [<beer, coke>, 2], [<chicken, coke>, 2]};
The item sets L1 and L2 can be used to produce association rule of the transaction.

RESULTS AND FINDING

This paper presented the Map/Reduce algorithm of the legacy Apriori algorithm that has been popular to collect the item sets frequently occurred. Theoretically, it shows that, it provides high performance computing depending on the number of Map and Reduce nodes. In this paper what we have observe is first we have to map the items and then using apriori finding the frequent items and then reduced it. Results of this paper shows that we can easily identify the frequent itemset using map and reduced nodes.

CONCLUSION

The paper proposes Apriori-Map/Reduce Algorithm and illustrates which theoretically shows that the algorithm gains much higher performance than the sequential algorithm as the map and reduce nodes get added. It reducing the time consumed in transactions scanning for candidate itemsets by reducing the number of transactions to be scanned. The item sets produced by the algorithm can be adopted to compute and produce Association Rule for market analysis.

The future work is to build the code following the algorithm on Hadoop frame and generate experimental data by executing the code with the sample transaction data, which practically proves that the proposed algorithm works. Besides, the algorithm should be extended to produce association rule.

REFERENCES

[1]. Jongwook Woo "Apriori-Map/Reduce Algorithm"

[2]. Ming-Yen Lin, Pei-Yu Lee, Sue-Chen Hsue "Apriori-based Frequent Itemset Mining Algorithms on MapReduce".

[3]. Mohammed Al-Maolegi, Bassam Arkok "An improved apiriori algorithm for association rules".

[4]. Dawein Sun, Vincent CS Lee, Frada Burstein, Pari Delir Haghighi "An efficient vertical Apiriori Mapreduce algorithm for frequent itemset mining". 10th conference on Industrial Electronics and Applications 2015.

[5]. Jongwook Woo "Market basis analysis" Volume 3 November/December 2013.

[6]. Xinhao Zhou, Yongfeng Huang "An improved parallel Association rules algorithm based on Mapreduce framework for big data" 2014-11th International Conference on fuzzy systems and knowledge discovery.

[7]. Sudhakar sign, P.K. Mishra, Rakhi Garg "Review on Apirori based algorithm on Mapreduce framework" International conference on communication and computing ICC 2014.

[8]. A.L. Sayeth Saabith, Elankovan Sundararajan, Azuraliza Abu Bakar "Parallel Implementation of apriori algorithm on the Hadoop-Mapreduce platform" Journel Theortical and applied information technology 31st March 2016.

[9]. Jyoti Yadav, Neha Sehta "Implement Mapreduce algorithm to generate frequent itemsets" International journal of computer application (0975-8887) Volume 179, April 2018.

[10]. Madhu M.N, Kiran Kumar, Anjan K Koundinya, Srinath N.K "Mapreduce Design and Implementation of Apirori Algorithm of handling voluminous dataset" An International Journal ISSN 2229-726X, 2012.

Cite this article:

AS Divyaa Lakshmi, P Santhiya, J Magelin Mary, "Map Reduce Apriori Algorithm Using Hadoop Framework in Big Data", Journal of Multidimensional Research and Review (JMRR), Vol.2, Iss.2, pp.01-08, 2021