I J C T A, 9(17) 2016, pp. 8541-8548

An Enhanced Apriori Algorithm for finding Frequency of Items over Big Transactional Data based on MapReduce Framework

Reshma Gummadi[*], Sudhir Tirumalasetty[**] and Sreenivasa Reddy Edara[**]

ABSTRACT

Frequent Itemset Mining is well known algorithm and plays an important role in datamining. This algorithms has its own restraint of wasting time for scanning the whole database penetrating on the frequent itemsets. There are numerous enhanced apriori algorithms which vary with different characteristics and presents an improvement on apriori by reducing the time depending on scanning only some transactions. The problem was tackled previously by generating an enhanced apriori with optimal scans over transactional databases. This paper shows an approach for the improvement of apriori algorithm according to the big data environment to solve conventional problems encountered in the apriori data mining. In this paper the extension capability of MR-Apriori algorithm to be done by using map reduce framework for finding the frequency of each individual item which helps for recommendation of items can be done by knowing the frequency of itemsets.

Keywords: apriori, frequent patterns, mapreduce, MRapriori, big data.

I. INTRODUCTION

Data mining is the essential process of discovering hidden patterns from large amount of data . This large datasets means data may reach to more than terabytes. Apriori algorithm is one of the most prevalent and dominant algorithm which is used to discover frequent patterns in mining[1] It is the most popular technique that has been deliberated by researchers. Extracting association rules is one of the core of the process of data mining. Finding the association rules in large datasets of many number of transactions between items which is important field of the research . Data mining is a multidisciplinaryeld[2].Hence the research of association rules are more helpful in many number of applications such as banking, market analysis, health care and manufacturing etc.

The advantage of mapreduce framework can be effectively used for identifying frequent itemsets. Hence improved apriori can be implemented in hadoop using mapreduce framework.

II. MAPREDUCE FRAMEWORK

MapReduce is a programming model. The programming model consists of two functions map, reduce[3]. map function defines a mapping from one set of key value pairs to reduce function. These functions are obvious to the size of the cluster that is requested. So they can be used unchanged for a small dataset or large dataset.

Hadoop is an opensource software framework for storing data and running applications on large clusters. It provides massive storage for any kind of data. Hadoop provides a distributed file system and a framework for the analysis on very huge datasets using mapreduce. Map reduce is an algorithm used to

[*] Department of Information Technology, Prasad V Potluri Siddhartha Institute of Technology, Guntur, Andhrapradesh 521134

[**] Department of Computer Science & Engineering, Vasireddy Venkatadri Insitutute of Technology, Guntur, Andhrapradesh 522508

solve the problems to analyze big data, defined as more than petabytes of data in distributed computing environment.

The MapReduce framework is proved to be reliable low cost, used by Yahoo and google on clusters of tens of thousands of machines. The basic function of map reduce model is it takes the input, calculate key value pairs from each pair of input, group all intermediate values by key, then continue this process over the resulting groups and finally reduce each group. The mapreduce implementations deals with the issues such as load balancing, fault tolerance etc.

For finding frequency of each item, when the data set is large

Algorithm

Datasource mapped to all map nodes;

C_k: candidate Itemset of size k

Min_support=(minsupport percent*max no of transactions)/100

L_k: Frequent itemset of size k

MRApriori(T,s)

L_1<-{large 1-itemset that appear in more than or equal to transactions}

For each candidate c in C_k

L_k= Ø

Map(T_id,I);

Reduce(I,values);

If($|T_{id}|$>=s)

then

L_k<-L_k{c}

End if

End for

K<-k+1

Return L_k

End apriori

/*Mapper for phase 1 would emit a <itemid,1>pair for each item across all transactions.*/

Map(Transction_id,item)

{

for(k=1;L_k!=˜;k++)

{

C_{k+1}=L_k join_sort(L_k)

Write(I,one);

}

}

/*Reducer collect all the emitted itemid keys from the 3 mappers and aggregate it to return the count for each itemid*/

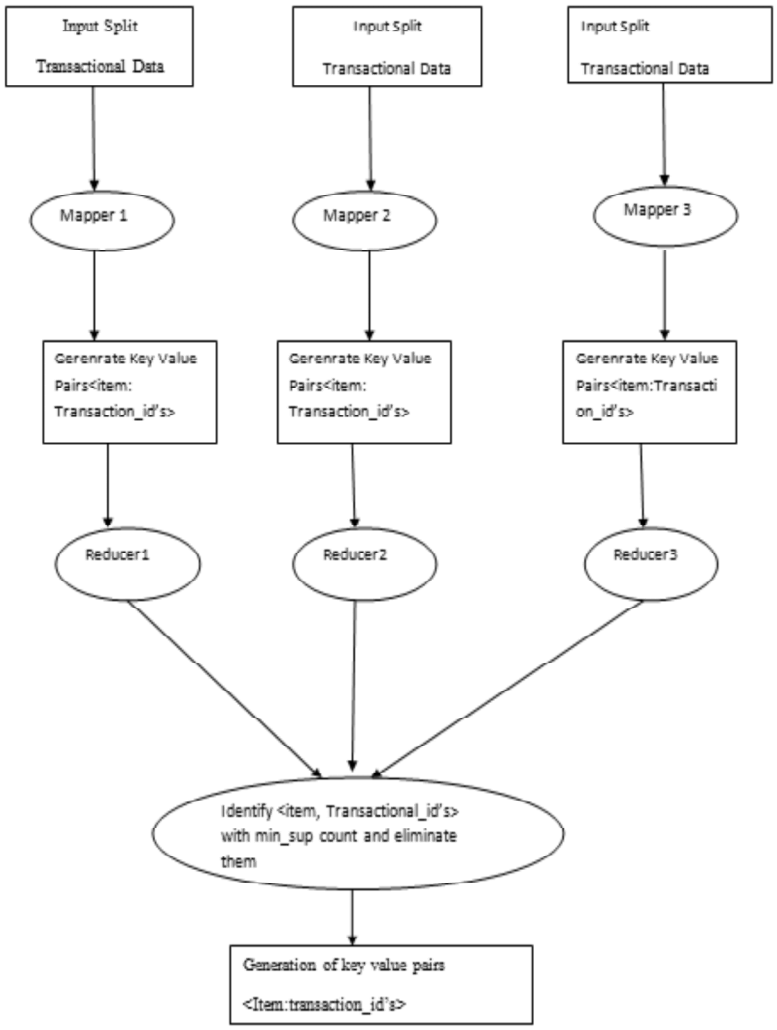
Reduce(Itemset,values)

If (hasMinsupport(minsuppercent, numof transactions, countItemId)

{
Write(itemids,countItemId)
}

Block diagram of applying mapreduce1 on transactional data

In the input split data can be sent as Transaction_id: The list of items purchased under T1



The output generated from the above mapReducer1 phase is sent to the input for mapreducer2 phase,

Here input data is sent like the corresponding individual Items belongs to the number of transaction_id's in which it contains. .i.e Item:Transaction_id's

Block Diagram of applying Map Reduce2 on transactional Data

Example

Consider the transactional database with some transactions and items purchased in each transaction can be represented with Item id's. This data set consists of 9 transactions and five items. Assume the value for minimum support is 3.

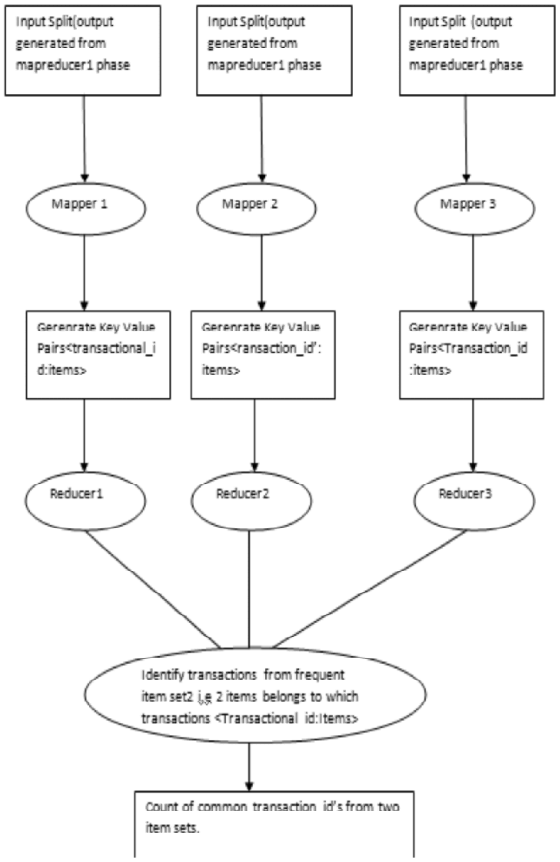


Table 1
Dataset of items relative to transactions

Tid	*Items*
T1	I1,I2,I5
T2	I2,I4
T3	I2,I4
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Explanation

The given data set is spitted in to three input files and each input split consists of the following transactional data sets.

Mapper1:

T1	I1,I2,I5
T2	I2,I4
T3	I2,I4

Mapper 2:

T4	I1,I2,I4
T5	I1,I3
T6	I2,I3

Mapper 3:

T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

The mapper phase is executed and the following <key,value>pairs are generated

,,,Mapper1 produces	Mapper2 produces	mapper 3 Produces
<I1:T1>	<I1:T4,T5>	<I1:T7,T8,T9>
<I2:T1,T2,T3>	<I2:T4,T6>	<I2:T8,T9>
<I4:T2,T3>	<I3:T5,T6> <I3:T7,T8,T9>	
<I5:T1>	<I4:T4>	<I5:T8>

These pairs are go to the reducer phase and generates the following output for reducer 1

<I1:T1,T4,T5,T7,T8,T9>

<I2:T1,T2,T3,T4,T6,T8,T9>

<I3:T5,T6,T7,T8,T9>

The output of reducer2 are

<I4:T2,T3,T4>

<I5:T1,T8>

From these two identify the number of transaction_ids with minimum super count and eliminate them .in the above example <I5:T1,T8> is eliminated because it contains only two transactional ids.

The output for mapreducer1 is given as input for mapreducer2

Input for MapReduce2:

<I1:T1,T4,T5,T7,T8,T9>

<I2:T1,T2,T3,T4,T6,T8,T9>

<I3:T5,T6,T7,T8,T9>

<I4:T2,T3,T4>

The mapper phase is executed and the following <key,value>pairs are generated

Mapper1 produces

<T1:I1,I2>

<T2:I2>

<T3:I2>

<T4:I1,I2>

<T5:I1>

<T6:I2>

<T7:I1>

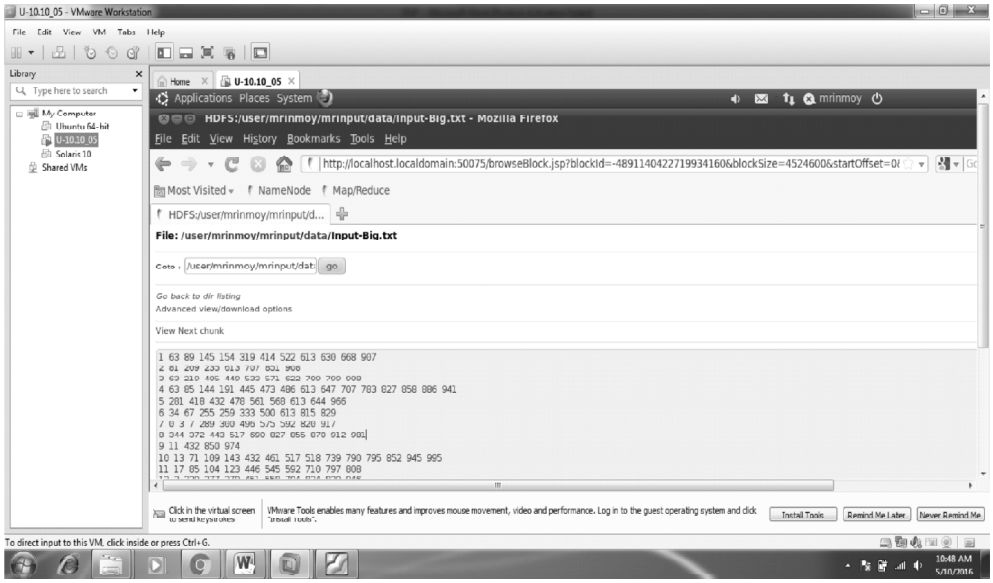
<T8:I1,I2>

<T9:I1,I2>

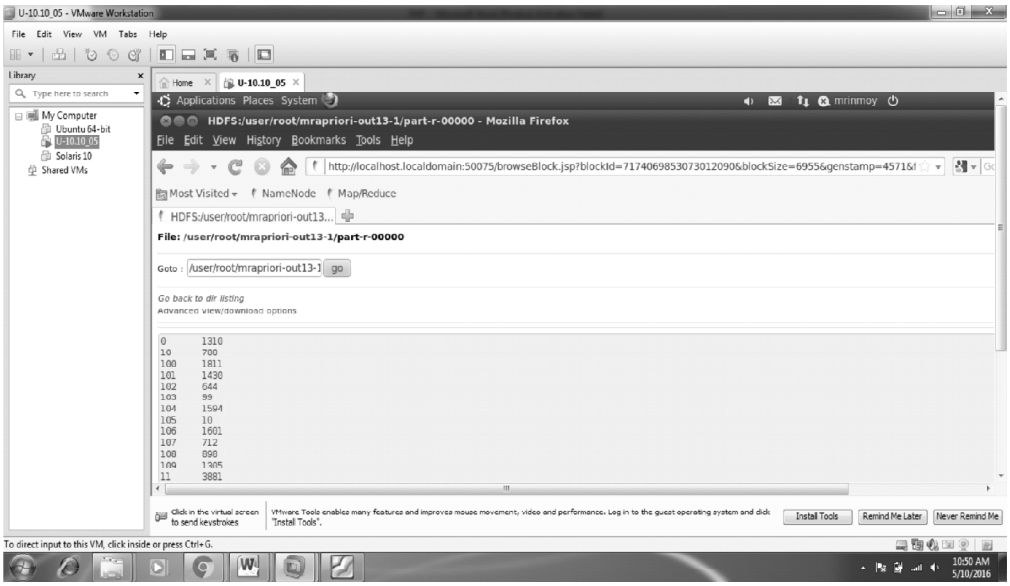
Frequent item sets are generated by map reducer phase 2..identifying frequent items using map reduce helps in accuracy of getting each item in count wise. As it is executing in hadoop platform it helps to retrieve accuracy of items in huge number of transactions. And it leads to minimum scanning time for the evolution of frequent items.

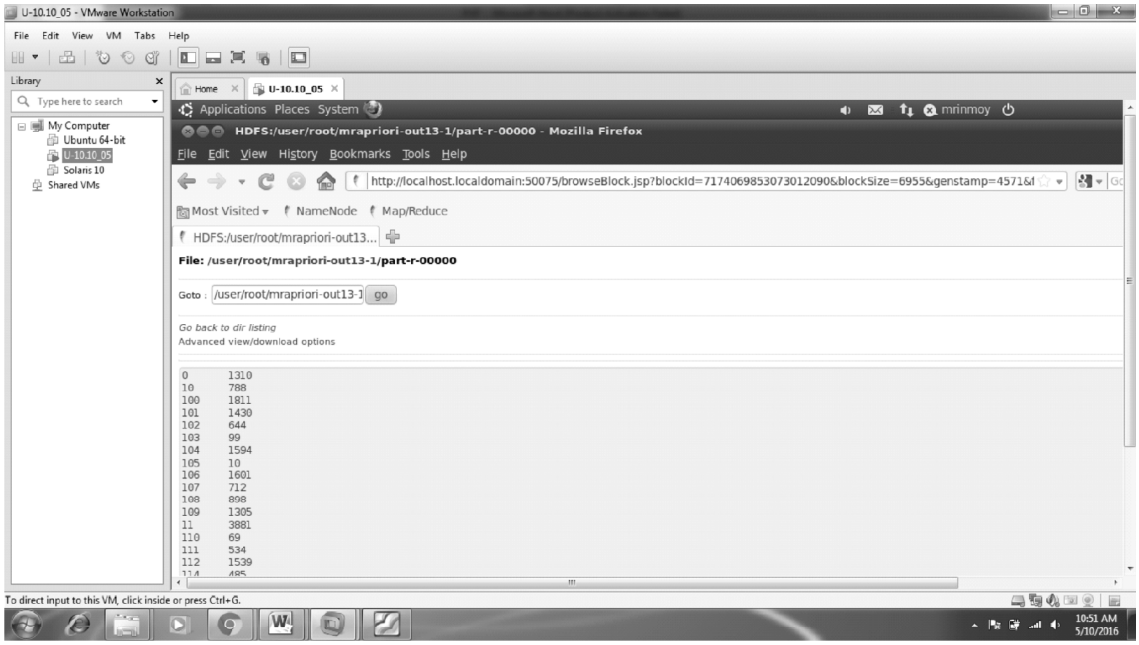
The snapshots of input and output are specified below.

Screen Input Data Set



Output





III. PERFORMANCE EVALUATION

The data stored in hadoop file system on a processor speed, storage and performance are the major factors to be considered for big data. When large datasets are implanted on this platform, there was a small difference in performance and speed. Initially, the set of 1000 transactions moved to HDFS. The dataset can be of any format and any size. From hadoop distributed file system the data set is extracted. By executing the map function and reduce function the output will be generated by representing the frequency of each item among large number of transactions. As shown in the output there are two columns in which first column represents Item id where as second column represents frequency of that particular Item. This data will help for the usage of items with higher frequency can be selected among large number of items. The execution time for the whole process is about 13 seconds. The total time taken by all mappings and reducers in occupied slots is in milliseconds

IV. CONCLUSION

The proposed method reduces number of scans and at the same time the frequency of each item is calculated and resulted for further recommendations of item from any number of transactions. This represents that hadoop environment is more efficient in generating frequency of item sets. Based on the dataset size the performance and execution may vary. In this execution the performance is high and execution time is very less.

REFERENCES

[1] Sudhir Tirumalasetty,Arun jadda and Sreenivasa Reddy Edara,"An Enhanced apriori Algorithm for discovering Frequent Patterns with Optimal Number of Scans" in IJCSI,2015.

[2] Data Mining: Concepts and Techniques Jiawei Han and Micheline Kamber Simon Fraser University

[3] Apriori-Map/Reduce Algorithm Jongwook Woo Computer Information Systems Department California State University Los Angeles,

[4] B. Mobasher, N. Jain, E.H. Han, and J. Srivastava, "Web Mining: Pattern Discovery from World Wide Web Transactions", Department of Computer Science, University of Minnesota, March 1996, Technical Report TR96-050.

[5] C. Ordonez, and E. Omiecinski, "Discovering Association Rules Based on Image Content", IEEE Advances in Digital Libraries, 1999.

[6] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, in ACM SIGMOD International Conference on Management of Data, May 1993, Washington, USA, pp. 207216.

[7] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, “Fast discovery of association rules. In Advances in Knowledge Discovery and Data Mining”, AAAI Press, 1996, pp. 307328.

[8] R. Bayardo, and R. Agrawal, “Mining the most interesting rules”, in 5th International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999, California, USA, pp. 145154.

[9] L. Klemetinen, H. Mannila, P. Ronkainen, et al., “Finding interesting rules from large sets of discovered association rules”, in Third International Conference on Information and Knowledge Management.