IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

A Survey on Parallel Mining of frequent itemsets in MapReduce

Indumathi S[1], Kavya D S[2], Madhusudhan V[3]

M.Tech, Computer Science and Engineering, REVA ITM, Bangalore, India[1,2,3]

Abstract: This paper shows the various parallel mining algorithms for frequent itemsets mining. We summarize the various algorithms that were developed for the frequent itemsets mining, like candidate key generation algorithm, such as Apriori algorithm and without candidate key generation algorithm, such as FP-growth algorithm. These algorithms lacks mechanisms like load balancing, data distribution I/O overhead, and fault tolerance. The most efficient the recent method is the FiDoop using ultrametric tree (FIUT) and Mapreduce programming model. FIUT scans the database only twice. FIUT has four advantages. First: I reduces the I/O overhead as it scans the database only twice. Second: only frequent itemsets in each transaction are inserted as nodes for compressed storage. Third: FIU is improved way to partition database, which significantly reduces the search space. Fourth: frequent itemsets are generated by checking only leaves of tree rather than traversing entire tree, which reduces the computing time.

Keywords: MapReduce, frequent itemsets, mining algorithm, ultrametric tree.

I.INTRODUCTION

Data mining is a process of discovering the pattern from the huge amount of data. There are many data mining technics like clustering, classification and association rule. The most popular one is the association rule that is divided into two parts i) generating the frequent itemset ii) generating association rule from all itemsets.

Frequent itemset mining (FIM) is the core problem in the association rue mining. Sequential FIM algorithm suffers from performance deterioration when it operated on huge amount of data on a single machine.to address this problem parallel FIM algorithms were proposed.

There are two types of algorithms that can be used for mining the frequent itemsets first method is the candidate itemset generation approach and without candidate itemset generation algorithm. The example for candidate itemset generation approach is the Apriori algorithm and for, without candidate itemsets generation is the FP-growth algorithm.

The important data-mining problem is discovering the association rule between the frequent itemset.in order to find best method for mining in parallel, we explore a spectrum for trade-off between computation, synchronization, communication, memory usage. Count distribution, data distribution, candidate distribution are three algorithms for discovering the associate rule between frequent itemsets.

Minimizing communication is the focus of the count distribution algorithm.it will thus even at the expense of winding up redundant duplication computation in parallel.

The data distribution effectively utilizes the main memory of the system.it is communication-happy algorithm. Here nodes to all other nodes broadcast the local data.

The candidate distribution algorithm for both, to segment the database upon the different transaction support and the patterns, exploits linguistics of a particular problem. Load balancing is also incorporated by this algorithm.[1]

II. RELATED WORK

Sandy moen's at al,[2] proposed two new method for mining frequent itemset in parallel on the Mapreduce framework

First method is the Dist-Eclat. This method distributes the search space evenly as possible among mapper. This technique mines large dataset but not massive datasets. This algorithm operates in three steps:

We use vertical database rather than transaction database.in the first step the vertical database is divided into equal sized blocks called shards and distributed to available mappers. Each mapper extracts the frequent singletons from each block and gives to the reducer. The reducer collects all the frequent tested. In the second step the set of frequent itemsets of size K ae generated (Pk). Frequent singleton itemsets are distributed to the mappers. Each mapper runs Éclat [3] to find frequent K-sized superset of items. The reducer collects all the frequent K-sized supersets of items and distributes it to the next batch of mappers. Round Robin is used for the distribution of the frequent itemset. The third step is the mining the prefix tree. The mutual information between the mappers are independent, so mapper complete each step independently.

Demerit:

This method returns a very large number of sets so this method is prohibited on Hadoop

Second method is the BigFIM over the problem of Dist-Eclat. There are 3 steps in this method

In the first K-FI's are generated using breath-first method.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

Every mapper takes the database and gives itemsets for which, we want to know the support .the reducer takes all itemsets and returns only the global frequent itemsets. These itemsets are considered as candidates and distributed to the mappers for breath-first search. This process continues K-times to generate K-FI's.next step is computing the possible extension.

The mapper gives local Tid-list to the reducer the reducer combines the local Tid-lists, to one Tid-list, and assigns prefix to mappers. The mapper in the final step works on individual prefix group. A prefix group fits in the memory as a conditional database. The diffsets are used to mine the frequent itemsets in the conditional database.

Enhilvathani et al [4] have used the Apriori algorithm for frequent item set generation on mapreduce programing model. For implementation of algorithm is given in five steps

In the first step the transaction dataset is partitioned that is Divided into n subsets done that are of map phase.in the second step the data subsets are formatted as <key1, value1>pair, key is Tid(Transaction id).

The mapreduce task is executed in third phase. The record of input item subsets are scanned by the Map function and candidate item sets input are generated by the map function

In the fourth step the output of the map function combined by combiner function in the local and it outputs <itemset, support count>, the intermediate pair generated by combiner function is divided by partition function in to "r" different partitions. Finally reduce function executes the reduce task, the key item set are sorted. In the supported count of the same candidates is added by reduce function to get the actual support count of the candidate in the transaction database. Compare with the minimum support count to gets the frequent item set Lp.

Demerit

Apriori algorithm has to scan the entire database repeatedly.

Suraj Ghadge et al [5] gives the overview of FP-Growth algorithm. The idea is to build one FP- tree, and divide into many parts and distribute them to different threads.

The FP-algorithm is divided into two parts:

1) Building FP-tree

Given minimum support and transaction database, the database is scanned once. All frequent itemsets(F) along with the itemset support is collected , the F is sorted in support-desenting order as FList(frequent itemset list0.

The root of FP-tree is labelled "null", for each of transaction in database do the following.

2) mining from FP-tree

Iterative procedure: set of conditional pattern base is produced in each step and calculated together Procedure FP Growth (Tree, α)

Algorithm:

Begin

/*Mining single prefix−path FP−tree */ if Tree contains a single prefix path then Begin

Let P be the single prefix−path part of Tree;

Let Q be the multipath part with the top branching node Replaced by a null root;

For each combination (denoted as β) of the nodes in the path P do

Generate pattern βUα with support = minimum support of Nodes in β;

Let freq pattern set(P) be the set of patterns so generated; End

Else let Q be Tree;

/* Mining multipath FP−tree */

For each item ai in Q do Begin

Generate pattern β = aiUα with support = ai.support; Construct β's conditional pattern−base and then β's Conditional FP−tree Tree β;

If Tree β = φ(13) then call FP−growth(Tree β, β);

Let freq pattern set(Q) be the set of patterns so generated; End

Return (freq pattern set(P)Ufreq pattern set(Q)U(freq Pattern set(P)× frequent pattern set

Demerits:

It still consumes large amount of time to operate on large data

Yaling Xun et al, [6] proposed an ultrametric tree for mining frequent iteam set ultrametric provides four advantages ones FP and Apriori like partitioning a database in a minimizing input output and compressed storage. FiDoop is algorithm designed to overcome the problems like load balancing, fault tolerance, automatic parallelization and distribution on large cluster.

Comparing with ultrametric tree algorithm FiDoop has many other features. In FiDoop the concurrently and independently decomposes itemsets small ultrametric tree is constructed by the reduces to perform combination operation as Well as mining these trees in parallel.

The FIUT algorithm consist of two phases. Two round of scanning is done in the first phase.

In the first phase support of all item set are computed and frequent item set are generated. In second scan infrequent item set are pruned in each transaction record, resulting in k-item set [K_ number of frequent items]. In phase two n item set are decomposed in to K-FIU tree, where K+1<=n<=M (M is the maximal value of K), and unioning original K-item sets. Mining of are frequent k item sets based on the leaves of K-FIUtree is done in the second phase without traversing the tree. The computing time is reduced in FIUT.

DOI 10.17148/IJARCCE.2016.52109

There are three steps in FiDoop

In transaction database stored in HDFC in the form of multiple input files in the form of < Long writable offset, Text record >, the frequency of items is computed by the mapper and mapper generates local one item set the local one item set from different mapper are sorted and merged given to reducer as global one item set the minsupport is applied to prune infrequent item set .The output from first map reduce is stored in the form of < Text item, long writable count >, stored in file named F-file this is given as input to the second mapreduce job FiDoop.

One item set from the first mapreduce job is given to the second mapreduce; second round of scan is applied to prune infrequent item set. The mapper returns a pair < Array Writable item sets, Long Writable One >, as on output that is shuffled and combined for the second job's reducer. After combination, reducer gives key value pair. The output of second mapreduce is < In Writable item number, Map Writable < Array Writable k-item, Long Writable Sum >>.

The third mapreduce is imported and expensive phase it is dedicated to 1) K-FIU tree construction 2) item set decomposition 3) frequent item set mining. The K-item sets given by the second mapreduce is decomposed into a list of small –sized sets by the mapper, this decomposition result is merged to construct FIU tree. The decomposition process is performed in parallel on each mapper. The map function returns <key, value> pair key is number of items value is FIU tree. Contains leaf and non-leaf node includes node –link and item name leaf node includes support and item name by their single reducer can be given item set with same number of items. Constructing K2-FIU and mining all frequent item sets without traversing the tree is done by the reducer

III. CONCLUSION

Mapreduce programming model is applied for existing parallel mining algorithm for mining frequent itemsets from database and solves the load balancing and scalability.

This paper gives the overview of algorithms designed for parallel mining of frequent itemsets .The Apriori and FP tree algorithm were used for mining frequent itemsets. Main drawback of Apriori algorithm is that the database has to be scanned many number of times and huge candidate keys needs to be exchanged between the processor. I/O and synchronization are the other problems in the Apriori algorithm.

The disadvantage of FP-growth, however, lies within the impracticableness to construct in-memory FP trees to accommodate large-scale databases. This drawback becomes a lot of pronounced once it comes to huge and two-dimensional databases.

To overcome these problems, FiDoop, an parallel frequent itemset mining algorithm is developed. FiDoop incorporates the ultrametric tree (FIU) rather than Apriori or FP-growth algorithm. The FIU tree achieves compressed storage. FiDoop runs three MapReduce jobs. The third MapReduce job is important. in third job the mapper independently decomposes itemsets and reducer built the ultrametric trees.

REFERENCES

[1] "Parallel Mining of Association rule." Rakesh Agarwal ,John C Safer

[2] "Frequent Itemset Mining for Big Data Sandy Moens, Emin Aksehirli and Bart Goethals Universiteit Antwerpen, Belgium

[3] "ECLAT Algorithm for Frequent Itemsets Generation "Manjit kaur , Urvashi Grag Computer Science and Technology, Lovely Professional University Phagwara, Punjab, India . International Journal of Computer Systems (ISSN: 2394-1065), Volume 01– Issue 03, December, 2014 Available at http://www.ijcsonline.com/

[4] "Implementation Of Parallel Apriori Algorithm On Hadoop Cluster" A. Ezhilvathani1, Dr. K. Raja. International Journal of Computer Science and Mobile Computing

[5] "Frequent Itemsets Parallel Mining Algorithms " Suraj Ghadge, Pravin Durge, Vishal Bhosale,Sumit Mishra. Department of Computer Engineering, JSPM's ICOER. International Engineering Research Journal (IERJ) Volume 1 Issue 8 Page 599-604, 2015, ISSN 2395-1621

[6] "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce" Yaling Xun, Jifu Zhang, and Xiao Qin, Senior Member, IEEE

.