Research and Improvement of Apriori Algorithm Based on Hadoop

Gao Pengfei [a], Wang Jianguo [b] and Liu Pengcheng [c]
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, Shaanxi Province, China
[a]gaopf1225@gmail.com, [b]wjg_xit@126.com, [c]294843945@qq.com

***Abstract*—Association rules can forcefully get a horizontal relation in the big data, the Apriori algorithm is one of the most significant association rules. Traditional mining based on parallel Apriori algorithms needs much more time in data IO with the increasing size of large transaction database. This paper improves the Apriori algorithm from compressing transactions, reducing the number of scans and simplifying candidate set generation. And then the improved algorithm is parallelized on the Hadoop framework. The experiments show that this improved algorithm is suitable for large-scale data mining and has good scalability and effectiveness.**

Keywords-Apriori algorithm; Hadoop; Association rules

I. INTRODUCTION

In the context of the development of big data "spraying wells", there is frequently a close relationship between vast amounts of data[1]. Analysis and decision making through data mining have become the mainstream of social development. In order to better find the relevance of transaction data sets, some researchers have discovered the concept of association rule mining technology[2]. With the attention of many researchers at home and abroad caused by the conception of the concept, they have done a lot of analysis in this field and put forward many data mining algorithms.

One of the most famous association rule algorithms is the Apriori algorithm, which is a classic association rule algorithm designed by Agrawal[3-4] in 1994. It is a level-by-level search iteration method that constructs a k-item set to constitute a k+1-item set. The main ideas of this algorithm are: Firstly, all frequency sets are counted from the transaction database, and the support of this frequent set must not be less than the minimum support degree; Secondly it enters into the process of strong association rule generation, and the rules need to satisfy the support and confidence thresholds at the same time; Thirdly, only all rules that contain collection items are retained. Once these rules are retained and generated, that are greater than or equal to the MinConfidence.

The design of the Hadoop[5] framework originated was from an open source project developed by the Apache organization Foundation. Because of its inter-temporal significance, the Hadoop framework has been widely used in the information field at home and abroad. There are two important modules in the Hadoop frame--Distributed File System HDFS and Distributed Computing Frame MapReduce[6]. As a distributed file system , HDFS functions aims to implement data storage. It will work in conjunction with the computational framework. MapReduce works to provide the underlying support for data calculations; And the idea of MapReduce[6-7] is based on a paper by Google. In short, its core method is "the decomposition of tasks and the statute of results."

II. BRIEF AND RESEARCH STATUS OF APRIORI ALGORITHM

A. Overview of Apriori algorithm

The Apriori algorithm is a level-by-level search iterative method that consists of a k-item set to construct a (k+1)-item set. First, obtain a frequent 1-item set. L1 can generate a frequent 2-item set L2, and L2 can generate a frequent 3-item set L3. According to this rule, when a frequent k-item

set cannot be found, the algorithm ends[8-9]. The specific operation is as follows:

1) Iterate through the initial transaction database and count the frequency of occurrence of the candidate set. The result is the support of the project. All projects whose all supports level no lower than the preset threshold generate a frequent 1-item set L1.

2) The algorithm uses L1 JOIN L1 to form a candidate C2-item set C2.

3) Using the items in C2, traverse the database again to obtain the support degree of each candidate set. All projects with support levels not lower than the support level generate frequent 2-item set L2.

4) The algorithm uses L2 JOIN L2 to form a set C3 of candidate 3-item sets.

5) Using the items in C3 to traverse the database again, the support degree of each candidate set can be obtained. All items with support levels not lower than the support level generate frequent 3-item set L3.

The above process is performed iteratively until the candidate set C k is empty. The Apriori algorithm does multiple IO operations on the database. Each stage consists of two parts, namely connection and pruning.

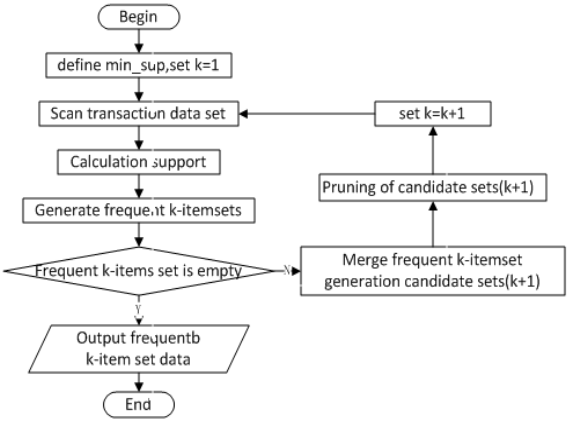

Figure 1. Apriori flow

B. Apriori algorithm instance analysis

Original transaction T10={A,C,D}, T20={B,C,E}, T30={A,B,C,E}, T40={B,E}. Suppose min_sup=2. Then L1={{A},{B},{C},{E}}, L2={{A,C},{B,C},{B,E},{C, E}}.

1) Self join, C3=L2×L2={{A,C},{B,C},{B,E}, {C,E}} × {{A,C},{B,C},{B,E},{C,E}}={{A,B,C},{A,C,E},{B,C,E}}

2) Pruning, Any frequent item set, its subset must also be frequent. For Candidate Set C3, clearing those subsets with infrequent options: The two items subset of {A,B,C} are {A,B},{A,C},{B,C}, where {A,B} is not an element of L2, so remove this option.; The two items subset of {A,C,E} are {A,C},{A,E},{C,E}, where {A,E} is not an element of L2, so remove this option; All the two items subset generated by {B,C,E} are {B,C},{B,E},{C,E}, the subsets produced by {B,C,E} all satisfy the requirements of L2. Therefore, this option is not deleted.

3) In this way, C3={{B,C,E}} obtained after pruning.

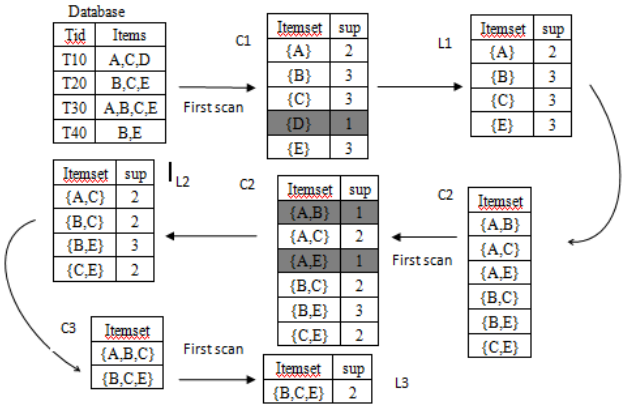

Figure 2. Apriori algorithm execution process

C. The shortcomings of Apriori algorithm

1) When the Apriori algorithm generates the candidate item set, it needs to perform the self-connection operation on the frequent itemsets obtained in the previous step. Then scan the transaction data set again and compare the candidate set formed by the self-connection with min_sup. During the self-connection operation, a large amount of comparison work will be performed.

2) Apriori algorithm need to rescan transaction datasets before pruning, and then compare with min_sup. Therefore, when the size of the transaction dataset is getting larger and larger, each scan will consume a lot of time, resulting in inefficient mining.

3) In the current situation where the data information has a high dimension and the type is complex, the classical Apriori algorithm can't satisfy users.

4) Because the classic Apriori algorithm is only applicable to a single machine, when the size of transaction data sets gradually becomes larger and larger, it will lead to inefficient mining, insufficient storage space, and even system crashes.

III. Parallel MEC-Apriori Algorithm Based on MapReduce Apriori algorithm

A. *Reduce frequent item sets self-connection comparison times and pruning steps*

In the processing of candidate sets, a method of transaction compression characteristics has been introduced. That is, according to the n-dimensional data item set, if itself is not a frequent item set, then the n-1 dimensional subset of the n-dimensional data item set is also not a frequent item set. Therefore, in the mining of candidate sets in the transaction database, the number of candidate candidate sets is compared and deleted because of the method of transaction compression characteristics, so that the number of candidate sets is gradually reduced, and the time efficiency of mining frequent itemsets is improved.

B. *Reduce the Number of Scanned Databases*

When mining frequent itemsets, the original transaction database is converted into a vertical data table, and then scan the vertical data table to mine frequent itemsets, because only one transaction database was scanned, a problem with frequent I/O was solved to some extent.

C. *Combining Apriori Algorithm and Hadoop Platform*

With the ever-increasing size of data, the traditional Apriori algorithm has been difficult to support its massive database of transactions. The solution to this problem is to add the Hadoop distributed platform to the Apriori algorithm[10], which not only makes the traditional Apriori algorithm run more efficiently, but also eases the storage pressure of the transaction database.

1) Generate frequent itemsets

The flow chart in this stage is shown in Figure.2

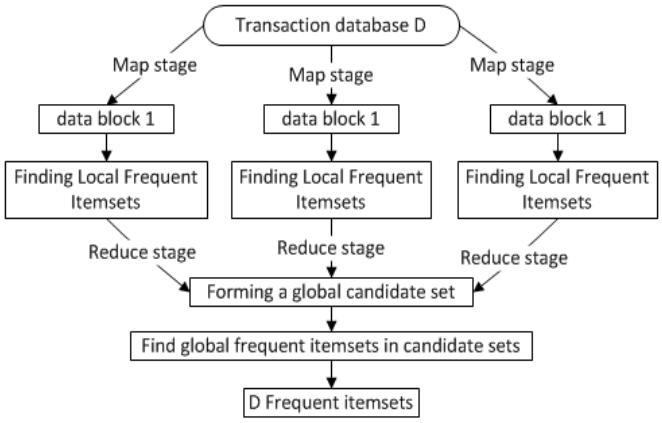

Figure 3. Generate frequent itemsets flow

a) The way of data blocks formatting

For the function of interface named Input Format implements Record Reader(Interface) is to convert data blocks into key-value pairs, eg: <key1,value1>.

b) Perform Map task

The idea of the first step is to generate the frequent item sets of each block.

c) Perform Reduce tasks

The key-value data output by the Combiner function is used as the input data of the Reduce phase. After a series of merging processes, some frequent item sets of the data module are obtained as a global candidate item set.

d) Scan *transaction* data set D

Call the Map function to rescan the formed global candidate frequent item set, and self- join , compare the minimum support count with the set of transaction items formed by the self-join, If it is less than the minimum support, then the last local frequent itemset is the final global frequent itemset, then pass it to the Reduce function and summarize it. Instead, it is necessary to iterate the local frequent itemsets until a frequent itemset is generated.

2) Generation of association rules

After association rules mine frequent item sets, it is necessary to generate strong rules. The emergence of strong rules is shown in Figure.3:

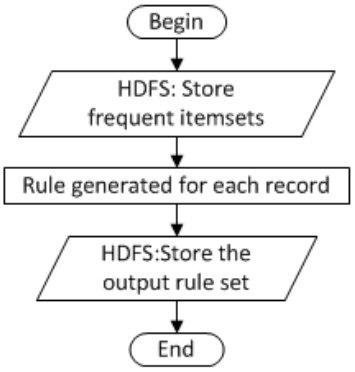

Figure 4. Generate strong rules flow

a) In the transaction dataset that holds the text, the input data of the Map must exist in the form of a key-value pair, so each row of data can be treated as a transaction. The key in a key-value pair is the offset of each row of data, and the value is represented as this row of data.

b) These key-value pairs are used as the input of the Map function, and then a set of frequent items conforming to the actual situation is obtained according to the set support threshold.

c) The output of Combiner function in Map stage is used as the input data of Reduce stage, then it is processed according to the local frequent itemsets generated in Map stage, and finally the strong association rules of the output are stored in HDFS.

IV. Experimental Assessment and Analysis

A. Setting Up a Hadoop Cluster Environment

The size of a Hadoop cluster is arbitrary. A small cluster can consist of a NameNode and several DataNodes. And a large cluster can consist of a NameNode and hundreds of DataNodes. Local mode, pseudo-distribution mode and fully-distributed mode are three modes built by Hadoop clusters. Considering the hardware configuration problem, This paper choses to use a virtual machine to set up a cluster environment, and the number of nodes in the cluster is 3, as shown in Figure.4

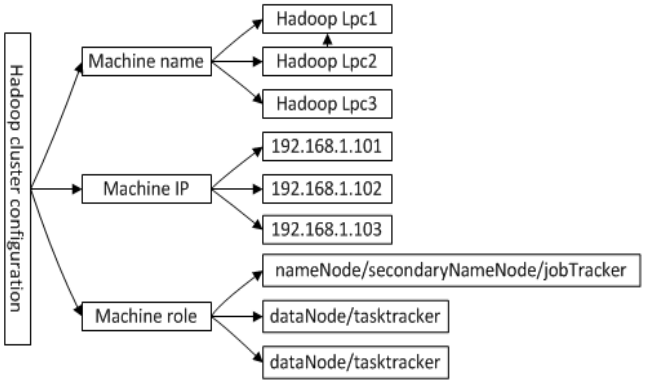

Figure 5. Build a cluster environment

B. Data Comparison Experiment

1) UCI experimental data

This experiment selects the retail file in the UCI database (association rules to study the classic data set) as the experimental transaction data set. By comparing the MEC-Apriori algorithm with the traditional Apriori algorithm, the results show that the time performance of the MEC-Apriori algorithm has been greatly improved in the mining of frequent itemsets and candidate itemsets, thus verifying the efficiency and feasibility of the improved algorithm.

2) Implementing the MEC-Apriori Algorithm Model

First, the experimental data set in the file retail is selected, and the data set in the retail is mined using the new MEC-Apriori algorithm, and then the association rule is obtained according to the user-defined support degree and the confidence threshold.

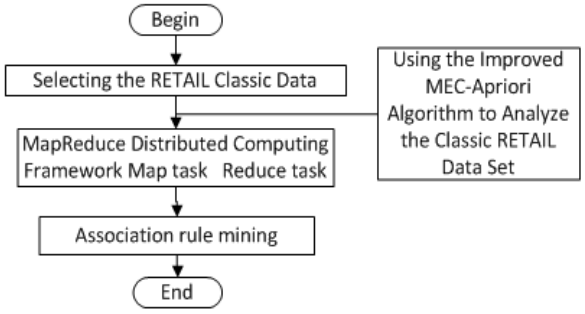

Figure 6. Simulation flow

3) Experiment content and result analysis

Experiment 1: Performance Comparison between Single Apriori Algorithm and MEC-Apriori Algorithm

The transaction data set for this experiment is stored as a file, Performance analysis of mining time before and after improved with 3 nodes Hadoop cluster test algorithm. First, on the premise that the number of nodes in the Hadoop cluster is unchanged, continuously increase the number of item sets in the experimental data item set, and set the minimum support to the same, that is, min_sup=0.3. The experimental results are shown in Table 1.1.

TABLE.I. COMPARING APRIORI WITH MEC-APRIORI MINING TIME

Transaction itemsets	Apriori Mining time/s	MEC-Apriori Mining time /s
2050	18.8	12.6
4150	25.4	14.8
6300	35.6	22.8
8150	59.2	35.7
11040	72.6	40.5

According to the experiment, the result obtained, convert the result to a line chart to make it more intuitive, Figure 4 shows the time Performance between MEC-Apriori and Apriori.

Horizontal axis: number of transaction item sets. Vertical axis: time/s.

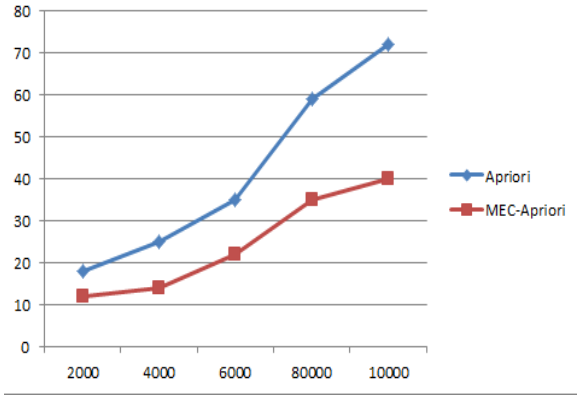

Figure 7. Improved and improved time performance charts

From the figure 4, the MEC-Apriori algorithm and the classical Apriori algorithm are on the premise of the same number of transaction itemsets, it is often better than Apriori algorithm in temporal performance, and with the increasing number of transaction item sets, apriori algorithm running on a computer can significantly improve the time of mining analysis. However, with the MEC-Apriori algorithm, as the number of transaction item sets increases, the time performance is getting better and better. Because with the increase in the number of transaction items, the nodes of the distributed cluster will gradually increase. In summary, the improved MEC-Apriori algorithm is superior to the classic Apriori algorithm in temporal performance.

Experiment 2: Performance Comparison between Apriori Algorithm and MEC-Apriori Algorithm under Different Supporting Degrees.

First ,this paper test the data set RETAIL, select the minimum support threshold range [0.02, 0.20]. And within this range, evenly increase the step: 0.02, so there will be a threshold of 10. Then, this paper use the data set retail to run the Apriori algorithm and the MEC-Apriori algorithm respectively, and record the running time (Note that the running time is second). Figure 5 shows the experimental data obtained by executing the above three algorithms. Horizontal axis: support; vertical axis: time/s.

Experiments show that the MEC-Apriori algorithm runs much less time than the Apriori algorithm under different support levels. The higher the support, the Apriori algorithm will run a little longer than the MEC-Apriori algorithm. In summary, the temporal performance of the MEC-Apriori algorithm under different support levels is always superior to the traditional Apriori algorithm.

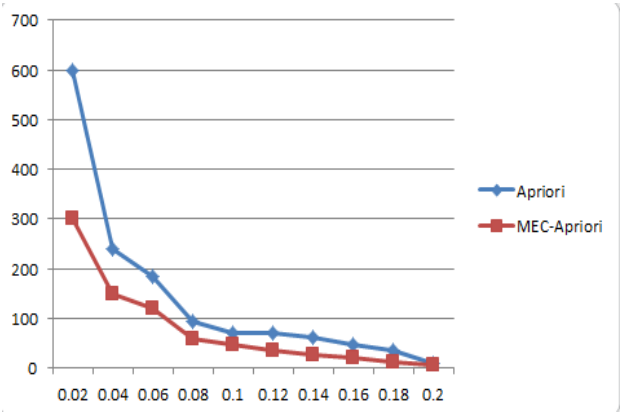

Figure 8. Performance comparison under different support levels

V. CONCLUSION

Aiming at the traditional Apriori algorithm, when mining frequent itemsets, you need to continuously scan transaction data sets , So that the system I / O overhead and other shortcomings. In this paper, we improved Apriori algorithm in three aspects: compression in the transaction, reducing the number of scanning areas, and simplifying the candidate set generation. At the same time, the improved algorithm is parallelized in the Hadoop framework. The simulation results show that compared with the traditional Apriori algorithm, the MEC-Apriori algorithm has good performance and security in temporal performance, mining frequent candidate itemsets and different support levels. However, it needs to be continuously improved in the future work.

REFERENCES

[1] K.WANG, Y.HE, J.HAN. Mining Frequent Itemsets Using Support Constraints. Proc2000 Int. Conf. Very Large Data Bases[J]. Cairo, Egypt, 2000.9: 43-52.

[2] Yan Xiaofei. Research on Association Rule Mining Algorithm[D]. Chongqing: Chongqing University, 2009:15-21.

[3] AGRAWAL R.SRIKANT R.Fast algorithm for mining a ssociation rules[C]//Proceedings of 20th Int. Conf. Very Large Data Bases(VLDB). Morgan Kaufman Press,1994:487-499.

[4] REN W J , YU B W. Improved Apriori Algorithm Based on Matrix Reducation[J]. Computer and Modern,2015,10. 2-3. (in Chinese)

[5] GUNARATHNE T , WU TL, QIU J ,et al .MapReduce in the Clouds for Science[C]//2010 IEEE Second International Conference on Cloud Computing Technology and Science (Cloudcom). IEEE,2010;565-572

[6] DEAN J,GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1):107-113.

[7] HE B S, TAO M, YUAN X M. Alternating direction me-thod with Gaussian back substitution for Separable convex programming [J]. SIAM J. Optimization, 2012, 22(2): 313-340.

[8] HE B S,LIAO L Z,YUAN X M. Alternating projection based prediction-correction methods for structured variational inequalities[J]. Computational Mathematics, 2006, 24(6):693-710.

[9] CHEN Z M, WAN L, YANG Q Z. An Inexact Direction Methodfor Structured Variational Inequalities[J]. Journal of Optimization Theory & Applications, 2014, 163(2): 439-459.

[10] Lu Jiaheng. Hadoop Combat [M]. Beijing: Mechanical Industry Press, 2011: 17-128.