Incremental Association Rule Mining Algorithm Based on Hadoop

Zhu Ying
School of Computer Science and Engineering
Xi'an Technological University
Xi' an, 710021, China
e-mail: 642454565@qq.com

Wang Jianguo
School of Computer Science and Engineering
Xi'an Technological University
Xi' an, 710021, China
e-mail: wjg_xit@126.com

***Abstract*—Aiming at the problems of low efficiency, low cost of time and space, this paper proposes an incremental association rule mining algorithm based on Hadoop load balancing. In combination with the tree structure, when the data in the database is constantly updated, it does not need to scan the original database to generate frequent item sets, and use the load balancing in the data distribution so that the master node distributes the data to the child nodes evenly. In the experiment of control variable method, the variables of minimum support and sample increment are processed respectively. The experimental results show that when the minimum support is unchanged and the transaction data set is increased, the incremental association rule mining algorithm based on Hadoop load balancing takes less than 14.3% of the Apriori algorithm. The number of association rules mined by the algorithm is more than that of the Apriori algorithm. And the memory usage of the Hadoop-based incremental association rule mining algorithm is much smaller than the Apriori algorithm; when the total amount of transaction data is constant and the minimum support is changed, the memory usage of the Hadoop-based incremental association rule mining algorithm is smaller than the Apriori algorithm. The Hadoop-based incremental association rule mining algorithm has some improvements in memory usage and efficiency.**

Keywords—Hadoop; Tree Structure; Load Balancing; Frequent item Sets; Association Rules

I. INTRODUCTION

In today's era of big data, data is an important asset of information society, while data processing and management are also facing great challenges, such as data large amount of information, difficult to handle, difficult to obtain value. Data Mining [1] is a deep data intelligence analysis method. The method involves knowledge of many related fields such as machine learning, deep learning, artificial intelligence, and information retrieval. It is mainly used to extract potential information data from a massive data set that can contribute to the value of business decisions.Association rule mining technology [2] is a very important part of data mining technology. Under the big data environment, it can find many valuable information from the huge and irregular data by discovering the relationship among the items in the data set. Research on association rule algorithm in big datatechnology environment is a very important and challenging research topic. Considering that the data in the database is constantly updated, the incremental association rule updating techniques have been proposed to effectively maintain the association rules of updated data.

Literatures[3-4] proposed incremental association rule update technology, it is a highly efficient maintenance of data updating association rules, the literature describes in detail the characteristics of the technology. In the literature [5], the frequent item sets mining algorithm of association rules is proposed, which is an important component of association rule analysis, data statistics, causality and other mining tasks. The idea of parallel implementation of algorithm is proposed in the literature [6]. Data is distributed by the master node to each node to reduce the load pressure. In the literature [7], the FUP algorithm needs to scan the original database several times to solve the frequent item sets. Of the data environment, the mining efficiency is low.

DOI: 10.21307/ijanmc-2019-015

In the literature [8-9], the paper introduces the characteristics and structure of the B+ tree, also mentioned the one-dimensional data index structure in the database.

When dealing with large amounts of data, processing efficiency can be compromised due to high server load.Therefore, this paper presents proposes an incremental association rule mining algorithm based on Hadoop load balancing. The implementation of this algorithm uses the Map/ Reduce programming model to parallelize the mining tasks, and uses load balancing to mine frequent item sets in the clusters to get the association rules.

II. FUP ALGORITHM (FAST UPDATE PARALLEL)

TABLE I. SYMBOLIC DESCRIPTION

symbol	means
DB	raw data set
db	new data set
DBUdb	all data sets
L_k	frequent item sets(k-order)
C_k	candidate set(k-order)

When calculating the frequent item sets, the FUP algorithm [10] makes use of the known support of the old frequent item sets, and then through the frequent analysis of the candidate item sets in DB and db, many candidates can be filtered out in the process of determining frequent item sets. Therefore, it is more efficient than re-running the Apriori algorithm, especially if the old and new frequent sets are not much different. However, like the Apriori algorithm, the FUP algorithm must spend a lot of time processing a large set of candidate items and it is necessary to repeatedly scan the database DB and db to pattern match the candidate sets. Since the FUP algorithm uses L_{k-1} to generate C_k, the generated candidate set of db is large, and many of them are infrequent, and some even are impossible to appear in db, which affects the efficiency of the algorithm. In addition, the FUP algorithm needs to perform K+1 layer scanning on the entire database DBUdb. In general, the original database DB is very large, so the k+1 layer scanning for the DB is not efficient.

The shortcomings of the current incremental update association mining algorithm:

1) When the database is updated, the generation of frequent item sets will repeatedly scan the processed data sets, so that the mining efficiency is not high;
2) generating a large number of candidate sets;
3) With the addition of the database, the minimum support threshold and the minimum confidence threshold do not change, which will affect the mining of strong association rules;

III. FUP ALGORITHM IMPROVEMENT

Due to the inefficiency of FUP algorithm in dealing with big data mining, this paper proposes an improved algorithm of FUP, Incremental Association Rule Mining Algorithm Based on Hadoop Load Balancing. Combined with the Hadoop distributed platform and the use of a tree structure to store information, it is only necessary to scan the DB and dbone time, which can effectively improve the scanning efficiency. When the database is updated, the Newton interpolation formula is used to estimate the support threshold to efficiently process the association rules.

A. Vertical data representation [11]

A traditional database can be represented by a two-dimensional matrix. Each column of a two-dimensional matrix can be treated as a project, and each row can be treated as a transaction of the database. In the traditional Apriori algorithm, horizontal data is used to represent a transaction. The typical method of implementing two-dimensional matrix is horizontal data. Most data mining algorithms use database access to calculate their support. In relational databases, the size of the data in the transaction database is relatively large, and there is a problem with the size of the computer memory at present, and there will be bottlenecks. This will lead to a contradiction: due to the continuous increase of data in the database. The size of the data will be larger and larger, but the memory and hard disk of the computer are limited. Therefore, when the data in the database reaches a certain scale, the computer memory and the hard disk will not be stored enough. Therefore, a new type of data representation is generated—vertical data representation. Vertical data means that each record in the database consists of a list of all the transaction records that it has appeared, as shown in Tab.2:

TABLE II. HORIZONTAL DATABASE

TID	Item
1	citrus fruit, yogurt, butter, ham
2	tropical fruit,yogurt, coffee
3	whole milk,citrus fruit,newspapers
4	tropical fruit, yogurt, cream cheese , whole milk
5	coffee, butter, yogurt, sauces, ham
6	butter, ham, cream cheese, sauces, newspapers

TABLE III. VERTICAL DATABASE

Item	TID
butter	1,5,6
citrus fruit	1,3
coffee	2,5
cream cheese	4,6
ham	1,5,6
newspapers	3,6
sauces	5,6
tropical fruit	2,4
whole milk	3,4
yogurt	1,2,4,5

B. *Tree structure stores data information*

The B+ tree [12] is a tree data structure, which is an n-fork sort tree. It is a variant tree of B-trees required by the file system. It features:

1) There are n keywords in the node of n sub-trees. Each keyword does not save data, only used for indexing, and all data is stored in leaf nodes.
2) All leaf nodes contain information about all keywords, and pointers to records containing these keywords, and the leaf nodes themselves are linked in order from the smallest to the largest.
3) All non-terminal nodes can be thought of as index parts, which contain only the largest (or smallest) keyword in their sub-tree (root node).

As shown in Fig. 1, it is a B+ tree of m=3(consisting of information in Tab.2).

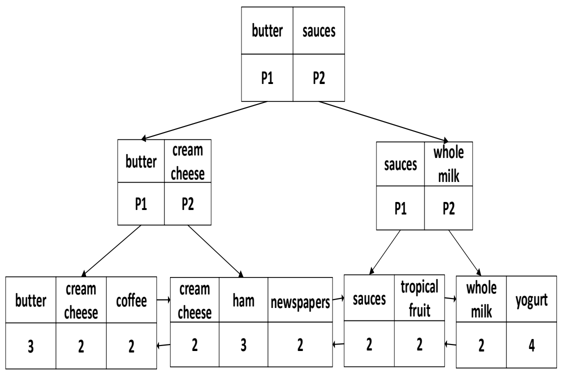

Figure 1. Order three of the B+ tree

C. *Hadoop distributed platform*

In this era of increasing data size, the traditional frequent item set mining algorithm has been difficult to support the mining of its massive transaction database. Even if it can be mined, it will lead to a particularly long mining time, or it may be due to the huge amount of data. Causes the program to fail. The solution to the problem of the traditional frequent item set mining algorithm is to add the Hadoop distributed platform to the frequent item set mining algorithm, which not only makes the traditional frequent item set mining algorithm run more efficiently, but also reduces the storage pressure of the database.

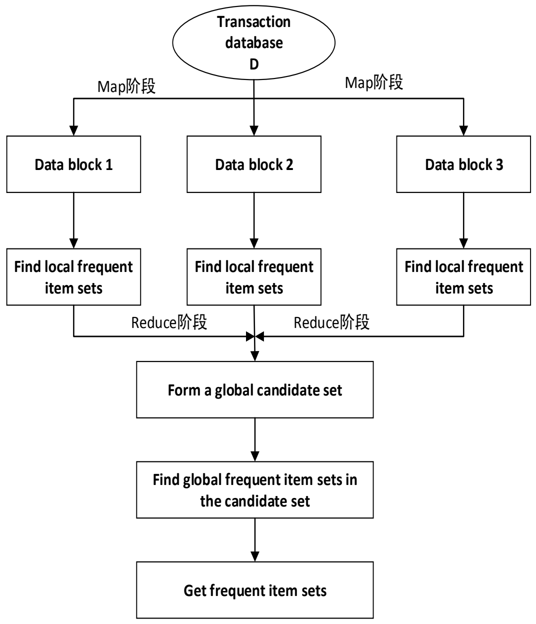
Transaction database D
Map阶段
Map阶段
Data block 1
Data block 2
Data block 3
Find local frequent item sets
Find local frequent item sets
Find local frequent item sets
Reduce阶段
Reduce阶段
Form a global candidate set
Find global frequent item sets in the candidate set
Get frequent item sets

Figure 2. Frequent item set generation step diagram

D. HBPT-FUP algorithm

Based on the original algorithm, there are some disadvantages of generating a large number of candidate sets and repeated retrieval processing of the original database. Combined with the characteristics of B+ tree, the data information is stored. Based on FUP algorithm, the incremental association rule mining algorithm under Hadoop platform is proposed: HBPT- FUP algorithm(Incremental Association Rule Mining Algorithm Based on Hadoop Load Balancing，Improved Algorithm of FUP Algorithm).

1) Basic ideas

① Statistics in the database transaction items, get transaction items set. ② Setup process load estimation, through the Map process using load balancing grouping way to read the transaction items, distributed to different Reduce nodes. The Reduce process is based on how the B+ tree is created and stores transaction item information. The bottommost leaf node of B+ tree contains all the item sets. The frequent item sets are obtained by comparing the number of items with the minimum support, and other infrequent item sets remain in their leaf nodes to ensure that future data updating become frequent nodes. The local frequent item sets are obtained by the B+ tree created by the Reduce nodes, and the local frequent item sets are merged into the global frequent item sets to get the association rules.

Compared with the previous incremental updating correlation mining algorithm, the algorithm has the characteristics of load balancing. When dealing with large amounts of data, due to a single server load is too high affect the processing speed, so need to use server clusters to deal with. The load factor is added to the estimation of the load, which can calculate the load of each frequent item more accurately.

The algorithm takes advantage of the B+ balanced tree properties. The leaf nodes at the bottom of the tree contain all the keywords of the whole tree, and they are linked together like a doubly linked list.

2) Estimated load

The load in the parallelization process is equal to the sum of the load on each node [11]. supposed that the load corresponding to the data item is L_{i} , the position in the frequent item set is P_{i}, the load influence factor is a_i (the frequency of the data item):

$$\mathrm{L_i} = \log(P_i + a_i) \tag{1}$$

3) Equalization of data

Frequent items in frequent item sets are divided into Q groups according to the balanced grouping strategy, which makes the data balance in the distribution process so as to improve the parallelization of the mining process.

If Q is less than the length of frequent itemsets, the Q items of frequent itemsets are assigned to each group in turn. The group's load is initialized by the sum of the frequent item loads in each group. And then repeat the following two steps until all the frequent items are assigned: ① To assign unallocated frequent items to the group with the lowest load; ② The load in each group is added up.

If Q is greater than the length of frequent itemsets, the first x of the frequent itemsets is assigned to the x group, which initializes the group load according to the load of the frequent items it contains, and then repeats the above two steps until all Frequent items are assigned.

4) Threshold setting

The minimum support threshold will change with the user's needs and the database update. When the threshold is set too low, the more rules are mined, but the rule is less useful. Conversely, when the threshold is set too high, there are very few rules for mining, which will lose some useful rules. Therefore, it is very important to set the appropriate threshold when dealing with incremental databases.

Definition 1: The support degree $\mathrm{Support(X)}$ of item set X is one of the most important metrics in association rule mining. It represents the probability that the item set $\{\mathrm{X}, \mathrm{Y}\}$ will appear in the total item set. The formula is:

$$\mathrm{Support(X \to Y)} = \frac{\mathrm{P(X,Y)}}{\mathrm{P(I)}} = \frac{\mathrm{P(X \cup Y)}}{\mathrm{P(I)}} = \frac{\mathrm{num(X \cup Y)}}{\mathrm{num(I)}} \quad (2)$$

Where I is the total transaction sets.num() represents the number of occurrences of a particular item set in a transaction set.

Definition 2: Confidence indicates the probability that Y is derived by the association rule $\mathrm{X \to Y}$ in the event that the prerequisite X occurs. That is, in the item set containing X, there is a possibility of Y. The formula is:

$$\mathrm{Confidence(X \to Y)} = \mathrm{P(X|Y)} = \frac{\mathrm{P(X,Y)}}{\mathrm{P(X)}} = \frac{\mathrm{P(X \cup Y)}}{\mathrm{P(X)}} \quad (3)$$

The Newton interpolation formula [13] that can be used to derive the support threshold from the above formula is:

$$\mathrm{Sup} = \mathrm{f}\left(\frac{\mathrm{C*n}}{\mathrm{M}}\right) \quad (4)$$

Where C is the confidence level, n is the number of rules, and M is the total number of attributes in the transaction set.

When the association rule is first mined, the setting of the minimum support threshold is a trial and error process. When the transaction database is updated, it is possible that the previously set threshold is no longer applicable, so an adjustment threshold is required.

In order to improve the accuracy of the estimation, the order of the Newton interpolation polynomial can be improved as the number of executions of the algorithm increases. Because the algorithm generates a parameter pair $(\mathrm{x_k}, \mathrm{Sup_k})$ each time it runs. If this node is farther from the nearest two nearest interpolation nodes in the interpolation formula, you can add this node as a new interpolation node and adjust the interpolation formula. The accuracy of the adjustment of the adjusted interpolation formula at point $\mathrm{x_i}$ is greatly improved.

Newton interpolation formula automatically determines the specific implementation process of support threshold, as follows:

Statistics are performed on some data in the experimental transaction data set (shopping data record), and the support degree and confidence corresponding to the rule $\mathrm{RD \to CD}$ are calculated, as shown in Tab.3 (support, confidence in parentheses):

TABLE IV. RULE SUPPORT AND CONFIDENCE CALCULATION RESULTS

RD / CD	ham	bread	coffee
butter	(0.4,0.55)	(0.13,0.28)	(0.2,0.3)
yogurt	(0.36,0.54)	(0.63,0.58)	(0.26,0.39)
cheese	(0.43,0.51)	(0.3,0.53)	(0.31,0.52)

a) For the data in Tab.2, set the reliability threshold to 0.5, the support threshold to 0.3, and run frequent pattern mining. The association rules available according to Tab.3 are:

butter->ham,yogurt->ham,yogurt->bread,

cheese->ham,cheese->bread,

cheese->coffee.

b) Set the reliability threshold to 0.5 and the support threshold to 0.4. Run frequent pattern mining. The association rules available according to Tab.3 are:

butter->ham, yogurt->bread, cheese->ham.

c) Set the reliability threshold to 0.5 and the support threshold to 0.6. Run frequent pattern mining. The association rules available according to Tab.3 are:

yogurt->bread.

The Newton interpolation formula for the known support threshold is $\mathrm{Sup} = \mathrm{f}(\frac{\mathrm{C*n}}{\mathrm{M}})$. Then can get the following three expressions:

$$f\left(\frac{0.5*7}{8}\right) = f(0.4375) = 0.3$$

$$f\left(\frac{0.5*4}{8}\right) = f(0.25) = 0.4$$

$$f\left(\frac{0.5*1}{8}\right) = f(0.0625) = 0.6$$

So you can get three interpolation nodes (x_k, Sup_k)as:(0.4375,0.3),(0.25,0.4),(0.0625, 0.6).

According to these three interpolation nodes, the difference quotient is shown in Tab.4:

TABLE V. DIFFERENCE QUOTIENT TABLE

Number of rules (n_k)	x_k	$f(x_k)$	**First order difference quotient**	**Second order difference quotient**
1	0.0625	0.6		
4	0.25	0.4	-1.07	
7	0.4375	0.3	-0.53	1.44

From Tab.4, a second-order Newton interpolation polynomial can be written:

$$Sup = f(x) = N_2(x) = 0.6 + (-1.07)(x - 0.0625) + 1.44(x - 0.0625)(x - 0.25) = 1.44x^2 - 1.52x + 0.69 \quad (5)$$

At this time, the support threshold can be estimated according to the Newton interpolation formula $N_2(x)$ and the number of association rules expected by the user.

a) Determining the association rules that the user desires to mine (when the database update is about to be performed next time, the number of expected association rules is set to the average value of the total number of association rules of the previous mining);

b) Calculate the value of X according to the formula $x = \frac{C*n}{M}$;

c) Substituting the value of X into $N_2(x)$ to calculate the estimated value of support Sup.

Assuming that the user expects to mine three association rules, the calculated x value is:

$$x = \frac{C*n}{M} = \frac{0.5*3}{8} = 0.1875 \quad (6)$$

Then the X value is substituted into $N_2(x)$ to get the support: $Sup = N_2(x) = 0.455$. Sup can be chosen to be a value close to 0.455. For example, take sup=0.45 as the support threshold to perform the mining algorithm.

E. Algorithm Implementation

1) Brief description of the algorithm steps

Using the tree structure to store information, a Hadoop-based incremental association rule mining algorithm HBPT-FUP is proposed. The algorithm is described as follows:

Step 1: traversing the transaction database, processing according to the vertical data representation to get the item set. In the Setup phase: initialize the number of packets Q, distribute the transaction items of the primary node to the Q child nodes. A vertical data set is created in each node, and the vertical data set is traversed to obtain an item set inserted into the B+ tree.

Step 2: In each child node, all the frequent item sets of the group are obtained according to the generated B+ tree.

Step 3: When the transaction database updates the record, follow the first step to balance all incremental load to each node. The main focus is on the update algorithm of the B+ tree.

Add a keyword to the leaf node at the bottom of the B+ tree. If the number of keywords contained in the node does not exceed M, the insertion is successful. Otherwise, the node needs to be split. The number of keywords included in the two new nodes after splitting should not be less than (M/2 + 1). If the parent node of the node has enough space (M/2 ~ M-1), the parent node should contain the smallest keyword in the right node. Otherwise, the parent node is split and a new keyword is added to the parent of the parent node, followed by a higher order until the new root node is generated (1 ~ M-1).

Step 4: Association rules generated after database update.

2) Algorithm flowchart

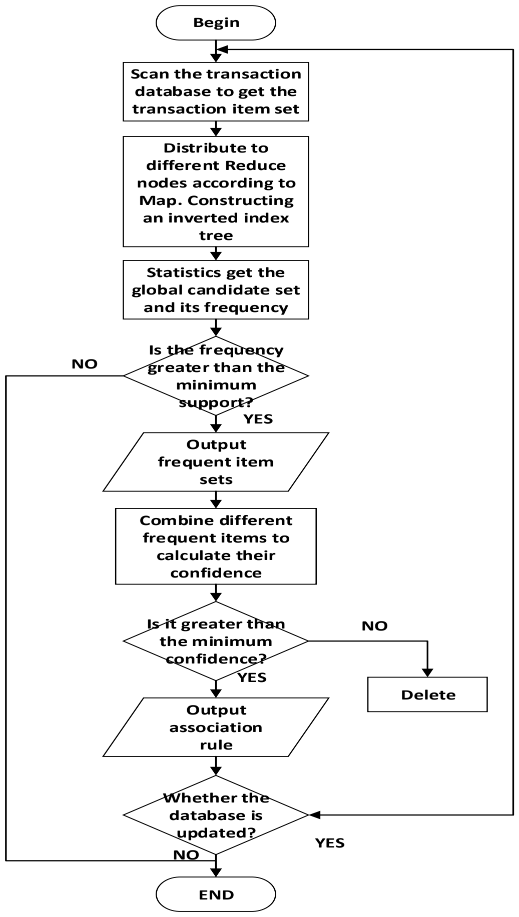
Begin
Scan the transaction database to get the transaction item set
Distribute to different Reduce nodes according to Map. Constructing an inverted index tree
Statistics get the global candidate set and its frequency
Is the frequency greater than the minimum support?
NO
YES
Output frequent item sets
Combine different frequent items to calculate their confidence
Is it greater than the minimum confidence?
NO
YES
Delete
Output association rule
Whether the database is updated?
YES
NO
END

Figure 3. HBPT-FUP algorithm step diagram

IV. EXPERIMENTRESULTS AND ANALYSIS

In this experiment, the file retail (association rule study classic data set) downloaded from the UCI database was used as the experimental transaction data set, and the HBPT-FUP algorithm was compared with the Apriori algorithm. Use java language to simulate on Win7, dual-core 2.3GHZ CPU, 4GB memory PC.

1) Analysis of the time complexity of the algorithm

When the data is updated, ①only scans and updates part of the data in the database;②scans the tree structure once and inserts the new item set. The analysis shows that when the minimum support is constant, the execution time of the algorithm is related to the amount of data updated each time. A small number of experimental samples are extracted from the dataset, and the minimum support degree of the Apriori algorithm is controlled (min_sup=0.45), and the amount of updated data is sequentially increased. The experimental time of the Apriori algorithm and the HBPT-FUP algorithm on a PC is recorded Figure 4 shows:

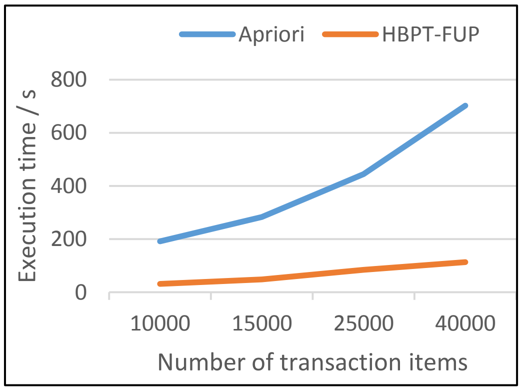

Figure 4. Apriori and HBPT-FUP algorithms time comparison

As shown in Figure 4, when the minimum support is constant, the execution time of the Apriori algorithm grows faster than the HBPT-FUP algorithm when the data set increases. When the data set is the same, the Apriori algorithm takes longer than the HBPT- FUP.

2) Comparison of the number of association rules mined.

When the database is updated, the minimum support threshold of the Apriori algorithm does not change (0.45), and the use of the Newton interpolation polynomial in the HBPT-FUP algorithm predicts the next minimum support threshold, which in turn makes the mined rules more efficient. In Experiment 1, the number of association rules mined each time is counted separately. The number comparison is shown in Figure 5:

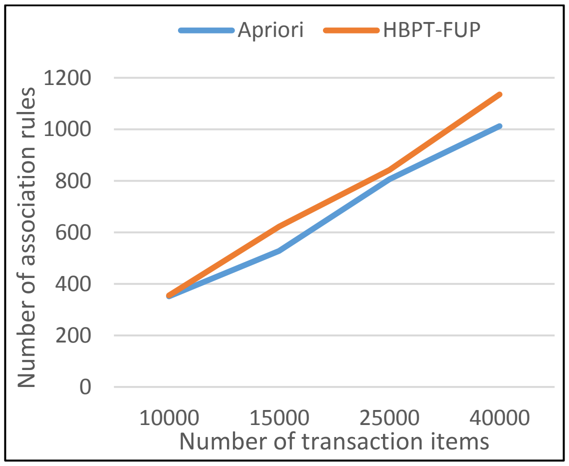

Figure 5. Comparison of the number of association rules

As shown in Figure 5, when the amount of update data increases, the minimum support remains unchanged, and the number of association rules mined by the Apriori algorithm is less than that extracted by the HBPT-FUP algorithm. After adjusting the minimum support, the HBPT-FUP algorithm mines some rules that are easily missed, which increases the validity of the rules.

3) *Analyze the spatial complexity of the algorithm*

①In the B+ tree, only the item set and its frequency are stored, so the size of the occupied memory space is related to the total number of items; ②storing the frequent item sets determined by the minimum support, Therefore, the memory space is related to the minimum support.

a) Study the impact of the number of transaction item sets on memory usage. A small number of experimental samples are extracted from the data set, and the minimum support degree of the Apriori algorithm and the HBPT-FUP algorithm is controlled (min_sup=0.45), and the amount of updated data is sequentially increased. The Apriori algorithm and the memory usage of the HBPT-FUP algorithm on one of the PCs are recorded separately, as shown in Figure 6. (Set the minimum support threshold of the HBPT-FUP algorithm to remain the same)

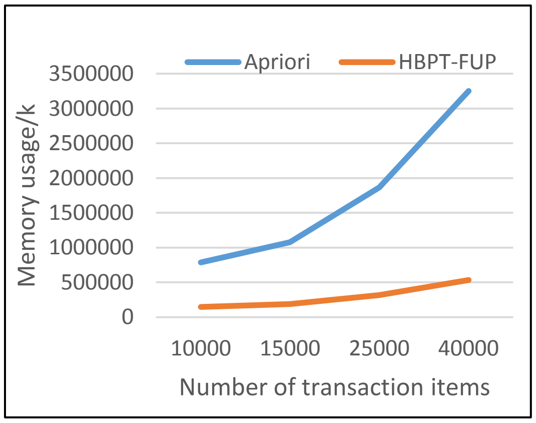

Figure 6. Memory usage comparison

As shown in Figure 6, when the minimum support is unchanged and the number of transaction items increases, the number of items generated increases, and the memory space occupied is larger. The Apriori algorithm updates the candidate set with the change of the number of transaction items. The memory space stores the candidate set and the frequent item set. The HBPT-FUP algorithm is based on distributed, and only stores data and its frequency in the tree structure of each node. Therefore, the Apriori algorithm takes up more memory space than the HBPT-FUP algorithm.

b) Study the impact of minimum support on memory usage. When the minimum support of the Apriori algorithm changes from 0.2 to 0.6, the data samples (10000) and increments (10000) remain unchanged, and the memory usage when running the Apriori algorithm and the HBPT-FUP algorithm on a PC is recorded separately. Figure 7 shows. (The minimum support threshold for the HBPT-FUP algorithm is automatically estimated based on the Newton interpolation polynomial when the database is updated)

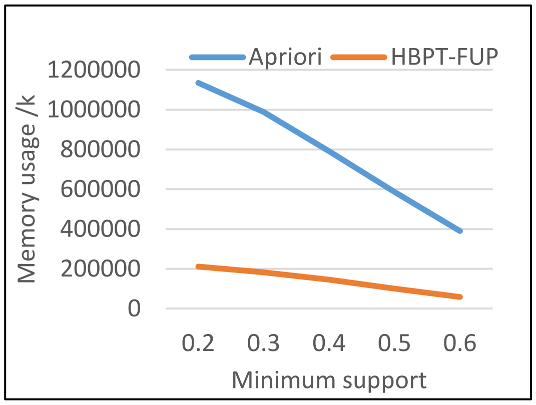

Figure 7. Memory usage comparison

As shown in Figure 7, the smaller the minimum support, the more the number of items generated, the larger the memory space occupied. In the case of the change of support degree, the Apriori algorithm updates the candidate set with the change of the support degree, and saves the candidate set and the frequent item set in the memory space. The HBPT-FUP algorithm does not generate the candidate set during the support change process, so The Apriori algorithm occupies more memory than the HBPT-FUP algorithm.

V. Conclusion

In this paper, a parallel update algorithm based on load balancing for incremental update association mining algorithm HBPT-FUP is proposed.The algorithm effectively realizes that when the database is updated, the original database is not scanned, and the newly added data is inserted into the tree based on the original B+ tree to obtain a frequent item set. Handle large-scale data with load balancing technology. Distribute data to different PCs in the cluster. The experimental results show that when the number of transaction records is the same as the amount of new data, the time spent by the HBPT-FUP algorithm is less than 14.3% of the Apriori algorithm, which improves the efficiency of mass data processing. When the minimum support changes, the memory space occupied by the HBPT-FUP algorithm on the same PC is much smaller than the Apriori algorithm. The algorithm has some improvements in terms of efficiency and memory usage.

Acknowledgment

First of all, I would like to thank my tutor during the master's degree, Professor Wang Jianguo. During my writing stage, Mr. Wang continued to give guidance and help, and taught me the direction of my own research. Mr. Wang has a wealth of professional domain knowledge and excellent data mining direction. Under the kind guidance of Mr. Wang, my ideas have been broadened, the bottlenecks in the research process have been solved, and my ideas have been implemented more steadily. At the same time, Mr. Wang is serious and responsible for his work and his mastery of professional knowledge. It is worth everyone to learn and respect.Secondly, I would like to thank Xi'an Technological University for guiding me to grow up. Finally, I am very grateful to the journal for including my thesis.

References

[1] Mao Guojun, Duan Lijuan, Wang Shi et al. The first edition of data mining principles and algorithms, Beijing Tsinghua University Press. 2005: 156-162.

[2] Rakesh Agrawal, TomaszImieliński,Arun Swami. Mining association rules between sets of items in large databases[J]. ACM SIGMOD Record,1993,22(2).

[3] FENG Yucai, FENG Jianlin. Incremental Updating Algorithm for Association Rules[J] .Journal of Software, 1998,9 (4): 301.

[4] Yuan Changan, Data mining theory and SPSS clementine jewel [M]. Beijing: Publishing House of Electronics Industry,2009.

[5] XIE Peng-jun. Study on Parallelization of Frequent Item sets Mining Algorithm Based on MapReduce [D]. Nanjing: Nanjing University, 2012.

[6] SHI Liang, QIAN Xue-zhong. Study and Implementation of Parallel FP-Growth AlgorithmBased on Hadoop [J]. Microelectronics and Computer, 2015, 4 (4):150.

[7] Incremental maintenance of discovered association rules and approximate dependencies[J]. Alain Pérez-Alonso, Ignacio J. Blanco Medina, Luisa M. González-González, José M. Serrano Chica. Intelligent Data Analysis. 2017(1):117.

[8] JES_US R, CAMPA~NA, JUAN M,et,al.Indexing fuzzy numerical data with a B+ tree for fast retrieval using necessity-measured flexible conditions[J].World Scientic,2009,17(1):1.

[9] HU Yanbo,ZHONG Jun. Based on clustering B + tree database index optimization algorithm [J]. Computer Applications, 2013,33 (9): 2474.

[10] David W.Cheung, Jiawei Han. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In Proc of the Twelfth International Conference on Data Engineering,1996.USA: IEEE, 1996(3):106-114.

[11] Chen Fengjuan.Vertical Data Format Mining of Probabilistic Data Sets[J].Journal of Anyang Teachers College,2016(02):41-43+69.

[12] Wang Yingqiang, Shi Yongsheng.Application of B+ Tree in Database Indexing[J].Journal of Yangtze University,2008,3(5):233.

[13] Chen Xiaojiang, Huang Yucan, et al. Numerical analysis. Beijing: Science and Technology Press.

[14] Alan Sexton,HayoThielecke. Reasoning about B+ Trees with Operational Semantics and Separation Logic[J]. Electronic Notes in Theoretical Computer Science,2008,218.

[15] HyunyoonYun,DanshimHa,BuhyunHwang,et al. Mining association rules on significant rare data using relative support[J]. The Journal of Systems & Software,2003,67(3).

[16] Jia Wang,HongguangLi,JingwenHuang,et al. Association rules mining based analysis of consequential alarm sequences in chemical processes[J]. Journal of Loss Prevention in the Process Industries,2016,41.