International Journal of Database Theory and Application
Vol.9, No.4 (2016), pp. 195-208
http://dx.doi.org/10.14257/ijdta.2016.9.4.18

Large-Scale Dataset Incremental Association Rules Mining Model and Optimization Algorithm

Guo Yu-Dong[1], Li Sheng-Lin[1], Li Yong-Zhi[1], Wang Zhao-Xia[1] and Zeng Li[2]

[1]*Department of military logistics information and logistics engineering, Logistics Engineering Institute, Chongqing 401311*
[2] *School of Civil Engineering and Architecture, Chongqing University of Science and Technology, Chongqing 401311*
guoyudong116046122@163.com

Abstract

Mining association rules is an important research direction in the field of data mining. Related studies have proposed many used to efficiently find large-scale database association rules algorithm, but the research on maintenance problem of association rules is less. Especially many transaction database is always in constant updates. Increase or decrease occurs when the database or dataset minimum support after the change, how to maintain the association rules have been, it got the attention of many researchers. Based on IFP-Growth increment of association rules mining model and to modify the FP-tree, put forward the suitable for transaction data and support the tree model of change, at the same time under different conditions is given incremental association rules mining algorithm, and reduce the frequency of the original dataset range query and query, and in a case of massive dataset multi-level tree structure decomposition, dynamic allocation rule tree branches, ensure load balancing, improve operation efficiency.

Keywords*: Data mining, Association rules, Support, Parallel computing*

1. Introduction

Many algorithms have been proposed to efficiently mine association rules. But few algorithms aim to solve the problem when datasets and support are changing at the same time. As many of the transactional database is always in constant updates. When the database changed, how to maintain the association rules we have already obtain is the problem which got the attention of many researchers. FUP [1] is the earliest incremental association rules mining algorithm, the algorithm only deals with the increase in new transaction in the database. FUP algorithm is based on the idea of Apriori algorithm, and studied the optimum processing algorithm using the pruning strategy of butyl DHP. Algorithm firstly comes from the increasing number of records in the mining frequent itemsets, and compare them with original frequent itemsets. According to the results of the comparison of FUP algorithm to decide whether to need to scan the original database. FUP2 [2] is a supplement to the FUP algorithm, it can not only deal with the new dataset, but also can remove part of the database. UWEP [3] algorithm is to study the rules of the new database maintenance. The algorithm on the new dataset get to join after no longer frequent itemsets using a dynamic pruning strategy in advance. UWEP algorithm analyze on the generation of candidate set and the support is better than the previous algorithms. SWF [4] algorithm is mainly studied two candidate set generation problems after updating the database. As a result, the efficiency of the algorithm is better than Apriori and FUP2, which have greatly improved.

ISSN: 2005-4270 IJDTA

The incremental mining algorithm is based on Apriori, and FIUA [5] is based on frequent pattern tree incremental updating algorithm of association rules. FIUA exploited original frequent itemsets and original FP-trees to efficiently mining new frequent itemsets after data changed or support threshold changed. Compared with FUP algorithm, the performance of FIUA has improved greatly. Literature [6] is to study the parallel incremental updating algorithm and methods to improve the efficiency of the algorithm.

2. VSIFP-Growth

2.1 Problem Model on the VSIFP-Tree

Suppose that our original dataset of transaction database is DS, the original dataset transaction volume $N = |DS|$, ds is an increment dataset of transaction database, the increment dataset transaction volume $n = |ds|$, $DS + ds$ is the updated dataset, the updated transaction volume is $N + n = |DS| + |ds|$.

Given as support counts of itemsets x in dataset DS, ds and $DS + ds$ are $x\_sup(DS)$, $x\_sup(ds)$ and $x\_sup(DS + ds)$ respectively. The following equation was established $x\_sup(DS) + x\_sup(ds) = x\_sup(DS + ds)$. Let the L_{DS} be a frequent itemset of DS, L_{ds} is a frequent itemset of ds, and L_{DS+ds} is a frequent itemset of $DS + ds$. $Sup(DS + ds)$ is the support of x in $DS + ds$ database. By the definition of support, $Sup(DS + ds) = x\_sup(DS + ds) / (|DS| + |ds|)$.

When minimum support changes, the construction of VSIFP-tree can be divided into the following several ways:

1) If minimum support of the new ds is lower than minimum support of the original DS ($min\_sup' < min\_sup$):

(1)If $x \notin L_{DS}$ and $x \notin L_{ds}$, maybe $x \in L_{DS+ds}$.

Firstly, we must determine the size of the $x\_sup(DS)$ and $min\_sup' \times |DS|$. Because of $x \notin L_{DS}$, so $x\_sup(DS) < min\_sup' \times |ds|$.

①If $x\_sup(DS) < min\_sup' \times |DS|$, then the support of the whole dataset is:

$$Sup(DS + ds) = x\_sup(DS + ds)/|DS| + |ds| =$$
$$x\_sup(DS) + x\_sup(ds) /|DS| + |ds| <$$
$$min\_sup' \times |DS| + min\_sup' \times |ds|/|DS| + |ds| = min\_sup'$$

②If $x\_sup(DS) \geq min\_sup' \times |DS|$ and $Sup(DS + ds) \geq min\_sup'$, then x is a frequent itemset of $DS + ds$.

(2)If $x \in L_{DS}$ and $x \in L_{ds}$, then $x \in L_{DS+ds}$.

Because of $x \in L_{DS}$ and $x \in L_{ds}$, so $x\_sup(DS) \geq min\_sup \times |DS|$ and $x\_sup(ds) \geq min\_sup' \times |ds|$, then the support of the whole dataset is:

$$Sup(DS + ds) = x\_sup(DS + ds)/|DS| + |ds| =$$
$$x\_sup(DS) + x\_sup(ds) /|DS| + |ds| >$$
$$min\_sup' \times |DS| + min\_sup' \times |ds|/|DS| + |ds| = min\_sup'$$

(3)If $x \in L_{DS}$ and $x \notin L_{ds}$, maybe $x \in L_{DS+ds}$.

On this occasion, x is a frequent itemset of $DS + ds$, the smaller minimum support is still a frequent itemsets of DS. Because of $x \notin L_{ds}$, we need to determine the size of $Sup(DS + ds)$ and min_sup'. If $Sup(DS + ds) \geq min\_sup'$, then x is a frequent itemset of $DS + ds$.

(4)If $x \notin L_{DS}$ and $x \in L_{ds}$, maybe $x \in L_{DS+ds}$;

Firstly, we must determine the size of the $x\_sup(DS)$ and $min\_sup' \times |DS|$.

①If $x\_sup(DS) \geq min\_sup' \times |DS|$, then x is a frequent itemset of DS. Because of $x \in L_{ds}$, so $x_{sup(ds)} \geq min\_sup' \times |ds|$.

$$Sup(DS + ds) = x\_sup(DS + ds)/|DS| + |ds| =$$
$$x\_sup(DS) + x\_sup(ds)/|DS| + |ds| >$$
$$min\_sup' \times |DS| + min\_sup' \times |ds|/|DS| + |ds| = min\_sup'$$

.

②If $x\_sup(DS) < min\_sup' \times |DS|$, then x is still an infrequent itemset of DS. Only when $Sup(DS + ds) \geq min\_sup'$, then x is a frequent itemset of $DS + ds$.

2)If minimum support of the increment dataset ds is greater than minimum support of the original dataset DS ($min\_sup' > min\_sup$), we can prove the following conclusion analogously:

(1)If $x \notin L_{DS}$ and $x \notin L_{ds}$, then $x \notin L_{DS+ds}$.

(2)If $x \in L_{DS}$ and $x \in L_{ds}$, maybe $x \in L_{DS+ds}$.

(3)If $x \in L_{DS}$ and $x \notin L_{ds}$, maybe $x \in L_{DS+ds}$.

(4)If $x \notin L_{DS}$ and $x \in L_{ds}$, maybe $x \in L_{DS+ds}$.

Through the above proof of algorithm of association rules in the incremental data sets and different support conditions, we can infer the following features of VSIFP-tree.

Feature 1 of VSIFP-tree, regardless of the minimum support increase or decrease, if the intersection of the frequent 1-itemsets of original dataset and frequent 1-itemsets of incremental dataset is not null, then proper subset of this intersection is included in the frequent 1-itemsets of new dataset.

Feature 2 of VSIFP-tree, under the condition of minimum support is unchanged, if the union set of the frequent 1-itemsets of original dataset and frequent 1-itemsets of incremental dataset is not null, then proper subset of complement set of this set in all items is not included in the frequent 1-itemsets of new dataset.

2.2 Algorithm Description of VSIFP-Growth

Suppose that our original dataset of transaction database is DS, ds is an increment dataset of transaction database. Given as support counts of itemsets x in dataset DS, ds and $DS + ds$ are $x\_sup(DS)$, $x\_sup(ds)$ and $x\_sup(DS + ds)$ respectively.$X_{countds}$ represent the number of occurrences of X in ds and $X_{countDS}$ represent the number of occurrences of X in DS.Suppose that we have got the frequent itemset of original dataset L_{DS} [7].

In Section 2.1 we analysis different situations when the dataset increase or decrease, support increase or decrease of data processing. Therefore, we can control in the process in different cases of search datasets to generate frequent itemsets, to reduce unnecessary redundant search. This algorithm VSIFP-Growth can build tree association rules in the shortest possible time based on FP-tree. The following program shows how our algorithm generate the VSIFP-tree to insert branches into the FP-tree.

Input: k is dimension of dataset, T is the transaction volume of dataset.
Output: The support counts $X_{countds}$.
Method:
/* Calculate the support counts X in D*/
VSIFP-Growth _GenCounts(k, T){
for(i=1 to k){
for all transaction $T \in ds$ do begin //scan dataset(ds)
for all 1-itemset $X \in T$ do begin
if $X \in C_{dsk}$ then $C_{dsk} = C_{dsk} \cup \{X\}$;
$X_{countds}++$;
}
}

Input: ds is an increment dataset of transaction database, min_sup is the original minimum support, min_sup' is the incremental minimum support.
Output: The frequent itemset and rules of $DS + ds$.
Method:
/* Calculate the frequent itemset $L_{(DS+ds)}$ and association rules $Rules$ */
VSIFP-Growth _Gen(k, T, $X_{countds}$, $X_{countDS}$, min_sup){
for(i=1 to k){
for all transaction $T \in ds$ do begin
if $X \in C_{dsk}$ then $X_{countds}++$;
if$\left((X_{countds} + X_{countDS}) \geq min_sup \times (|DS| + |ds|)\right)$
then $L_{(ds+DS)k} = L_{(ds+DS)k} \cup \{X\}$;
}
suppose that $Q = L_{(DS)k} - C_{dsk}$;
for all k-itemset $X \in Q$ do begin
for all $(k-1)-$itemset $Y \in \left(L_{(DS)k-1} - L_{(DS+ds)k-1}\right)$ do begin
if Y in X then $Q = Q - \{X\}$; break;
if(Q is not null){
L = FP-Growth(Q, ds, min_sup);//invoke FP-growth algorithm
}
if($min\_sup - min\_sup' \neq 0$ || Q is not null){
$Rules$ = GenerateRules(L, min_sup);
else{
$Rules$ = select Rules from Rules where Rule.support >= min_sup;
}
Return L, $Rules$;

}

2.3 An Example of VSIFP-Growth

In this paper, an effective set-based algorithm called VSIFP-growth (variable support incremental frequent pattern growth) is built to reduce the complexity of incremental tree construction. VSIFP-growth algorithm refers to the dataset and support change at the same time be able to generate a new association rule tree. The new association rule tree has different structure to the original FP-tree, but they both have the same mining results. VSIFP-growth can save operation time and does not affect the mining results compared with the original FP-tree.

In order to explain the VSIFP-growth algorithm implementation of the process in case of the minimum support and incremental dataset changes at the same time, we use a concrete example to discuss the FP-tree based VSIFP-tree construction algorithm of mining association rules [8].

Table 1 shows an original dataset of transaction database and sets the $\mathrm{min\_sup}$ as 0.6(3/5). Figure1 illustrates the original FP-tree for Table 1 and Figure2 is a new VSIFP-tree constructed with using original dataset and $\mathrm{min\_sup}'$ as 0.4(2/5).

(1) In the Figure1, subfigure a) shows the original FP-tree with the $\mathrm{min\_sup} = 0.6$ and frequent 1-itemsets of DS is $\mathrm{L_{DS1}} = \{(\mathrm{D}:4), (\mathrm{B}:3), (\mathrm{C}:3), (\mathrm{T}:3)\}$. When the original dataset DS increased by new transaction dataset ds, minimum support reduce to $\mathrm{min\_sup}' = 0.4$ from $\mathrm{min\_sup} = 0.6$ at the same time. In the Figure2, subfigure a) shows the increment FP-tree with the $\mathrm{min\_sup}' = 0.4$ and frequent 1-itemsets of ds is $\mathrm{L_{ds1}}' = \{(\mathrm{B}:2), (\mathrm{N}:2), (\mathrm{Y}:2), (\mathrm{Z}:2)\}$.

(2) Now we want to know what items the frequent 1-itemsets $\mathrm{L_{DS1}}$ of new dataset $\mathrm{DS} + \mathrm{ds}$ includes.

The subfigure b) shows VSIFP-tree model as minimum support declines and frequent 1-itemsets of DS is $\mathrm{L_1}' = \{(\mathrm{D}:4), (\mathrm{B}:3), (\mathrm{C}:3), (\mathrm{T}:3)\}$

Table 1. The Original Dataset of Transaction Database ($\mathrm{min\_sup} = 0.6$)

Transaction ID	Items											Ordered items
	A	B	C	D	E	M	N	S	T	Y	Z	
001	A	B	C	D	E			S		Y		D B C
002		B	C	D		M			T			D B C T
003			C	D	E			S	T			D C T
004	A	B					N			Y	Z	B
005				D		M			T			D T

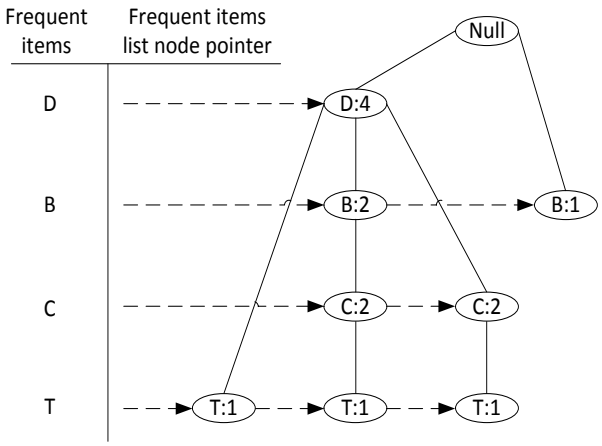


Figure 1. The Original FP-Tree Model ($\mathbf{min\_sup = 0.6}$)

Table 2. The Increment Dataset of Transaction Database ($\mathbf{min\_sup' = 0.4}$)

Transaction ID	Items											Ordered items
	A	B	C	D	E	M	N	S	T	Y	Z	
006		B				M	N			Y	Z	B N Y Z
007	A						N	S	T			N
008		B			E					Y	Z	B Y Z

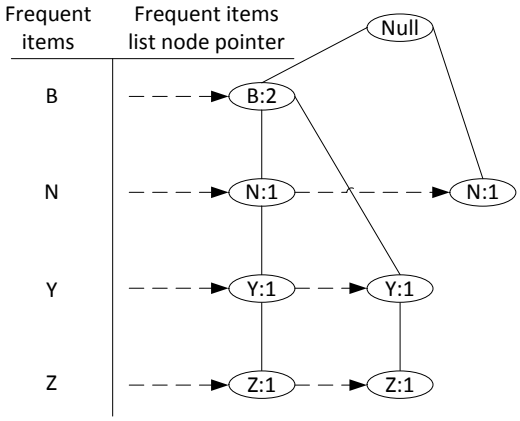


Figure 2. The VSIFP-Tree Model ($\mathbf{min\_sup' = 0.4}$)

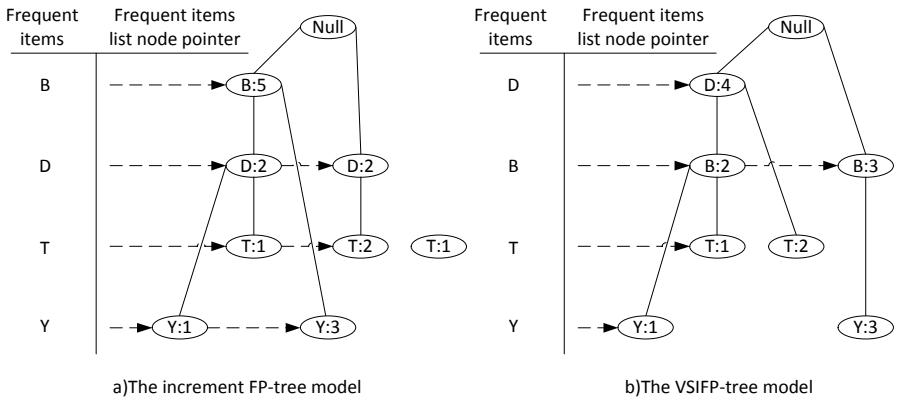


Figure 3. Tree Structure Contrast after Data Merging and Support Changing

3. Parallel Computing for Large-Scale Dataset

3.1 PVSIFP-Growth Algorithm Description Based on MapReduce

In the practical application of large-scale dataset, the object of the association rule mining is often a huge centralized or distributed data sources. If the single machine for association rules mining, storage capacity and the mining efficiency is bound to become a bottleneck in the process of mining, which can't meet the needs of large data mining. But with the rapid development of parallel computing, using the method of parallel processing mass data mining problems can not only meet the needs of huge amounts of data mining, but also can greatly improve the mining efficiency. As a result, the parallel data mining has become an effective way to solve big data mining. On the other hand, in a lot of practical data mining application, often also has the problem of incremental updating. Many applications in the field of database are all in the constantly updated, resulting in the patterns of original excavated or create new pattern. In order to solve this problem, must be on the original model, combining with the new data mining again, namely incremental association rules mining processing [9].

For stand-alone environment cannot meet the requirements of large data mining problems, the researchers began to consider through the distributed parallel computing environment to solve the problem. Parallel programming model based on MapReduce by Google [10], has a strong function and a variety of advantages, and many studies of MapReduce cluster architecture improvement plans are put forward [11], to provide a good direction for the research of parallel association rules and platform. LI, used the MapReduce calculated model, which is applied to the FP-Growth algorithm, and proposed an algorithm of parallel computing model based on MapReduce (parallel the frequent pattern Growth algorithm, PFP-Growth) [12]. For parallel incremental updating of association rules research, qiu-yang Chen uses traditional single parallel computing model, based on message passing interface (message passing interface, MPI) puts forward a parallel incremental updating association rules algorithm, the parallel pruning and fast updating, PPFUP).

In view of the huge amounts of data mining and incremental updating problems, this paper designs and realizes a MapReduce and VSIFP-growth based parallel variable support incremental updating algorithm for mining association rules[13]. The algorithm model is obtained by using the MapReduce, to update VSIFP-tree incrementally in a distributed computing environment and optimize too much time consumption problem caused by scanning the original database for every time of incremental updating data. This algorithm can scan the original transaction database to a minimum and raise the efficiency of parallel computing. At the same time, we adopt the method of dynamic load balancing adjustment group to optimize the itemsets grouping problems in the process of distributed computing, making the whole distributed computing can achieve maximum load balancing. Finally, to access the performance of PVSIFP-growth, we conducted experiments to observe the influence of various support threshold and dataset size values in PVSIFP-growth.

3.2 PVSIFP-Growth Algorithm Modelling Procedure Based on Mapreduce

In a distributed computing environment, to build the initial PVSIFP-tree on the original transaction database DB, mainly includes the following three steps.

Step 1 Partitioned indexes construction of project support count and project. In Hadoop, Mapper function based on HDFS file fragmentation to process the transaction record for the unit, therefore, Mapper function can be used to calculate the support degree of each shard project counts, and build the shard of partitioned

indexes of the project. For each shard, the Mapper function input for relative fragmentation at the beginning of offset and transaction records, namely the key value pair for $\langle \text{key} = \text{offset}, \text{value} = T_i \rangle$. The pseudo code of this step is shown below.

```
/* The Mapper procedure to count the index of the blocks*/
Class: CountingAndIndexMapper
itemBlockIndexs[] = new Array();//initialize the block indexes
blockSize = 0; // initialize the size of block indexes
/*Mapper initialization*/
Function: setup(context)
blockSize = readBlockSize(context);// read parameters
/*Mapper processing
* Build block indexes for each items in the transaction itemset
* Input: Ti and offset of transaction records */
Function: Mapper (key = offset, value = Ti)
Foreach item Ii in Ti do
Callcollect(< Ii,1 >);
bockID = ceil(key/blockSize);
buildBlockIndexs(Ii, blockID); end;// build block indexes
/*Mapper clean: Write partitioned indexes to the local file system */
Function: cleanup(context)
writeToLocalFS(itemBlockIndexs)
/* The Reducer procedure */
Class: Counting Reducer
Function: reducer(key = Ii, value = List(Ii))
Foreach item '1' inList(Ii) do
C = C + 1; end;
Callcollect(< Ii,C >);//output count
```

Assume that item sets in the fragmentation class DB_i is $IS_i = \{I_{i1}, I_{i2}, \ldots, I_{im}\}$. The block size for blockSize of partitioned indexes as DB_i is segmented into $BS = \text{ceil}(\text{sizeof}(DB_i/\text{blockSize}))$ a block, the ceil for integer arithmetic, blockID_j is the first j block. Suppose that project I_{ik} in the subdivision of the fragmentation class DB_i support count is $\text{Counting}(DB_i)$, and it will be distributed in the m blocks $(m \leq BS)$, each block count is $\text{Counting}(I_{ik})_{\text{blockID}}$. At this point I_{ik} partitioned indexes are shown in Figure 4.

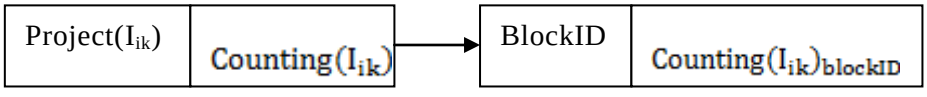


Figure 4. Block Index of the Project I_{ik}

For each records(T_i) of project(I_{ik}) in Mapper function of MapReduce, if the offset of record belongs to the last block of the I_{ik} partitioned indexes, then the block index count need to plus "1". If not, create a new node after the index list node, the node ID is the serial number of blocks to record(T_i), count to 1. After all local transaction record processed, we can obtain partitioned indexes of local projects [14].

Step 2 project group. Get all the frequent itemsets by using project support count and deposited in the table called FList. Then use a grouping scheme to divide the project(FList) into Q group, and save each group project in table called GList. Optimization grouping scheme must make each packet to keep a balanced load.

Load balancing is to make the time of consumption to be close to each group while computing frequent itemsets on the VSIFP-tree. If we want to calculate the optimal grouping, then it must build all the possible group to tree structure and mining the tree. However, because of so many the possibility of grouping, large amount of the process of calculation, and current balance may become unbalanced after incremental updating, so this algorithm need to maintain the load balancing in a dynamic environment.

This paper employ a dynamic load balancing strategy for grouping of project. The strategy is mainly to dynamically adjust the new frequent itemsets grouping using the last result to calculate the load of each group each time incremental updating data. So we can dynamically adjust the group load and make the group to reach a final equilibrium. But when calculate on the original transaction dataset, there is no incremental data for the last time incremental updating calculation, therefore we choose the stochastic grouping method for grouping FList. The complexity of this randomized computational method is O(I), which can be finished in a stand-alone environment. The pseudo code to build PVSIFP-tree for each grouping is shown below.

```
/* The Mapper and Mapper procedure of PVSIFP-Growth*/
Class: PVSIFP_Growth
GList = null;
Function: setup(context)
GList←readGList(context) //read GList from the configuration file
/*Mapper processing
* Divide transaction into groups according to the GList
* Input: T_i and offset of transaction records */
Function: mapper(key, value = T_i)
for i = 0 to GList.size do
if(T_i ∩ GList[i] ≠ null) then
Callcollect(< i, T_i >);// output to a specific group
end if; end;
/* Reducer processing
* Build PVSIFP-tree for each group
* Input:Group number gid and its records S(T_i) */
Function: reducer(key = gid, value = S(T_i))
localFUFPTree = null;
foreach T_i in S(T_i)
CallbuildPVSIFPTree(T_i, localPVSIFPTree);
end;
CallwriteTreeToLocalFS(localFUFPTree);//writes each PVSIFP-tree into a local file
CallFPGrowth(localFUFPTree, gid); /running FP-Growth algorithm on the tree of each
group
```

Step 3 Build VSIFP-tree for each group(GList). First of all, distribute the transactions in the database to the corresponding group according to the GList group. The distribution operation can be done in a Mapper function, which can output

multiple key value pair$(key = groupID, value = T_i)$. At this point, all the transaction records which belongs to the same group will be distributed to the same Reducer. And then in each group of Reducer function, construct a VSIFP-tree to the transaction record data of the group. On this occasion, mine the frequent itemsets for the group, and finally combine each group of frequent itemsets to get the global frequent itemsets. After building each grouping VSIFP-tree, we need to store VSIFP-tree serialization of each group to the file system, for incremental updating.

Through the above three steps, we finished the original transaction dataset DS of mining frequent itemsets. We calculate not only all the frequent itemsets of the current dataset, but also record and store the support degree of each project, each block index record stored on the server project, project group GList data and each grouping VSIFP-tree.

4 Experimental Results and Analysis

4.1 The Experimental Data and the Environment

In this paper, we design the experiment to analyze PVSIFP-Growth performance by comparing the running time of PVSIFP-Growth algorithm and FP-Growth algorithm on two datasets. PVSIFP-Growth algorithm and FP-Growth algorithm are realized by Matlab (R2014b). This article adopted Webdocs.data as the experimental dataset provided by http://fimi.ua.ac.be/data [15].

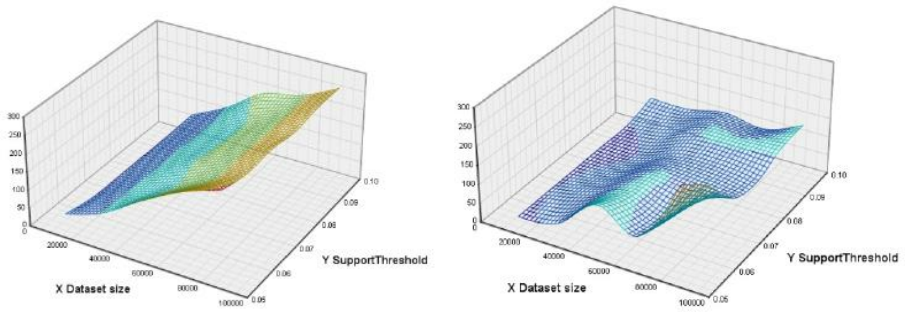
The parallel computing environment in this experiment is made up of four nodes master-slave Hadoop cluster. One of the nodes as the namenode and jobtracker, other nodes as a datanode and tasktracker. And all nodes keep isomorphism of hardware configuration and software configuration. The processor is Intel Core duo 2.6 GHz, computer memory size is 8 GByte, the operating system is Ubuntu14.04, and Hadoop used 2.6.0 version, which keep the default cluster configuration parameters.

4.2 Incremental Calculation Performance Test of the PVSIFP-Growth Algorithm

Under the condition of the support threshold rise, because PVSIFP-Growth just need to centralized seeking for new frequent pattern in frequent pattern, running time of PVSIFP-Growth is obviously faster. Here we compare the running time of two kinds of algorithm only when the support threshold value fall and data scale increase [16].

The first dataset is transactional sparse dataset(Retail), and each record contains 1 to 50 transaction items, 88162 records in total.

The Figure4(a) and Figure4(b) respectively shows the running time comparison of FP-Growth algorithm and PVSIFP-Growth algorithm. In this experiment the scale of dataset vary from 17625 to 88126(step size 17625), and the support threshold decreased from 0.1 to 0.05(step size 0.01). In this figure, X coordinates stand for the dataset size(record), Y coordinates stand for the support threshold, Z coordinates is the run time. We can conclude that the running time of PVSIFP-Growth algorithm is less than the FP-Growth.

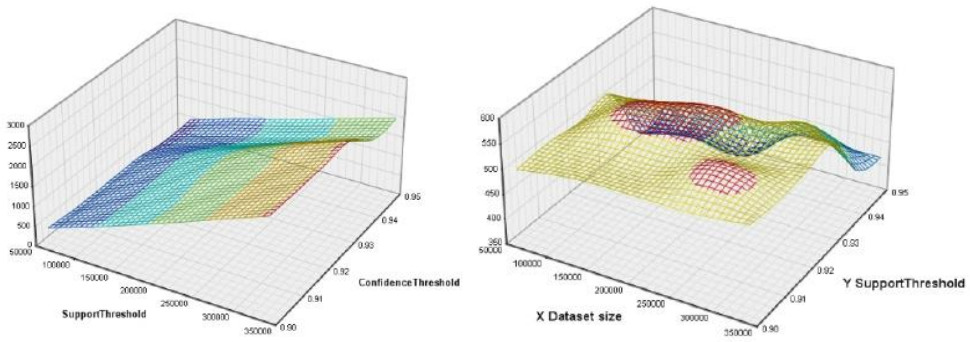


(a) (b)

Figure 5. The Running Time of FP-Growth and VSIFP-Growth Algorithm on Retail

The second dataset is transactional dense dataset (Accidents), and each record contains 1 to 80 transaction items, 340183 records in total.

The Figure5(a) and Figure5(b) respectively shows the running time comparison of FP-Growth algorithm and PVSIFP-Growth algorithm. In this experiment, the scale of data set vary from 68036 to 340183(step size 68036), support threshold decreased vary from 0.95 to 0.9(step size 0.01). The X coordinate stand for the data set size (record), Y coordinates stand for the support threshold, Z coordinates is run time. We may draw the conclusion that the running time of PVSIFP-Growth algorithm is less than the FP-Growth.



(a) (b)

Figure 6. The Running Time of FP-Growth and VSIFP-Growth Algorithm on Accidents

5. Conclusion

By incorporating the VSIFP-Growth (Improved FP-Growth) algorithm and the parallel computing mining technique into VSIFP-Growth, we propose the PVSIFP-Growth algorithm for frequent itemsets generation. The major advantages of VSIFP-Growth and the parallel computing based on MapReduce are that they reduce the need to rebuild frequent pattern tree and facilitate the task of tree construction. The PVSIFP-Growth algorithm inherit VSIFP-Growth algorithm, which can generate association rules when both database increase or decrease and minimum support of dataset changed. The memory requirement of PVSIFP-Growth on a single processor is also lower than that of FP-Growth. Experimental results showed that our PVSIFP-Growth algorithm is more than an order of magnitude faster than the FP-Growth algorithm. However, VSIFP-Growth incurs a potential problem in the parallel

computing mining method. An appropriate dataset scale value and datanode count in this method are important, because they affects the mining performance of our algorithm. As a result, our future work will involve finding an appropriate dataset scale value and datanode count between the conditional VSIFP-Growth algorithm and the parallel computing mining technique in the PVSIFP-Growth mining method.

Acknowledgement

First of all, I would like to show my deepest gratitude to my doctoral supervisor, Prof. Li Sheng-Lin, a respectable, responsible and resourceful professor. I learned a lot of scientific research and academic, engineering projects knowledge from him, which will become the wealth of my life. Secondly, thanks to Prof. Wang Zhao-Xia for helping me, who has provided me with valuable guidance in every stage of the writing of this thesis and taught me association rules algorithm and the main technical points of the large data parallel computing. With her help, I have conducted a follow-up design and experiment of algorithm improvement and completed this paper. Finally, thanks for laboratory classmates, at ordinary times we help each other overcome many difficulties. In addition, thanks to Li Chang-Lin and Li Xue-Long, who have helped me realize part of the algorithm code design.

References

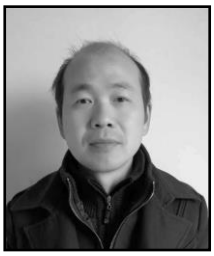
[1] W. C. David, "Maintenance of discovered association rules in large databases: An incremental updating technique", Data Engineering, 1996. Proceedings of the Twelfth International Conference on. IEEE, **(1996)**.
[2] C. D. W. Lok, S. D. Lee and B. Kao, "A General Incremental Technique for Maintaining Discovered Association Rules", DASFAA, vol. 6, **(1997)**.
[3] A. N. Fazil, A. U. Tansel and E. Arkun, "An efficient algorithm to update large itemsets with early pruning", Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, **(1999)**.
[4] L. C. Hung, C. R. Lin and M. S. Chen, "Sliding-window filtering: an efficient algorithm for incremental mining", Proceedings of the tenth international conference on Information and knowledge management. ACM, **(2001)**.
[5] Z. Y. Quan, Z. H. Sun and X. J. Ji, "Incremental updating algorithm based on frequent pattern tree for mining association rules", Chinese Journal of Computers-Chinese Edition, vol. 26, no. 1, **(2003)**, pp. 91-96.
[6] C. Qiuyang and L. Jin, "Research on optimization of parallel incremental updating algorithm for association rules", Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications), vol. 47, no. 14, **(2011)**.
[7] T. Lu, J. Hong and S. Qiuzi, "An Improved Incremental Updating Algorithm for Association Rules", Computer Applications and Software, vol. 29, no. 4, **(2012)**, pp. 246-248.
[8] L. K. Chung, I. E. Liao and Z. S. Chen, "An improved frequent pattern growth method for mining association rules", Expert Systems with Applications, vol. 38, no. 5, **(2011)**, pp. 5154-5161.
[9] X. L. Xin, "Research on The Optimization of Enrollment Data Resources Based on Cloud Computing Platform", Review of computer engineering studies, http://dx.doi.org/10.18280/rces.020203, vol. 2, no .2, **(2015)**, pp. 9-12.
[10] D. Jeffrey and S. Ghemawat, "MapReduce: simplified data processing on large clusters", Communications of the ACM, vol. 51, no. 1, **(2008)**, pp. 107-113.
[11] J. Jing, "A new multi-master framework of MapReduce", Journal of Beijing University of Posts and Telecommunications, vol. 35, no. 4, **(2012)**, pp. 89-93.
[12] L. Haoyuan, "Pfp: parallel fp-growth for query recommendation", Proceedings of the 2008 ACM conference on Recommender systems. ACM, **(2008)**.
[13] R. Jin, C. Kou and R. Liu, "Bi-clustering Algorithm of Differential Co-Expression for Gene Data", Review of computer engineering studies, http://dx.doi.org/10.18280/rces.010102, vol. 1, no. 1, **(2014)**, pp. 7-12.
[14] Y. Yong and G. S. Song, "Parallel and incremental updating algorithm for association rules based on mapReduce", Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), vol. 5, **(2014)**, pp. 19.
[15] B. Goethals and M. J. Zaki, "Frequent Itemset Mining Dataset Repository", < http://fimi.ua.ac.be/data>, **(2015)**.

[16] Z. Chun, "The Financial Risk Analysis and Forewarning Research Based on Data Mining Technology", Diss. Beijing University of Chemical Technology, **(2012)**.

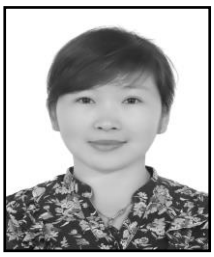
Authors

Guo Yu-Dong, was born in Heilongjiang province, China, in 1987. He has been studying for the Ph.D. degree of Military in the Department of military logistics information and logistics engineering, Logistics Engineering Institute. He is working on information system integration and big data mining.

Li Sheng-Lin, received B.S. degree in Mathematics from Southwest University, China, in 1985, and Ph.D. degree in Logistical Engineering University of P.L.A China, in 2008. He is working on information management engineering and computer science.



Li Yong-Zhi, was born in Hubei province, China, in 1977. He has received B.S. degree in Oil-Gas Storage and Transportation Engineering from Logistical Engineering University of P.L.A China. He is working on information management engineering and computer science.



Wang Zhao-Xia, received Ph.D. degree in Tsinghua University. She is a master supervisor and now working in Department of military logistics information and logistics engineering, Logistics Engineering Institute, Chongqing. Her main research direction is data management and information system architecture.

Zeng Li, received Bachelor of Electrical Engineering Hunan University in 1982. She won "Excellent Teacher" honor at Chongqing University of Science and Technology in 1993. Now she is working in School of Civil Engineering and Architecture, Chongqing University of Science and Technology. Her research interests is construction facilities engineering and design of architectural hydropower.