The Study of Improved FP-Growth Algorithm in MapReduce

Hong SUN
University of Shanghai for Science and Technology
Shanghai Key Lab of Modern Optical System
Shanghai, China
e-mail: sunhong_sh@sohu.com

Huaxuan Zhang
University of Shanghai for Science and Technology
Shanghai Key Lab of Modern Optical System
Shanghai, China
e-mail: 963422575@qq.com

Shiping Chen
University of Shanghai for Science and Technology
Shanghai, China
e-mail: chensp@usst.edu.cn

Chunyan Hu
University of Shanghai for Science and Technology
Shanghai, China
e-mail: hhuchy@163.com

***Abstract*—As FP-Growth algorithm generates a great deal of conditional pattern bases and conditional pattern trees, leading to low efficiency, propose an improved FP-Growth(IFP) algorithm which firstly combine the same patterns based on the situation whether the support of the transaction is greater than the minimum support(min_sup) to mine the frequent patterns. Thus the IFP cuts down on the space and improves the efficiency. It also makes it easy to be paralleled. Further more, combine the IFP algorithm with the MapReduce computing model, named MR-IFP(MapReduce-Improved FP), to improve the capability to deal with the mass data.**

Keywords: FP-Growth, IFP algorithm, MapReduce

***Introduction* (Heading 1)**

I. INTRODUCTION

Association Rules is a key problem in data mining. Agrawal with his partners firstly proposed association rules between item-sets in clients' transaction database in 1993. Then much work about association rules mining have been done[1]. Agrawal et. al. presented the most classic Boolean algorithm called Apriori [2] .This algorithm retrieves all the frequent item sets whose support is no less than the minimum support(min_sup) in database of transaction sets through iteration. The min_sup is set by the client in advance. Then the K-itemsets generate the (k+1)-itemsets, and those whose support are less than the min_sup will be pruned at the same time. The work has been done when the (k+1)-itemsets is null. However the Apriori algorithm will produce a large number of candidates while computing, and it scans the database for so many times as to reduce the efficiency. Concerning this issue, Jiawei Han put forward the FP-Growth algorithm. This algorithm scans the database for only twice, saving much time and space. But as the database is growing at an increasingly speed, the classic FP-Growth produces a great deal of conditional pattern bases and conditional pattern trees that the memory could not meet the need and it still costs time. As a result, paralleling the FP-Growth and promoting the efficiency became two leading ways in association rule mining. [3] proposes an improved FP-Growth algorithm that combines the sub-tree with same patterns. It uses depth-first method and doesn't need generate so many pattern trees. While it saves space and speeds up the computing, its top-down merging and mining with regulation strategy again lower the speed. This paper makes a further improvement and uses the MapReduce programming model, proposing the Improved FP-Growth algorithm in MapReduce, MR-IFP algorithm. The contributions of this paper include not only the efficiency promotion and memory saving , but also its capability dealing with mass data.

II. FP-GROWTH ALGORITHM

The basic idea of FP-Growth algorithm is taking advantage of the tree to condense the transaction, and remain the relationship between the property of the transaction in the same time. This algorithm will not generate an itemset of candidates, and mined the date by increasing the frequent itemset[4,5]. The important step of FP-Growth algorithm is the process to construct the FG tree, which needs to scan the transaction itemset twice: scan the database of transaction T once to find out the frequent 1-itemset L, then arrange the support count in descending order to get the L_NULL;take the "Null" as the root node when scan the transaction itemset for the second time, then construct the FP-tree base on L_NULL.

Here is an example to show how to construct FP-tree. Transaction itemset is shown in Table 1, define min_sup=20%, meaning that the minimum support is 2:

TABLE I. TRANSACTION ITEMSET T

Tid	Items
1	I1,I2,I5
2	I2,I4
3	I2,I3
4	I1,I2,I4
5	I1,I3
6	I2,I3
7	I1,I3
8	I1,I2,I3,I5
9	I1,I2,I3

1) Scan the transaction itemset T, find out items whose support can meet the condition, combine those itemsets to get the frequent 1-itemset L, then arrange L in descending order base on the support count to get L_NULL as shown in Table 2:

TABLE II. L_NULL

Item-name	Support-counts
I2	7
I1	6
I3	6
I4	2
I5	2

2) Construct the original FP-tree and take the "Null" as the root node.

3) Every row in the L_Null stands for a frequent itemset, point the corresponding pointer to it's node in the FP-tree.

4) Traverse the transaction itemset T, adjust the sequence of all the itemset in T according to L_null, shown in Table 3:

TABLE III. ADJUSTED TRANSACTION ITEMSET T

Tid	Items
1	I2,I1,I5
2	I2,I4
3	I2,I3
4	I2,I1,I4
5	I1,I3
6	I2,I3
7	I1,I3
8	I2,I1,I3,I5
9	I2,I1,I3

Build a branch for each transaction in Table 3. Share the path if the path of branch can be shared, and record the number of shared transaction in each node. The constructed FP-tree is shown in Fig.1:

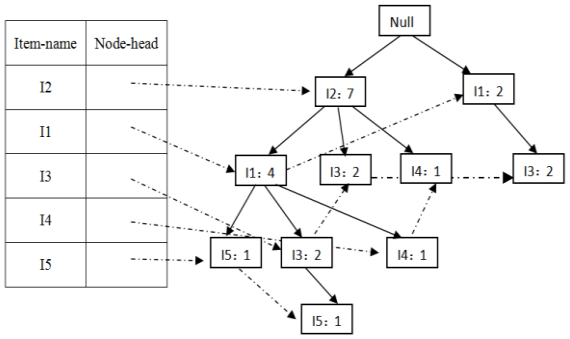


Figure 1. FP-tree

The next step is mining the frequent itemsets from the FP-tree after constructing the original FP-tree. The steps are as follows:

1) Produce conditional pattern base for every node in the FP-tree;

2) Build the corresponding conditional FP-tree by the conditional pattern base;

3) Mine the conditional FP-tree recursively and increase the frequent itemset belong to it in the same time: produce the involved frequent itemset immediately if the conditional FP-tree only contain one path. Otherwise increase the , (is the suffix pattern and Ei is the last item of L_null, the minimum item of support count). Then construct the conditional pattern base and conditional FP-tree (The conditional pattern base is all the branch which take Ei as their leaf node in FP-tree. The conditional FP-tree of is a new FP-subtree taking the conditional pattern base as it's transaction and constructed in the same way of the original FP-tree).

The thus obtained conditional pattern base and frequent itemset are shown in Table 4:

TABLE IV. CONDITIONAL PATTERN BASE AND FREQUENT ITEMSET

item	conditional pattern base	frequent itemset
I5	<I2,I1:1>, <I2,I1,I3:1>	<I2,I5:2>,<I1,I5:2>,<I2,I1,I5:2>
I4	<I2,I1:1>,<I2:1>	<I2,I4:2>
I3	<I2,I1:2>,<I2:2>,<I1:2>	<I2,I3:4>,<I1,I3:4>,<I2,I1,I3:2>
I1	<I2:4>	<I2,I1:4>

Unlike the Apriori algorithm,FP-Growth won't scan the database for many times and produce lots of candidate itemsets. FP-tree can save all the information used to mine the frequent itemset, long pattern belong to each transaction won't be cut off, the higher a item's frequent is the easier it can be shared, the number of node contained in tree will not more than the data of semi-intoxicated item in the database. But FP-Growth algorithm will produce large number of conditional pattern tree recursively in mining frequent pattern. It needs too large memory to apply in broad-scale database.

III. IMPROVED FP-GROWTH ALGORITHM

A. *Items-constraint frequent pattern mining algorithm, ICFP-Mine[3]*

The FP-Growth algorithm, based on FP-Tree, is unable to distinguish whether one pattern is frequent while mining each sub-tree as the same patterns might scatter in different trees. As a result, the algorithm has to create a large number of conditional pattern trees and conditional pattern bases recursively in the process of mining the frequent patterns. [3] proposes an items-constraint frequent pattern mining algorithm, ICFP-Mine. It combines the same patterns in one tree, making it direct to judge whether the sub-tree is frequent. There is no need to build frequent pattern tree. The next step is to mine the frequent itemsets using the depth-first method and mining with regulation strategy.

The ICFP-Mine is superior to the traditional FP-Growth in memory occupancy and time costs. But its top-down merging and mining with regulation strategy limit its computing speed. The premise items-constraint is not pervasive. This paper uses the idea of combining and makes a further improvement.

B. *Improve FP-Growth Algorithm*

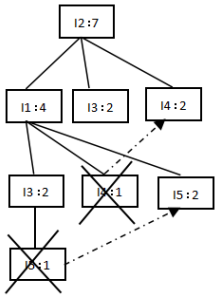
Improved FP-Growth algorithm: use the bottom-up merging firstly, and then depth-first mining.

Before merging, we focus on the leaf nodes whose support are less than the min_sup: If the father nodes or brothers of father nodes have the same pattern with the leaf nodes, combine them and then prune these leaf nodes; If there is not, just prune the leaf nodes. Next we focus on other leaf nodes(support is no less than the min_sup), the operation is merging the leaf nodes with the fathers' or brothers' of fathers'(have same pattern). We keep these leaf nodes instead of pruning them.

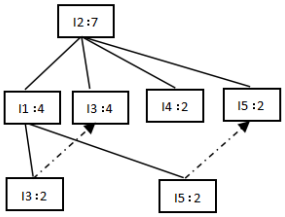
We use the same example to illustrate the progress. The FP-Tree is as Fig.1 shown. The Null node has two leaf nodes:I2, I1. We deal with these two sub-trees respectively for further paralleling computing.

For I2 sub-tree, consider the leaf nodes with support less than min_sup. Follow the bottom-up principle, start with the farthest leaf node I5. The support of I5 is less than min_sup, joint it with the upper level nodes whose name is also I5(the same pattern). Among three branches of <I2,I1>, the support of I4 is less than the min_sup, then joint it with the upper I4. After that, prune the leaf nodes I5 and I4 at the bottom level, as shown in Fig.2(a).

Now we can focus on the leaf nodes that has support greater or equal to the min_sup on the basis of Fig.2(a). At this bottom there are two nodes I3 and I5. We joint these two leaf nodes with upper I3 and I5, and keep them instead of pruning them, as shown in Fig.2(b).



(a) Processing of leaf nodes with support less than min_sup in I2 sub-tree



(b) Processing of leaf nodes with support less than min_sup in I2 sub-tree

Figure 2. IFP-Tree Mining

After combination, mine the frequent itemsets and depth first. We can get the result directly from the Fig.2(b). Follow the depth-first principle: {<I1,I3:2> , < I1,I5:2> , <I2,I1,I3:2> , <I2,I1,I5:2> , <I2 , I1:4> , <I2,I3:4> , <I2,I4:2>, <I2,I5:2>}.

The same operation with I1 sub-tree. Get the frequent itemsets: {<I1,I3:2>}.

Finally joint these two itemsets. Combine those items having same prefix path: {<I1,I3:4> , < I1,I5:2> , <I2,I1,I3:2> , <I2,I1,I5:2> , <I2,I1:4> , <I2,I3:4> , <I2,I4:2>, <I2,I5:2>}. The result is in accordance with the one in Table 4.

IFP algorithm adopts the bottom-up method to combine the same pattern and depth-first strategy to mine frequent itemsets. It is less complex and much faster than ICFP-Mine. It produces less intermediate results and needs less memory. As it is not items-constraint, the algorithm is more applicable. If there is a demand of constraint, just add it before pattern merging.

IV. IFP-ALGORITHM IN MAPREDUCE

Although IFP has advantages over other traditional FP-Growth algorithm, dealing with mass data with TB or larger size is still inefficient. The solution is to parallel the computing to several nodes. Cloud computing is born with the ability of handling large-scale data and computing. So deploying the improved FP-Growth algorithm to the cloud platform is a promising way to solve the problem[6].And MapReduce is widely adopted by the cloud computing as a programming mode. Working the improved FP-Growth algorithm with the MapReduce, the steps are as followed:

1) Calculate the frequency of the transaction database via one MapReduce task. FP-Growth algorithm has to scan the database once before building the tree. Generate the item header table based on the calculation in a descending order[9]. Then complete a FP-Tree.

2) Master node splits the FP-Tree to several pieces. As the example shown in Fig.1, Master should split the FP-Tree into two pieces as I2 sub-tree and I1 sub-tree, send them to two slave nodes for mining using IFP algorithm.

3) Two salve nodes send the result back to the Master. The Master combines these two frequent itemsets as the part 3.2 introduced.

As for other FP-Trees, the Null node may have many child nodes, just split them into appropriate pieces to relevant slave nodes. In actual mining, the FP-Tree may be too large that after splitting the sub-tree is still large for one node to compute. So we can go on splitting until meet the demand of memory and speed. Then send them back to upper nodes to gather and get the final result[5].

V. CONCLUSION AND PERSPECTIVE

This paper makes an optimization of traditional FP-Growth, named IFP(Improved Frequent Pattern) algorithm. This algorithm joints the same patterns based on the situation whether its support is less than the min_sup after building the FP-Tree. It uses the depth-first method mining the frequent itemsets and saves a great deal of space. It improves the efficiency considerably and is easy to parallel. Further more, the work that combines the improved FP-Growth algorithm with MapReduce programming model implements parallelization, greatly improves its capability dealing with

mass data. The next step is to conduct experiments on Hadoop and take the redundant situation into consideration.

ACKNOWLEDGMENT

The National Natural Science Foundation of China(61170277).

Supported by Innovation Program of Shanghai Municipal Education Commission(12zz137).

Top Discipline Construction Projects of Shanghai(S1201YLXK).

The Innovation Project of Shanghai Graduate Education(No.SHGEUSST1301

REFERENCES

[1] Cai Weijie, Zhang Xiaohui, Zhu Jianqiu, Zhu Yangyong. Survey of association rule generation [J]. Computer Engineering, 2001, 27（5）: 31-33

[2] Wei Zhang, Hongzhi Liao, Na Zhao.Research on the FP-Growth algorithm about Association Rule Mining[C] //International Seminar on Business and Information Management,2008:315-318

[3] Zhao Xiaomin, He Songhua, Li Xianpeng, Yin Bo. Improved FP-Growth algorithm and its applications in the business association[J].Computer Applications, 2008,28（9）: 2341-2348

[4] Yang Yun, Luo Yanxia. Improved algorithm based on FP-Growth[J].Computer Engineering and Design, 2010,31（7）: 1506-1509

[5] Tan Kelin, Sun Zhihui. An Algorithm of Mining FP-tree in Parallel [J].Computer Engineering and Applications, 2006,13:155-157

[6] Zhu Xiaofeng, Li Linjuan, Xu Xiaolong, Chen Jianxin. MapReduce Based Association Rule Incremental Updating Algorithm [J].Computer Technology and Development, 2012,22（4）: 115-118

[7] Liu Peng. Cloud Computing［M］.Beijing: Publishing House of Electronics Industry, 2010.

[8] Ding Linlin, Xin Junchang, Wang Guoren, Huang Shan. Efficient Skyline Query Processing of Massive Data Based on Map-Reduce [J].Chinese Journal of computers, 2011,34（10）: 1785-1796

[9] Lv Xueji, Li Longshu. Research on Improved FP-Growth Algorithm with MapReduce[J]. Computer Technology and Development, 2012, 22(11):123-126