Computer Life (CPL) ISSN: 1819-4818
DELIVERING QUALITY SCIENCE TO THE WORLD

The algorithm for mining association rules in big data

Bo He [a], Jiru Zhang [b]

School of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China;

[a] 29659807@qq.com, [b] heboswnu@sina.com

Abstract: The algorithm for mining association rules in big data was proposed, namely, MGFI algorithm. Firstly, local frequent itemsets were computed by mapreduce. Secondly, the data were collected. Finally, global frequent itemsets were gained by mapreduce. Theoretical analysis and experimental results suggest that MGFI algorithm is fast.

Keywords: Data mining; global frequent itemsets; big data; mapreduce; FP-tree.

1. INTRODUCTION

There are some data mining algorithms[1], such as CD [2] and FDM [3]. Most of them adopt Apriori-like algorithm, so that a lot of candidate itemsets are generated and the database is scanned frequently. This causes heavy communication traffic among the nodes. Aiming at these problems, this paper proposes the algorithm for mining association rules in big data, namely, MGFI algorithm.

Big data is being generated by everything around us at all times. Every digital process and social media exchange produces it. Systems, sensors and mobile devices transmit it. Big data is arriving from multiple sources at an alarming velocity, volume and variety. To extract meaningful value from big data, you need optimal processing power, analytics capabilities and skills.

MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

2. RELATED DEFINITION AND THEOREM

2.1 Description of mining association rules

The global transaction database is *DB*, the total number of tuples is *M*. Suppose P_1, P_2,…, P_n are *n* nodes, node for short, there are M_i tuples in DB_i, if DB_i (i=1,2,…,n) is a part of *DB* and stores in P_i, then $DB=\bigcup_{i=1}^{n} DB_i$, $M=\sum_{i=1}^{n} M_i$.

mining association rules can be described as follows: each node P_i deals with local database DB_i, and communicates with other nodes, finally, global frequent itemsets of global transaction database are got.

2.2 Related Definition

Definition 1 For itemsets X, the number of tuples which contain X in local database $DB_i(i=1,2,\dots,n)$ is defined as local frequency of X, symbolized as $X.si$.

Definition 2 For itemsets X, the number of tuples which contain X in global database is global frequency of X, symbolized as $X.s$.

Definition 3 For itemsets X, if $X.si \geq min\_sup*M_i(i=1,2,\dots,n)$, then X are defined as local frequent itemsets of DB_i, symbolized as F_i. *min_sup* is the minimum support threshold.

Definition 4 For itemsets X, if $X.s \geq min\_sup*M$, then X are defined as global frequent itemsets, symbolized as F.

2.3 Related Theorem

Theorem 1 If itemsets X are local frequent itemsets of DB_i, then any nonempty subset of X are also local frequent itemsets of DB_i.

Corollary 1 If itemsets X are not local frequent itemsets of DB_i, then the superset of X must not be local frequent itemsets of DB_i.

Theorem 2 If itemsets X are global frequent itemsets, then X and all nonempty subset of X are at least local frequent itemsets of a certain local database.

Theorem 3 If itemsets X are global frequent itemsets, then any nonempty subset of X are also global frequent itemsets.

Corollary 2 If itemsets X are not global frequent itemsets, then superset of X must not be global frequent itemsets.

2.4 FP-tree and FP-growth algorithm[4,5]

Definition 5 FP-tree is a tree structure defined as follow.

(1) It consists of one root labeled as ”null”, a set of itemset prefix subtrees as the children of the root, and a frequent itemset header table.

(2) Each node in the itemsets prefix subtree consists of four fields: item-name, count, parent and node-link.

(3) Each entry in the frequent-item header table consists of three fields: i,Itemname. ii, Side-link, which points to the first node in the FP-tree carrying the item-set. iii, Count, which registers the frequency of the item-name in the transaction database.

FP-growth algorithm adopts a divide-and-conquer strategy. It only scans the database twice and does not generate candidate itemsets. The algorithm substantially reduces the search costs. The study on the performance of the FP-growth shows that it is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm.

3. MGFI ALGORITHM

3.1 Design Thoughts of MGFI Algorithm

MGFI sets one node P_0 as the center node, other nodes compute local frequent itemsets with FP-growth algorithm and mapreduce. Then the nodes send local frequent itemsets F_i to the center node P_0. P_0 gets local frequent itemsets F'($F' = \bigcup_{i=1}^{n} F_i$) which are pruned by the searching strategies of top-down and bottom-up. P_0 sends the remain of F' to other nodes. For local frequent itemsets $d \in$ the remain of *F'*, P_0 collects local frequency *d.si* of *d* from each node and gets global frequency *d.s* of *d.* Global frequent itemsets are gained by mapreduce.

F' are pruned by the searching strategies of top-down and bottom-up which adopted one after another. Pruning lessens communication traffic.

The searching strategy of top-down is described as follow.

Confirming the largest size *k* of itemsets in *F'*.

2) Collecting global frequency of all local frequent *k*-itemsets in *F'* from other nodes P_i .

3) Judging all local frequent *k*-itemsets in *F'*, if local frequent *k*-itemsets *Q* are not global frequent itemsets, then *Q* are deleted from *F'*, else turn to *4)*.

4) Adding *Q* and any nonempty subset of *Q* to global frequent itemsets *F* according to theorem 3 . Deleting *Q* and any nonempty subset of *Q* from *F'* .

The searching strategy of bottom-up is described as follow.

1) Collecting the global frequency of all local frequent 2-itemsets in *F'* from other nodes P_i .

2) Judging all local frequent 2-itemsets in *F'*, if local frequent 2-itemsets *R* are global frequent itemsets, then *R* are Added to global frequent itemsets *F* and *R* are deleted from *F'* , else turn to *3)*.

3) Deleting *R* and any superset of *R* from *F'* according to Corollary 2.

Each node adopts FP-growth algorithm to compute local frequent itemsets in MGFI. Adopting FP-tree structure and mapreduce, FP-growth algorithm greatly reduces database scanning times and runtime compared with Apriori-like algorithm.

3.2 Description of MGFI Algorithm

The pseudocode of MGFI is described as follows.

```
Alogrithm  MGFI
Input: The local transaction database $DB_i$ which has $M_i$ tuples and $M=\sum_{i=1}^{n} M_i$ , n nodes $P_i(i=1,2,\dots n)$, the center
node $P_0$, the minimum support threshold min_sup.
Output: The global frequent itemsets F.
Methods: According to the following steps.
step1. /*each node adopts FP-growth algorithm and mapreduce to produce local frequent itemsets*/
for(i=1;i<=n;i++) /*gaining global frequent items by mapreduce*/
{Scanning $DB_i$ once;
computing local frequency of local items $E_i$ ;
$P_i$ sends $E_i$ and local frequency of $E_i$ to $P_0$;
 }
$P_0$ collects  global frequent items E from $E_i$;
E is sorted in the order of descending support count;
$P_0$ sends E to other nodes $P_i$;   /*transmit global frequent items to other nodes $P_i$ */
for(i=1;i<=n;i++)
{creating the $FP\text{-}tree^i$; /*$FP\text{-}tree^i$ represent FP-tree of $DB_i$ */
 $F_i$ =FP-growth($FP\text{-}tree^i$, null);
}
```

```
Step2./* $P_0$ gets the union of all local frequent itemsets and prunes*/
for($i$=1;$i$<=$n$;$i$++)
$P_i$ sends $F_i$ to $P_0$;    /* $F_i$ represent local frequent itemsets of $P_i$ */
$P_0$ combines $F_i$ and produces $F'$; /* $F'=\bigcup_{i=1}^{n} F_i$ */
Pruning $F'$ according to the searching strategy of top-down;
Pruning $F'$ according to the searching strategy of bottom-up;
/*The searching strategies of top-down and bottom-up are described in section 3.1 */
$P_0$ broadcasts the remain of $F'$;
Step3./*computing global frequency of itemsets by mapreduce*/
for($i$=1;$i$<=$n$;$i$++)
{ for each items $d \in$ the remain of $F'$
$P_i$ sends $d.si$ to $P_0$; /*computing $d.si$ aiming at $FP\text{-}tree^i$ */
}
for each items $d \in$ the remain of $F'$
 $d.s=\sum_{i=1}^{n} d.si$;   /* $d.s$ represents global frequency of itemsets $d$ */
step4./*getting global frequent itemsets by mapreduce*/
for each items $d \in$ the remain of $F'$
if ($d.s$>=$min\_sup*M$)
$F=F \cup d$;
```

4. COMPARISON EXPERIMENTS OF MGFI

This paper compares MGFI with classical distributed algorithm CD and FDM. All tests are performed on 100M LAN, 6 PC as distributed nodes and 1 server as center node. The experimental data comes from the sales data in July 2012 of a supermarket.

Comparison experiment: It is a way of changing the minimum support threshold while adopting fixed number of nodes. MGFI compares with CD and FDM in terms of communication traffic and runtime. The results are reported in Figure 1. and Figure 2.

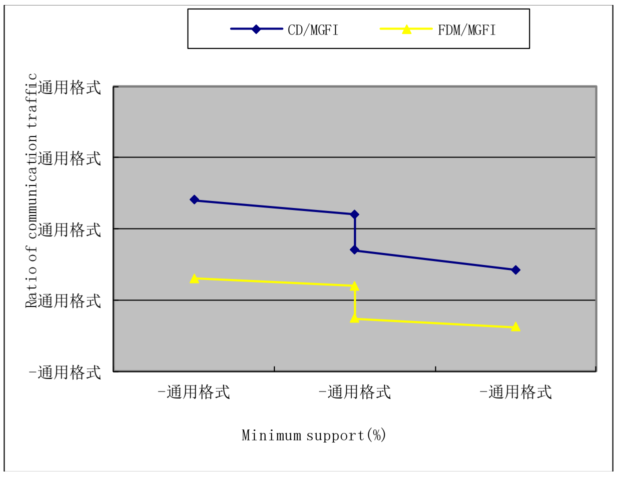

Figure 1. Comparison of communication traffic

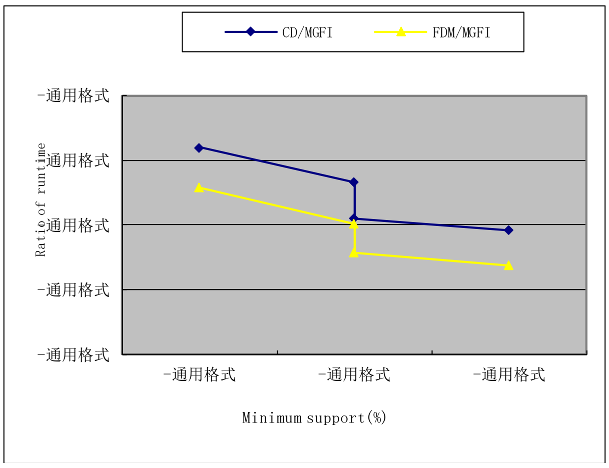

Figure 2. Comparison of runtime

The comparison experiment results indicate that under the same minimum support threshold, communication traffic and runtime of MGFI decreases while comparing with CD and FDM.

5. Example of MGFI Algorithm

The database *DB*, as show in TABLE I. *Min_sup* is the minimum support threshold, *min_sup*=0.4.

TABLE I. database DB

database	ID	Transaction
DB	100	a, b, c, k, m, f, e, l, p
	101	c, k, b, m, o, q
	102	a, b, c, d

According *min_sup*=0.4, all frequent items can be got. All frequent items are sorted in the order of descending support count. As shown in TABLE II.

TABLE II. The frequent items and support count

Frequent Items	Support count
c	3
a	2
k	2
b	3
m	2

All frequent items *E*={*c*, *b*, *a*, *m*, *k*},the FP-tree is constructed according to *E*, as shown in figure 1. According to Theorem 2, If item x is not frequent item, and $\{x\} \subseteq X$, then itemsets X must not be frequent itemsets. Hence the FP-tree only contains frequent items.

The frequent itemsets are computed by FP-growth algorithm and *FP-tree*. *F*={{*c*, *b*, *a*}, { *c*, *b*, *m*, *k*}, {*c*, *b*}, {*c*, *a*},{*b*, *a*}, { *c*, *b*, *m* }, { *c*, *b*, *k* },{ *c*, *m*, *k* }, { *b*, *m*, *k* }, {*c*, *m*} , {*b*, *m*}, {*c*, *k*}, {*b*, *k*}, {*m*, *k*}}.

6. CONCLUSION

MGFI pruned by the searching strategies of top-down and bottom-up. The global frequent itemsets are gained by mapreduce. Theoretical analysis and experimental results suggest that MGFI is fast and effective.

ACKNOWLEDGMENTS

This research is supported by the research fund for humanities and social sciences of the ministry of education under grant No.19XJA910001 and the postgraduate innovation fund project of chongqing university of technology under grant No.clgycx 20203114.

REFERENCES

[1] Chen ZB, Han H, Wang JX. Data Warehouse and Data Mining[M].Beijing: Tsinghua University Press, 2009.

[2] Agrawal R, Shafer JC. Parallel mining of association rules[C]. IEEE Transaction on Knowledge and Data Engineering, 1996, 962-969.

[3] Cheung DW, Han JW, Ng WT, Tu YJ. A fast distributed algorithm for mining association rules[C]. In: Proceedings of IEEE 4th International Conference on Management of Data, Miami Beach, Florida,1996, 31-34.

[4] Han JW, Pei J, Yin Y. Mining frequent patterns without Candidate Generation[C]. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, Texas, United States,2000,1-12.

[5]He B, Yue W, Yang W and Yuan C. Fast Algorithm for mining association rules Based on Distributed Database [C]. Rough Sets and Knowledge Technology, Chongqing, 2006, 415-420.