The Algorithm of Mining Frequent Itemsets Based on MapReduce

Bo He

Abstract Cloud computing is large scale and highly scalable. The data mining based on cloud computing was a very important field. The paper proposed the algorithm of mining frequent itemsets based on mapReduce, namely MFIM algorithm. MFIM algorithm distributed data according horizontal projection method. MFIM algorithm made nodes compute local frequent itemsets with by FP-tree and mapReduce, then the center node exchanged data with other nodes and combined; finally, global frequent itemsets were gained by mapReduce. Theoretical analysis and experimental results suggest that MFIM algorithm is fast and effective.

Keywords FP-tree · MapReduce · Frequent itemsets · Cloud computing

1 Introduction

The key for mining association rules is finding frequent itemsets [1]. There are various serial algorithms for mining association rules, such as Apriori [2]. However, the database for mining association rules is generally large, traditional serial algorithms cost much time. In order to improve efficiency, some parallel mining algorithms were proposed, which include PDM [3], CD [4], FDM [5]. Most of them

B. He (✉)
School of Computer Science and Engineering, ChongQing University of Technology, Chongqing 400054, China
e-mail: heboswnu@sina.com

B. He
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

B. He
Shenzhen Key Laboratory of High-Performance Data Mining, Shenzhen 518055, China

S. Patnaik and X. Li (eds.), *Proceedings of International Conference on Soft Computing Techniques and Engineering Application*, Advances in Intelligent Systems and Computing 250, DOI: 10.1007/978-81-322-1695-7_62,

divide global transaction database into equal n fractions according to horizontal method. In addition, most parallel mining algorithms adopt Apriori-like algorithm, so that a lot of candidate itemsets are generated and database is scanned frequently. Cloud computing is large scale and highly scalable. The data mining based on cloud computing was a very important field. Then, the paper proposed the algorithm of mining frequent itemsets based on mapReduce, namely MFIM algorithm.

2 Related Description

The global transaction database is DB, and 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 ($i = 1, 2, \ldots, n$) 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 gained by mapReduce.

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

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

Definition 3 For itemsets X, if $X.si \geq \text{min\_sup}*M_i$ ($i = 1, 2, \ldots, n$), then X is 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 $X.s \geq \text{min\_sup}*M$, then X is defined as global frequent itemsets, symbolized as F. If $|X| = k$, then X is symbolized as F_k.

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

3 MFIM Algorithm

MFIM distributes data according to horizontal projection method that divides M tuples in global transaction database into M_1, M_2, …, M_n ($\sum_{i=1}^{n} M_i = M$). The aggregation including M_i tuples in the ith node represents $\left\{T_i^j | T_i^j = O_q \text{ and } q = n \times (j-1) + i\right\}$, T_i^j represents the jth tuple of the ith node, O_q represents the

qth tuple of global transaction database DB. DB is divided into n local databases $DB_1, DB_2, \ldots, DB_n$ as large as $\lfloor \frac{M}{n} \rfloor$, namely $DB = \bigcup_{i=1}^{n} DB_i$. Because, DB_i gets the tuples of DB via regular separation distance, and global transaction database is divided into n local database evenly, MFIM reduces data deviation.

MFIM sets one node P_0 as the center node, other nodes P_i 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 strategy of top–down. P_0 sends the remaining of F' to other nodes. For local frequent itemsets $d \in$ the remaining 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 strategy of top–down. Pruning lessens communication traffic.

The strategy of top–down is described as follow.

(1) 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 and Deleting Q and any nonempty subset of Q from F'.

The pseudo code of MFIM is described as follows:

Algorithm MFIM Input: The local transaction database DB_i that has M_i tuples and $M = \sum_{i=1}^{n} M_i$, n nodes P_i ($i = 1, 2, \ldots, n$), the center node P_0, the minimum support threshold min_sup.

Output: The global frequent itemsets F.

Methods: According to the following steps:

Step 1: /* distributing data according to horizontal projection method*/

```
for(q=1;q<=M;q++)
{ if (q mod n= =i)
  { if (i= =0)
      i=n;  /*the q^th tuple is in the n^th node*/
P_0 makes O_q insert to DB_i ;  /*O_q represents the q^th tuple in
DB */
    }
}
for(i=1;i<=n;i++)
P_0 transmits DB_i to P_i ;
```

Step 2: /*each node adopts FP-growth algorithm to produce local frequent itemsets by FP-tree and mapReduce*/

```
for(i=1;i<=n;i++) /*gaining global frequent items*/
 {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-tree^i; /*FP-tree^i represent FP-tree of DB_i
*/
          F_i =FP-growth(FP-tree^i, null);
/* node adopts FP-growth algorithm to produce local frequent
itemsets F_i  */
}
```

Step 3: /* 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 */
                                                  F' = ⋃_{i=1}^{n} F_i
P_0 combines F_i and produces F';  /*          */
Pruning F' according to the strategy of top-down;
P_0 broadcasts the remain of F';
```

Step 4: /*computing global frequency of itemsets*/

```
for(i=1;i<=n;i++)
{ for each items d∈the remain of F'
    P_i sends d.si to P_0; /*computing d.si aiming at FP-tree^i
*/
}
for each items d∈the remain of F'
                  ∑_{i=1}^{n} d.si
 d.s=             ;   /* d.s represents global frequency of
itemsets d */
```

Step 5: /*getting global frequent itemsets by mapReduce*/

```
for each items d∈the remain of F'
if (d.s>=min_sup*M)
F=F∪ d;
```

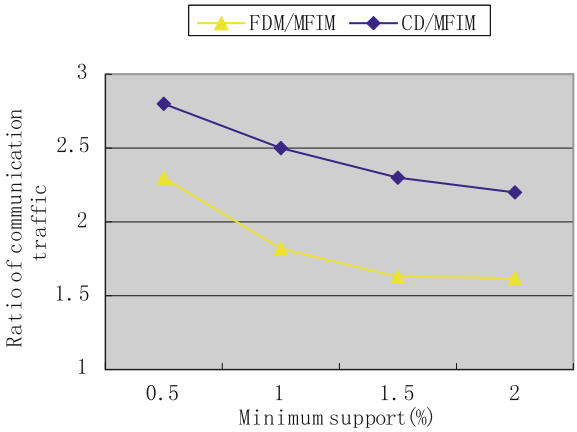

Fig. 1 Comparison of communication traffic

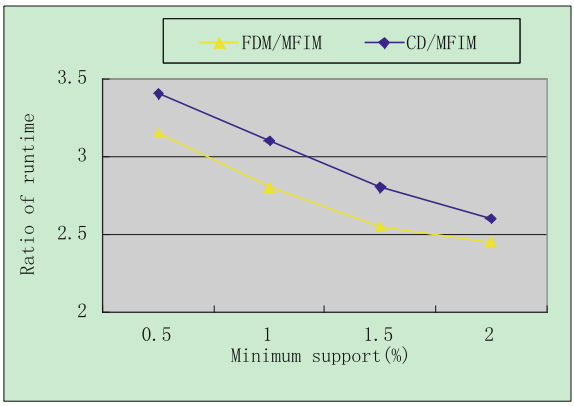

Fig. 2 Comparison of runtime

4 Experiments of MFIM

This paper compares MFIM to classical parallel algorithm CD and FDM, takes advantage of VC++6.0 to realize CD and FDM. MFIM compares to CD and FDM in terms of communication traffic and runtime. In the experiments, the number of tested nodes is five except center node. The experimental data comes from the sales data in June 2012 from a supermarket. The results are reported in Figs. 1 and 2.

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

5 Conclusions

MFIM makes nodes calculate local frequent itemsets independently by FP-growth algorithm and mapReduce, then the center node exchanges data with other nodes and combines by the strategy of top–down. It can promote highly the efficiency of data mining.

Acknowledgments This research is supported by the fundamental and advanced research projects of Chongqing under grant No. CSTC2013JCYJA40039 and the science and technology research projects of Chongqing Board of Education under grant No. KJ130825. This research is also supported by the Nanjing university state key laboratory for novel Software technology fund under grant No. KFKT2013B23 and the Shenzhen key laboratory for high-performance data mining with Shenzhen new industry development fund under grant No. CXB201005250021A.

References

1. Chen, Z.B., Han, H., Wang, J.X.: Data Warehouse and Data Mining. Tsinghua University Press, Beijing (2009)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining frequent itemsets. In: Proceedings of the 20th International Conference Very Large Data Base, Santiago, pp. 487–499 (1994)
3. Park, J.S., Chen, M.S., Yu, P.S.: Efficient distributed data mining for frequent itemsets. In: Proceedings of the 4th International Conference on Information and Knowledge Management, Baltimore, pp. 31–36 (1995)
4. Agrawal, R., Shafer, J.C.: Distributed mining of frequent itemsets. IEEE Trans. Knowl. Data Eng. **8**(6), 962–969 (1996)
5. Cheung, D.W., Han, J.W., Ng, W.T., Tu, Y.J.: A fast distributed algorithm for mining association rules. In: Proceedings of IEEE 4th International Conference on Management of Data, Miami Beach, pp. 31–34 (1996)
6. He, B.: Fast mining of global maximum frequent itemsets in distributed database. Control Decis. **26**(8), 1214–1218 (2011). (in Chinese with English abstract)