MapReduce-based Balanced Mining for Closed Frequent Itemset

Guang-Peng Chen[1], Yu-Bin Yang[1], Yao Zhang[2]
1. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
2. Jinling College, Nanjing University, Nanjing 210089, China
E-mail: yangyubin@nju.edu.cn

***Abstract*—Mining closed frequent itemset (CFI) plays an essential role in many real-world data mining applications. With the emergence of abundant large-scale data sets, it now turns to be a significant and challenging issue to mine CFI concurrently. This paper proposes a parallel balanced mining algorithm for CFI based on the MapReduce platform. The proposed algorithm adopts Greedy strategy to group items aiming to balance the computation burdens among all parallel tasks, which is consisted of three main steps: (1) *Parallel Counting*, (2) *Global Construction of Frequent List (F_list) and Group Map (G_map)*, (3) *Parallel Mining for Closed Frequent Itemset*. Experimental results validate the method and show its effectiveness as satisfied speedup and scalability are both achieved in large-scale CFI mining tasks.**

Keywords-component; Closed frequent itemset; MapReduce; Data mining; Hadoop; Cloud computing

I. INTRODUCTION

Mining Closed Frequent Itemset (CFI) from transaction database is a fundamental task for a mass of data mining applications. A number of CFI-Mining algorithms [1,2,3] have already been successfully proposed, in which *AFOPT-closed* is a variation of *AFOP-Tree* algorithm [2, 3] used to mine closed frequent itemset by simply scanning the original database twice. In the first scan, all frequent items in $database$ are counted and sorted in descending order by their frequency, denoted as F_list. The second scan is performed to construct an *AFOP-Tree*, on which the mining task is then completed. However, it needs to use computing resources intensively so that it can only handle small data sets, rather than large-scale ones, smoothly.

Hadoop, a prevalent cloud computing infrastructure, is best known for MapReduce and its distributed file System. Many parallel data mining algorithms based on MapReduce were developed to handle large-scale datasets. Particularly, Li et al. [4] proposed a MapReduce-based PFP algorithm to mine frequent itemset. In our previous work [5], an algorithm MCFI was also proposed to mine CFI based on Hadoop. But it fails to address the issue of balancing computation burdens among parallel tasks. To address the above issues, this paper presents a balanced mining algorithm for CFI based on MapReduce.

II. THE MAPREDUCE-BASED BALANCED MINING ALGORITHM FOR CLOSED FREQUENT ITEMSET

In this section, we present a framework of balanced mining for CFI based on MapReduce as shown in Fig. 1. The input of **Step 1** and **Step 2** are both the original database, which is split by Hadoop automatically. Each part of the framework is described in detail as follows.

A. Parallel Counting

This MapReduce pass scans the database once and counts the frequency of each item [4].

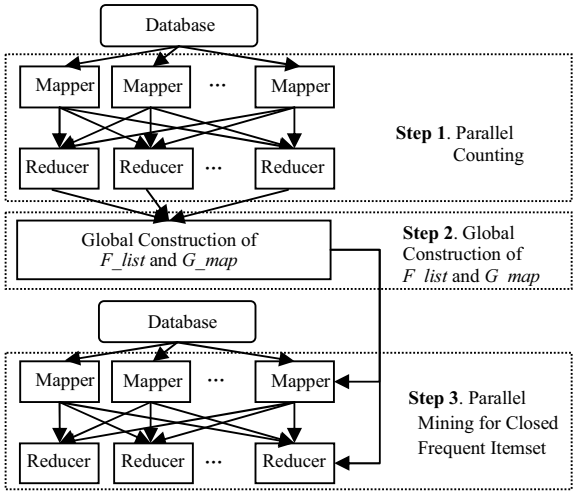


Figure 1. The framework of balanced mining algorithm.

B. Constructing Global F_list and Grouping Items

In this step, items are sorted according to their frequency in descending order. Infrequent items, of which frequency is less than *min_sup* are removed and the *Global F_list* is constructed by using the remained ones. Then, as shown in the following **Algorithm 1**, we group items in *F_list* under greedy strategy. The basic assumption of this method is that the computation burden is proportional to the number of transactions.

Algorithm 1 Constructing *G_map*

```
1: Procedure: constructGList (F_List, NG, S)
2: { ave = S/ NG ;
3:   Pretreat F_list, ensuring that adjacent items with the
4:   same support are allocated in the same group;
5:   for( G_id=0 to NG)
6:    { Calculate k according to the following Eq. (1);
7:      Put those k items into G_map with G_id;
8:      Delete those k items from F_list;
9:    }
10:   return G_map;
11: }
```

NG is the number of groups we expect to generate, *S* is the sum of the support values of all items in *F_list,* and *ave* is the average support values of *NG* groups.

$$k = \arg \min_{0 \le k < F\_list.size()} \left| \sum_{i=0}^{k} (F\_list[i].\sup) - ave \right|. \quad (1)$$

C. Parallel Mining for Closed Frequent Itemset

This step concurrently mines closed frequent itemset on Hadoop platform, and is the key step of our algorithm. The

978-0-7695-4752-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICWS.2012.19

pseudo codes are presented in **Algorithm 2**.

Algorithm 2 Parallel Mining for Closed Frequent Itemset

```
1: Procedure:Mapper( key,  value = transaction )
2:  { sort value by F_list;
3:    a[ ] = Split(value);
4:    last_gid= null;
5:    for ( i= a.length -1 to 0 )
6:     { if(last_gid  !=  a[i]. G_id)
7:          output(a[i]. G_id, Text(a[0]+a[1]+ ⋯+a[i]) );
8:        last_gid= a[i]. G_id ;
9:     }
10:  }
11: Procedure:Reducer(current_G_id ,  Iterable values )
12:  { r = constructAFOP-Tree(values) ;
13:    closedFITree =  AFOPT-closed ( r, min_sup) ;
14:    for( cFItems : closedFITree)
15: {if(cFItems.lastItem( ). G_id == current_G_id )
16:     output(current_G_id,  cFItems + ' Support values' ) );
17:   }
18:}
```

III. Experimental Results

We carried out experiments on a 6-node cluster running on Hadoop 0.21.0, one of which works as master and the others work as slaves. The dataset we adopted is "webdocs", almost the largest open dataset (1.37GB) in related fields. Different *min_sup* values are adopted in the experiments. According to the scale of our cluster, the number of group is set as 15.

Fig. 2 shows the speedup values of our algorithm for mining "webdocs". We can also see that up to 4 slaves, the speedup is very close to the ideal speedup. When the number of slaves is 5, the speedup grows more slowly than before. That phenomenon is mainly caused by the well-known Amdahl's law [4]. When the number of slaves reaches a certain level at which each slave's computation burden is very small, it will not be more effective than before to continue expanding the size of the cluster. However, if the size of dataset increases, adding more slaves into the cluster can still achieve higher speedup value. Therefore, we may infer that our algorithm is scalable for large-scale CFI Mining tasks.

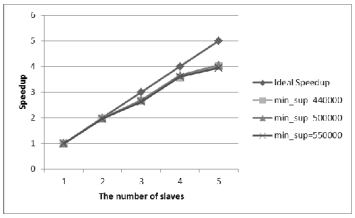
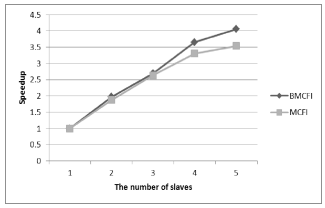
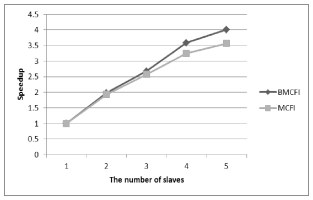


Figure 2. The speedup of the proposed algorithm on "webdocs"dataset.

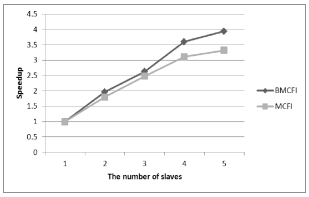
We compared the performance of the proposed algorithm (BMCFI) to that of MCFI mentioned in Section I, both ran in the same experiment setting. Fig. 3 shows the speedup values of them with different *min_sup* values on the "webdocs" dataset. At the beginning, their speedup values are almost same. As the number of slave nodes increases, the proposed algorithm performs much better than MCFI. This improvement holds for all different *min_sup* values, as shown in Fig. 3. The results prove that our grouping strategy is able to balance the computation burdens of parallel tasks efficiently.



3-(a) *min_sup* =400,000



3-(b) *min_sup* =440,000



3-(c) *min_sup* =550,000

Figure 3. Performance comparisons between the proposed algorithm and MCFI.

IV. Conclusion

In this paper, we present a balanced mining algorithm for closed frequent itemset based on MapReduce. Greedy grouping items strategy is adopted to balance computation burdens of all parallel tasks so as to improve parallelization performance. Experimental results are illustrated to demonstrate the simplicities and effectiveness of the proposed framework.

Acknowledgment

This work is supported by the Program for New Century Excellent Talents of MOE China (Grant No. NCET-11-0213), the International Science and Technology Cooperation Program of China (Grant No. 2010DFA11030), National 973 Program of China (Grant No. 2010CB327903), the Natural Science Foundation of China (Grant Nos. 61035003, 61021062), and the Natural Science Foundation of Jiangsu, China (Grant No.BK2011005, BK2010054).

References

[1] S. Shankar and T. Purusothaman, "Utility Sentient Frequent Itemset Mining and Association Rule Mining: A Literature Survey and Comparative Study," International Journal of Soft Computing Applications, Issue 4, pp.81-95, 2009.

[2] Guimei Liu, Hongjun Lu, Jeffrey Xu Yu, Wang Wei, and Xiangye Xiao. AFOPT: An Efficient Implementation of Pattern Growth Approach. Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementation, 2003.

[3] G. Liu, H. Lu, Y. Xu, and J. X. Yu. Ascending frequency ordered prefixtree: Efficient mining of frequent patterns. Proc. of 8th Int. Conf. Database Systems for Advanced Applications. Kyoto, Japan, 2003: 65–72.

[4] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. Pfp: parallel fpgrowth for query recommendation. Proc. of the 2008 ACM conference on Recommender systems. Lausanne, Switzerland, 2008: 107-114.

[5] CHEN Guangpeng, YANG Yubin, GAO Yang, and SHANG Lin, "Mining closed frequent itemset based on Mapreduce", CCDM2011. Guang Zhou, China, 2011. (in Chinese)