2013 8th International Conference on Communications and Networking in China (CHINACOM)

A Parallel Algorithm of Association Rules Based on Cloud Computing

Wang Yong[1], Zhang Zhe[1], Wang Fang[2]
[1]Department of Computer Science and Engineering, [2]School of Information and Communication
Guilin University of Electronic Technology
Guilin, China
ywang@guet.edu.cn, 289640431@qq.com

***Abstract*—In view of the traditional parallel FP-growth algorithm (PFP)that suffers from two major limitations, namely, multiple database scans requirement (i.e., high I/O cost) and high inter-processor communications cost, therefore we design and implement a parallel association rules mining method based on cloud computing. The algorithm adopts the separation strategy to simply visit a local database only once, thus, the inter-processor communication I/O overhead is reduced. What's more, the MapReduce model is used to solve the problem of huge amounts of data mining, as well as the calculated execution taking place in the local data storage node, which can avoid large amounts of data on the network transmission and reduce the communication overhead. By using ordinary PC structures, Hadoop cluster experimental results verify that the proposed algorithm based on cloud computing offers higher efficiency and has a good speedup.**

Keywords-FP-growth; Parallel Computing;Cloud Computing; MapReduce; Association Rules

I. INTRODUCTION

Association rule mining is an important branch of data mining as it analyzes the scale of transaction records in the collection database and finds the interesting relationships among the massive amounts of data. Among these, frequent pattern (FP) mining has become a very important part[2-5].The candidate generation-and-test methodology, called the Apriori technique, was the first technique to compute frequent patterns[1].However, this approach generates a large number of candidate datasets and repetitively scans the database to verify whether it is frequent or not. Han et al. [2] proposed a novel data structure and method of mining frequent patterns, the Frequent Pattern (FP) tree data structure which only stores compressed necessary information for mining. Unlike the Apriori technique, the FP-tree only scans a database twice. Meanwhile, for mining the variable length of frequent itemsets, it's effective and scalable and it is a highly efficient algorithm addressing the limitations of the Apriori technique.

However, in practice, the data mining algorithms often need to deal with distributed and large databases. Therefore, large-scale parallel and distributed FP mining algorithms have aroused much research [2-4]. Through the study of FP-growth algorithm, a large number of parallel and distributed FP-growth(PFP-growth) mining algorithms have been proposed, such as, FPForest[3]，PFP-tree[4]，LFP-tree[3]，and MLPT[5]. The FP-tree-based algorithms are to divide the database and then to distribute each part of the database to nodes or processors for mining to distribute the computation loading. During the mining process, the nodes will exchange required transactions with each other. Although many algorithms have been proposed, the execution efficiency of frequent pattern mining is still a challenge for the researchers due to the data explosion. Simultaneously, all the algorithms which are proposed previously need scan the local transaction database twice.

None of those PFP-growth algorithms have completely solved the problem of computing performance limit, in spite of optimization of communication costs and task scheduling [2-6].Meanwhile, with the increase of application and speed of the network, the type and quantity of frequent itemsets are growing. However, because of the limitations of computing resource in face of large sample data sets, the computational complexity and the cost of communication will grow exponentially, that greatly restricts the application of the FP-growth algorithm. Therefore, in this paper, we propose the CPFP data mining method which is novel but similar to PFP-growth based on the MapReduce framework in the Hadoop platform. It also can effectively solve the bottleneck problems of traditional PFP-growth algorithm by means of powerful parallel processing of cloud computing, and good ability of load balancing. Through empirical evaluations on various simulation conditions, the proposed algorithms can not only ensure correctness of the data mining results, but also inordinately improve the efficiency of data mining to speed up the whole process.

The rest of the paper is organized as follows. Section Ⅱ introduces the improvement of the traditional PFP-growth algorithms, discusses the structure and procedures of CPFP. Section Ⅲ, experimental results will be revealed. Finally, Section Ⅳ concludes the paper.

II. THE IMPROVEMENT OF PFP-GROWTH ALGORITHMS BASED ON CLOUD COMPUTING

In this paper, in the open source of Hadoop platform, MapReduce model was chosen to shunt data from the database, and then to build the local FP-tree in the connection nodes which effectively improve the efficiency of data mining and reduce the communication overheads as well as offer good speedup.

978-1-4799-1406-7

A. Cloud Computing Platform Overall Framework Design

The design of Web-based cloud computing data mining system is presented in Figure 1, nodes are divided into three categories: the master node (Master), which is responsible for the scheduling and coordination between the computing nodes; the storage node for storing the data mining algorithms; And ServiceNode, for the storage to store sub good block of XML files and execution of tasks assigned by the Master, and finally providing the result to the Master [7].

Fig 1 Web-based cloud computing data mining system

In the cloud computing environments the data privacy is an important issue. Since the clouds are distributed physically and each cloud node provides its own computation ability, the trusty of the nodes cannot be preserved [12].In order to preserve the data privacy, a "Master-slave" model which consists of some clouds may be designed. The Master cloud includes a kernel node, abbreviated as KN-node, which can only access the database. Other clouds are the ones that consist of two parts, of which one is computing node, and the other is similar to a router, named CONN-node, which every cloud has the only one. Meanwhile, the CONN-node should maintain a table to record the status of the nodes of its cloud, such as the node's ID or its availability etc, so that can connect with other clouds effectively. And it can also duplicate the data via internet if a node needs data from KN-node, then the node can download the data from CONN-node via intranet. By using this transmission policy, the network latency can be minimized.

B. the Improvement of PFP-growth Algorithms Based on Cloud Computing

1) Introduction of PFP-growth Algorithms

In [7], SK Tanbeer et al put forward a PFP-growth method of frequent itemsets mining, which has been extended to mine maximal frequent itemsets and frequent closed itemsets one by one. The algorithm creates FP-tree with a header table based on frequent items' support count by scanning the data, then the header data items is arranged in descending order. After getting a new table, we make use of Branch sorting method [9], abbreviated as BSM. Firstly, it peels the branches of the root node. Secondly, those branches are stored in different temporary array. Then according to the sequence of data items, we rearrange the temporary array item in order to reconstruct the FP-tree. At last, comparing the new header data items with the minimum support (min_sup), the FP-tree is restructured by deleting the itemsets of which the support count is less than min_sup. And then the algorithm recursively build prefix tree and conditions for finding frequent itemsets.

Fig 2 Proposed architecture for frequent pattern mining

2) The Improvement of PFP-growth Algorithms Based on Cloud Computing

Figure 3 architecture of PFP-growth algorithm model

In this paper, we use the open source of Hadoop platform which is designed for multi-tier architecture of PFP-growth algorithm model, as shown in Figure 3.At first, the initial data set is uploaded to HDFS (Hadoop Distributed File System) in the pre-treatment which is divided into a plurality of subsets of data, and then the block of data is transmitted to the CONN-nodes. Meanwhile KN-node in its own cloud by the MapReduce model, finds out the global support count of the data item in parallel, and the global data items are in descending order, denoted by GHT. Secondly, CONN-node scans the database and establishes the local FP-tree. Then KN-node broadcasts GHT to CONN-nodes. According to BSM [7], the FP-tree based on GHT may be reconstructed. We could mine local frequent itemsets through the establishment of conditions for tree and output frequent itemsets to KN-node. Finally we find out the global frequent itemsets.

3) The Improvement of PFP-growth Algorithms Based on Cloud Computing

A cloud-based platform PFP-growth algorithm is designed combining with the MapReduce programming model for cloud computing platform. Such algorithm is described as follows:

- KN-node reads the data from database. It reads support count of data items through the MapReduce model in its own cloud. At the same time, in the process of vast amounts of information processing, the Mapper operation tends to generate a large number of writing operations, but sometimes the reducer tends to update in an untimely manner. Hadoop platform supports the combiner. The combiner can be used to accumulate the output of on the Mapper function of the same data item, so that it can reduce the communication and I / O overhead [10].Meanwhile, KN-node reads data to the HDFS pre-processing in parallel, and divides the dataset into a plurality of data blocks. Then these blocks can be transmitted to CONN-nodes through the network. The pseudo code is as follows:

```
Mapper (LongWritable offset,Text value = transaction)
{   for ( each item Ai in value)
        EmitIntermediate (itemset,1);
}
Combiner （Text key, Iterable <IntWritable> values )
{    int supi=0;
     for each 1 in list:
       supi+=1;
     EmitIntermediate (Ai, supi);
     //supi denoted as sum of the same itemset
}
Reducer (Text key,Iterable <IntWritable>  values)
{    int sum = 0;
     for ( IntWritable value: values)
     {    sum = sum + supi;
      }
      EmitIntermediate (key, sum);
  }
```

- According to the support count of each data item in the previous step, we can easily get the header table of global data itemset, and put them in descending order of all the support count of itemset. Then we get a new header table of data items, denoted by GHT;
- CONN-nodes receive a data block, and establish the local FP-tree in parallel. Then by means of STEP 2, CONN-nodes can receive GHT which is transmitted by the network. And CONN-nodes make use of BSM [7] that restructure all branches, one-by-one, from the root of FP-tree. The Mapper function read the branches of FP-tree to a temporary array. While restructuring a branch, BSM sorts each path in the branch according to GHT order by removing it from the tree, sorting it into a temporary array, and again inserting it into the tree in sorted order. The Reducer function receives all of transactions and constructs a new CP-tree (Closed pattern) which is the tree of the condition of key. Then, FP-growth algorithm is applied that recursively mine frequent closed itemsets in the tree. Those frequent closed itemsets we get is the local frequent closed itemsets. The pseudo-code is shown below.

```
Mapper (Long Writable offset,Text value = transaction)
{    string a[]= Split( value) ;
     a[] order value by GHT;
     for( i = a.length-1; i ＞＝ 0; i--)
          EmitIntermediate ( Text ( a[i]),Text ( a[0]+a[1]+…+a[n]) ) ;
}
Reducer (Text key,Iterable<Text> values)
{    r = constructCP-tree (key, values);
     FCItemsTree = CP-tree (r,  min_sup);
     //FC 为 frequentclosed
     for( FCItems : FCItemsTree)
     {
     EmitIntermediate (Text (key) , Text (FCItems +FCItems' support));
     }
}
```

- CONN-nodes send the excavated local frequent itemsets to KN-node, and Mapper will sort the order of itemset in values by their length from long to short, and then output. Reducer then added the itemsets received to the FP-tree. Note that the length of the current itemsets must be no more than the length of the beginner itemsets, therefore, Reducer would only add those closed itemsets to FP-Tree of discarded others. t Then only the last frequent itemsets of Reducer can be output. At this time, the frequent itemsets become the global frequent itemsets [11]. The pseudo-code is shown below.

```
Mapper (LongWritable offset, Text value =FCItems + FCItems' support)
{    itemset = value. FCItems;
     Sort the Order of itemset in values by their length from long to short;
```

```
        for ( item: itemset)
            EmitIntermediate (item, value);
}
Reducer (Text key,Iterable < Text> values)
{       FP-Tree = null;
        for ( itemset :values)
        {       if ( itemset is closed in FP-Tree)
                    insert itemset into FP-Tree;
        }
        for ( itemset : FP-Tree)
       {
                if ( key is the last item of this itemset)
                    EmitIntermediate (Text (key) , Text (itemset));
        }
}
```

III. EXPERIMENTAL RESULTS

In order to evaluate the performance of the algorithm, we use the three larger data sets, such as mushroom, connect, pumsb, in the field of mining frequent itemsets common test data sets. And for the same data set, we choose different min_sup in the experiment. Experiments mainly consist of three cloud system: the first cloud includes 1 kernel node and the other two clouds totally have 5 computing nodes. As KN-node doesn't take part in computing, a total of 5 nodes can be used for mining association rules. Hadoop cluster configuration is shown in Table 1.

Table 1 Hadoop cluster configuration

	CPU	RAM	Disk
KN-node	Pentium Dual Core	1GB	320G
Computing node	Pentium Quad Core	1GB	160G

As is seen from Figure 4, for the same data sets, the execution time of different min_sup are not the same, but the tendency of execution time is consistent. With the increase of datanodes, the overhead of communication is also growing, but for single datanode, the data processing efficiency of multi-nodes is still a great advantage. In Figure 4(a), since the data set is small, the execution time is increasing when the number of data nodes add, but in (b) and (c), for the connect data set, when min_sup is 40000 the result of execution time is optimal, so as to the pumsb data set when min_sup is 140000. Because as the number of the min_sup increases, the number of frequent closed itemsets gradually reduces, the amount of data which step 3 and step 4 need to be processed are reduced. Meanwhile,the proportion of time overhead used for parallel co mputing also decreases.

At the same time, we can also see from the figure, the optimal execution time of (c) is better than (b), (b) is better than (a). Because when the data set is small (size of data set: mushroom <connect <pumsb), the time used for computing in each node is less. With the increase of data note number, the proportion of time overhead used for parallel computing also decreases. However, for the large data set, the computing time of each node is inherently long. For this reason, the proportion of time overhead used for parallel computing is relatively lower. We can draw the conclusion that the execution time of this algorithm in the large scale data sets is better than that in the small scale data sets, which also shows that this parallel algorithm has more advantages in dealing with large scale data sets.

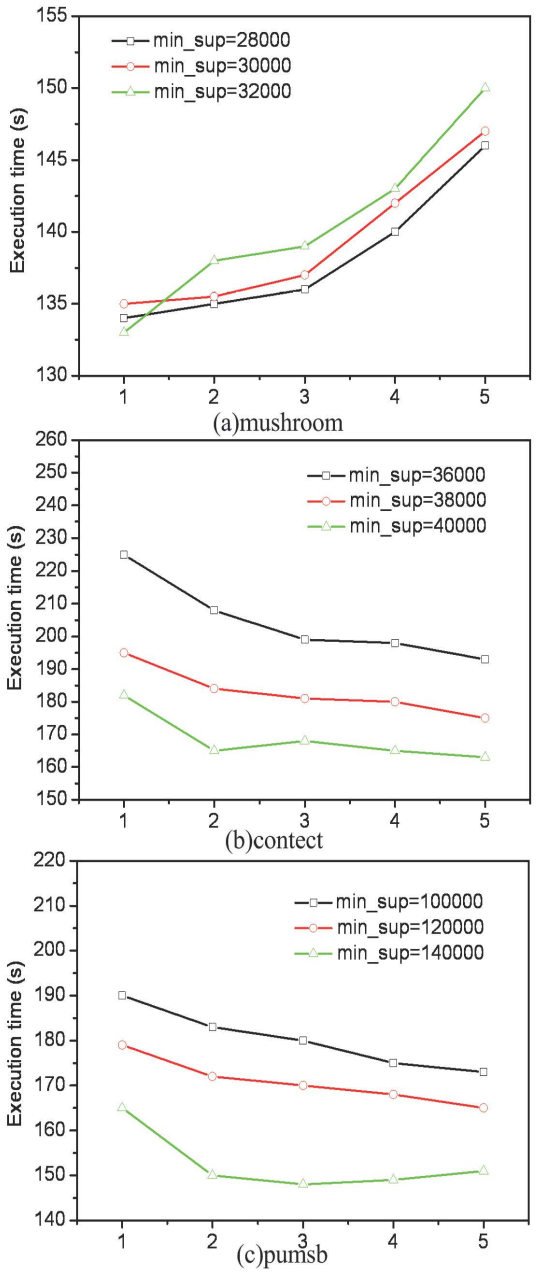

Figure. 4. Execution time in different experiments

Supported by the National Natural Science Foundation of China under Grant No.61163058

IV. CONCLUSIONS

In this paper, a parallel association rules mining method based on cloud computing is designed and implemented. The experiments show that the method is much more effective than conventional parallel data mining algorithms. At the same time, a framework for cloud computing platform system is proposed, which can make effective use of cloud nodes in a cloud environment to protect data privacy. Meanwhile, MapReduce model and established local FP-tree are used to the local frequent item sets. Though CPFP has good communication and I/O overhead, there is also something undesirable, especially for smaller data sets or nodes.

REFERENCES

[1] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*. Vol. 1215. 1994.

[2] Han, Jiawei, Jian Pei, and Yiwen Yin. "Mining frequent patterns without candidate generation." *ACM SIGMOD Record*. Vol. 29. No. 2. ACM, 2000.

[3] Javed, Asif, and Ashfaq Khokhar. "Frequent pattern mining on message passing multiprocessor systems." *Distributed and Parallel Databases* 16.3 (2004): 321-334.

[4] Yu, Kun-Ming, Jiayi Zhou, and Wei Chen Hsiao. "Load balancing approach parallel algorithm for frequent pattern mining." *Parallel Computing Technologies*. Springer Berlin Heidelberg, 2007. 623-631.

[5] Zaïane, Osmar R., Mohammad El-Hajj, and Paul Lu. "Fast parallel association rule mining without candidacy generation." *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001.

[6] Chen, Dehao, et al. "Tree partition based parallel frequent pattern mining on shared memory systems." *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. IEEE, 2006.

[7] Yu, Kun-Ming, and Jiayi Zhou. "Parallel TID-based frequent pattern mining algorithm on a PC Cluster and grid computing system." *Expert Systems with Applications* 37.3 (2010): 2486-2494.

[8] Miao Cheng. "Web Data M ining Based on Cloud-computing." COMPUTER SCIENCE38.B10 (2011): 146-149.

[9] Tanbeer, Syed Khairuzzaman, et al. "Efficient single-pass frequent pattern mining using a prefix-tree." *Information Sciences* 179.5 (2009): 559-583.

[10] Li, Lingjuan, and Min Zhang. "The strategy of mining association rule based on cloud computing." *Business Computing and Global Informatization (BCGIN), 2011 International Conference on*. IEEE, 2011.

[11] Guang-Peng Chen. et al. "Closed Frequent Itemset Mining Based on MapReduce." Pattern Recognition and Artificial Intelligence 25.2 (2012): 220-224.

[12] Lin, Kawuu W, and Yu-Chin Luo. "A fast parallel algorithm for discovering frequent patterns." Granular Computing, 2009, GRC'09. IEEE International Conference on.IEEE, 2009