Closed Sequential Pattern Mining Algorithm on Hadoop Platform

Wang Lifeng*

School of Information Engineering, Shandong Youth University of Political Science, Jinan 250000, Shandong, China

*LCorresponding author (E-mail: sdwanglf@163.com)

Liu Dong

School of Information Engineering, Shandong Youth University of Political Science, Jinan 250000, Shandong, China

Wang Zhiyong

School of Information Engineering, Shandong Youth University of Political Science, Jinan 250000, Shandong, China

Abstract

In order to improve the efficiency of the traditional sequential pattern mining algorithm when it is applied to process the large-scale datasets, this paper is to bring forward MR-BIDE on the basis of the research on BIDE algorithm and MapReduce, subdivides the mining tasks and makes it operated in parallel. Deployed on the platform of Hadoop, it realizes the function of the parallel computing in the distributed processing of the big datasets and solves the problems of the space complexity and the time complexity in serial computing in stand-alone environment. The results from an experiment shows that MR-BIDE enhances the efficiency of the sequential pattern mining, and especially in multi-node environment, it has higher capability to extend and speed-up ratio.

Key words: Hadoop Platform, Closed Sequential Pattern, MapReduce Programming Framework, Parallel Computing.

1. INTRODUCTION

With the increase in the size of data, the performance of a stand-alone computer is improved to its top limit so that the traditional mining algorithm in sequential mode cannot be well performed with it. The diversified, fine-grained and durable data, with the decrease of the minimum threshold of support configured, lead to a rapid increase in the quantity of the sequential modes, but not all of them are of any value. It reduces the efficiency of the mined results and undermines the expressive capability of these datasets. It is needed to reduce the size of the result sets as possible as we can, and as well, it is urgently needed to apply the parallel computing in mining because the bad operating efficiency increases the cost of network communications. Hadoop is an open-source platform developed by Apache to perform parallel computing and distributed processing and storage. And MapReduce is a programming framework of parallel computing to process the large-scale data on this platform. By studying Hadoop-related technology, many researchers at home or abroad brought forward new mining algorithms and their applications. Dean. J et al, based on MapReduce, put forward a method to simplify the large-scale datasets (Dean J and Ghemawat S, 2008); Huang Liqin, by improving the tradition mining algorithm, put forward Apriori based on MapReduce (Huan Liqin and Liu Yanhuang, 2011) and Yu Xiao brought about his GSP algorithm based on MapReduce as well (Yu Xiao, Ma Chuanxiang and Li Weiliang, 2015). In order to mine the information of data flows, Feng Zongwei, Zheng Suhang et al sequentially brought forward an information mining algorithm out of the mass data flows under the distributed environment. Modarakis N., Yan Yuliang et al, in order to have the large-scale datasets better processed, put forward a mining algorithm based on the sequential mode of Spark (Yan Yuliang, Dong Yihong and He Xianmang, 2015; Zhang Peng, Duan Lei and Qin Pan, 2017).

In 1995, R Agrawal et al were the first who put forward the concept of sequence mining (R. Agrawal and R. Srikant, 1995), which, gradually, grew into an important filed in data mining studies. It aims to find out knowledge in the data collected (Shie B E, Yu P S and Tseng V S, 2013; Yen S J and Lee Y S, 2013), and mined the valuable information out to help make right decisions (Li Z, Han J, Ding B and Kays R, 2012). The process is like this: according to the minimum degree of support, the sequence datasets containing sequential elements obtained are traversed to find out all the sets of elements which meet with the conditions. Traditionally, the researches about mining algorithms in sequential mode include traditional sequence mining, weighted or restrained sequence mining, incremental sequence mining and multi-dimensional sequence mining, etc. If the minimum threshold of support is not properly configured, the result sets will generate a large quantity of modes,
of which many are not valuable. Jianwei Han et al, after studying the closed item sets mining, put forward Clospan algorithm (X. F. Yan, J. Han and R. Afshar, 2000), BIDE algorithm (J. Y. Wang and J. W. Han, 2004) to process closed sequential mode. The closed item sets mined out with closed sequential mining algorithm, on one hand, have the same effect with the completed sets minded out with the sequential mining algorithm. It, at same time, efficiently avoids unnecessary generation of subsequences and produces more compact results. On the other hand, it increases the efficiency. Clospan is an algorithm based on Prefixspan. It mainly reduces the space for search, but the optional sequences generated occupy more RAM. BIDE bi-directionally reads whether there are extended events with prefixes or suffixes, not storing the optional sequences but producing directly the closed sequences.

The paper is to bring forward MR-BIDE on the basis of the research on BIDE algorithm and MapReduce, subdivides the mining tasks and makes it operated in parallel.

2. CLOSED SEQUENCE MINING ALGORITHM

2.1. Basic Concepts

Some concepts needing clarification are as follows:

Definition 1: Given two sequences \( s = \langle \alpha_1 \rangle \langle \alpha_2 \rangle \cdots \langle \alpha_m \rangle \) and \( s' = \langle \beta_1 \rangle \langle \beta_2 \rangle \cdots \langle \beta_n \rangle \), \( s' \) includes \( s \), i.e., if and only if there is a set of integers \( 1 \leq i_1 < i_2 < \cdots < i_n \leq m \) and \( \alpha_1 \subseteq \beta_{i_1}, \alpha_2 \subseteq \beta_{i_2}, \cdots, \alpha_n \subseteq \beta_{i_n} \) is true. \( s \) is called the subsequence of \( s' \), and \( s' \) is called the ultra sequence of \( s \), expressed as \( s \subseteq s' \); if the length of the two sequences is different, the \( s' \) is called true ultra sequence of \( s \), expressed as \( s \subset s' \).

Definition 2 Frequency sequence \( S \), if there is not an ultra sequence with the same degree of support, the sequence \( S \) is closed, and the frequency sequence is called as closed sequential pattern.

2.2. The Basic Idea of BIDE Algorithm

According to the sequence database \( D \), scan, forwards and backwards, the frequency \( k \)-sequence pseudo projection database. Two key judgments are contained in this algorithm: the pruning judgment and extended judgment. Given frequency \( n \) sequence \( S_f = \langle l_1 l_2 \cdots l_n \rangle \), \( l_n \) is the last \( l_n \) in sequence \( D \) which includes \( S_f \). If there exists \( I' \) which generates new sequence \( S_f' = \langle I' l_1 l_2 \cdots l_n \rangle \) earlier than \( I' \) and they have the same degree of support, \( S_f \) can be pruned; if \( I' \) generates new sequence later than \( l_n \) as \( S_f'' = \langle l_1 l_2 \cdots l_n I' \rangle \), and \( S_f'' \) and \( S_f \) have the same degree of support, \( S_f \) can be extended forwards, called as forward extending sequence, and otherwise, it is backwards extended, called as backward extending sequence. Therefore, the BIDE algorithm is a process in which an analysis about the Pro or Post sequential relation is made and the prefix-generating tree is built up.

3. REALIZATION OF THE MR-BIDE ALGORITHM

3.1. Key Technology of Hadoop Platform

Hadoop contains two key functional modules: HDFS and MapReduce.

HDFS realizes the function of the distributed processing and storage of the data files under the clustered environment. It follows the principal and subordinate mode. The namenode of it is called as the principal HDFS, responsible for maintaining indexes, files and managing data blocks. The data nodes are called as the subordinate HDFS, providing storage for data and services of reading or writing for clients.

MapReduce is a programming model for processing distributed big datasets. The fragmentation and submission of data and the assignment, operation, upgrading and computation of tasks are managed by the principal node JobTracker and operated by the client nodes TaskTracker. JobTracker manages the tasks and resources in the cluster; TaskTracker is responsible for assigning tasks of Map and Reduce. The operation is shown in figure 1.

(1) The dataset is uploaded and imported into HDFS, and then it is divided into many Splits and stored in the data nodes in the cluster.

(2) Stage of Executing Map Start the process of Hadoop MapReduce, call JobTracker, every sub-tree of sequence mining as a unit, create Map tasks, JobTracker calls TaskTracker and execute Map tasks.
(3) Shuffling and sorting Split the Key-value pairs(<key, value>) in between and sort by the value of the keys.

(4) Stage of Executing Reduce the Reduce process sums up the counts in the form of (key, list(value)), value indicating the degree of support.

3.2. Realization of the MR-BIDE Algorithm

While computing in serial mode in a stand-alone PC, the traditional BIDE algorithm consumes a lot of RAM resources in dealing with large-scale data sets. In order to solve the problems of space complexity and timing complexity resulted by serial sequence mining algorithm, we proposed MR-BIDE algorithm based on MapReduce. Taking advantage of the distributed storage capacity of Hadoop and the parallel computing mechanism of MapReduce, we extend the process of sequence mining to a cluster. It mines the local frequent items in parallel, thus constructing the global frequent closed sequence set. The implementation process is as follows:

(1) Master distributes Map task and Reduce task. The workers scan the frequency 1-sequence got from the database. A pruning step is added here; taking every frequency sequential pattern as a prefix, delete the corresponding sequence in frequency 1-sequence, and form the pseudo projection database.

Design of the Function Map

Input: The sequence data S and the minimum support min_sup. The key-value pairs are the item set corresponding to the first address offset and the transaction database.

Output: <key, value> pair, key is prefix, value is the number of appearance correspondingly;

sub_process BIDE(α, L, S | α)

scan S in parallel

for each task t in S

Map(t line offset, t)

F1 = find_1_t.itemsets; /*The first round of Map process.*/

for (k = 2; L_{k-1} \neq \emptyset; k++)

C_k = BIDE(α, F_{k-1}, S | α); /* Call the sub-process BIDE()*/

for each BIDE_F; /*Scan S, construct pseudo projection database, and compute the degree of support*/

splits ← α; /*Split into different sections according to the prefix, worker processes it in parallel*/

for each item α in t

out(α, 1) /*Generate key/value pairs and optional 1-item set.*/

if 1-sequence pattern satisfies the pruning conditions

delete S[α]; /*Add pruning step here, and discard the sequence corresponding to frequency 1-sequence and all the non-frequency items.*/

else

if there exists forward-extending or backward-extending event

Save and combine the prefixes into new sequence prefixes /* Write into HDFS, and optimize the
transmission of the results using Combiner.*/
else  Save and form closed 1-sequence.
end map

Programmed Function Reduce

Input:  out(α,1) /*key-value pairs output by map
Output: The frequency 1-item set corresponding to the prefixes in this branch.
Scans S(α) in parallel;
for each α in values
reduce( α, α.value)
sum=0
if(α.support≥min_sup&&F ==Null)
sum++  /*Sum up the number of the appearance of the sequence*/
out(α.value ,sum)    /*Form Frequency 1-item set*/
sum++  /*Sum up the number of the appearance of the sequence*/
Sum up the count of the new prefix sequences.
End reduce  /* The first round of MapReduce process finished*/

(2)Sort the set generated from the frequency 1-item set in process (1), and input them in the second round
map in turn, then generate optional 2-item set. Through the pruning judgment and extending judgment, input the
results, enter the reduce process and generate frequency 2-item set. Vertically, it generates closed frequency
item set trees composing of the prefixes of the branches in turn; horizontally, the Master distributes the Map
tasks, continuously generates map and reduce tasks to finish mining the closed frequency sequence item sets in
every sub-trees.

(3)Master deploys the position of the intermediate files generated in the Reduce nods. Sort the intermediate
results according to <k-sequence, result> pairs, regulate and simplify the local sub-trees generated by the CPUs
in nodes, and produce the output files of global sequence pattern. Then the MapReduce returns to the point
where it is called.

According to the analysis on the process, MR-BIDE makes full use of parallel computing of MapReduce
and processes the data respectively, and saves the pseudo projection database of the intermediate results into
HDFS-a distributive storage. It reduces the tension brought out by frequent scanning of the original database,
minimizes the space for searching and balances the task loads in different nodes, and realizes the process of the
parallel mining.

4. RESULTS OF EXPERIMENT AND ANALYSIS

The experiment was done under the operating system Centos 6.5. Operating under the Hadoop-2.6.5, this
small experimental platform was set up with a cluster of 5 data nodes and 1 namenode connected with a GbE
Switch. Table 1 is the configuration of the nodes.

| Table 1. Basic Configuration of the Nodes in Cluster |
|-----------------|-----------------|-----------------|
| Master Name     | Node Type       | IP              |
| Master          | Namenode        | 192.168.142.39  | Intel(R) core(TM) i5 CPU, 8G RAM |
| Slave1          | Datanode        | 192.168.142.40  | Intel(R) core(TM) i5 CPU, 8G RAM |
| Slave2          | Datanode        | 192.168.142.41  | Intel(R) core(TM) i5 CPU, 8G RAM |
| Slave3          | Datanode        | 192.168.142.42  | Intel(R) core(TM) i5 CPU, 8G RAM |
| Slave4          | Datanode        | 192.168.142.43  | Intel(R) core(TM) i5 CPU, 8G RAM |
| Slave5          | Datanode        | 192.168.142.44  | Intel(R) core(TM) i5 CPU, 8G RAM |

This experiment is testing the efficiency of the algorithm from two sides, using the logs as the dataset under
a private cloud environment in a lab. Respectively, it tests the efficiency and effectiveness of the algorithm with
the changing minimum support, the size of the dataset and the quantity of data nodes. Experiment 1 was done to
test the efficiency of MR-BIDE and BIDE mining in a stand-alone environment (using Master as a stand-alone
PC). The comparison is the processing time of the algorithms. Experiment 2 was done in the Hadoop cluster,
setting the minimum support threshold as 5%, to test the performance of the MR-BIDE algorithm using the data
set in Table 2. The reference is the speed-up ratio.

Figure 2 shows the comparison of processing time between BIDE and MR-BIDE when they are processing
the same dataset in a stand-alone environment, under different degrees of support. The experimental results
show that MR-BIDE is better than the BIDE algorithm in processing time.
Figure 2. Processing time in different degree of support

When the degree of support is less than 5%, i.e. when the minimum support threshold is set, with the decrease of the degree of support, the size of the result set becomes larger and the time consumed will be more. But the MR-BIDE, because it uses Mapreduce mode and mines the subsequence in parallel (including closed testing and extending judgment), it reduces the time cost in processing the middle result set so that they consume quite different time in operation with the minimum support. However, if the threshold of support is set smaller, the size of the sequence satisfying the conditions will become larger, the initial scanning time consumed will become longer. When the minimum support degree is more than 5%, the sequences satisfying the minimum support condition are reduced and the size of the result set becomes smaller, therefore, the running time of the algorithms becomes stable.

Table 2. Size of dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Lines of the log (10 thousand)</th>
<th>Size of the pre-processed file</th>
<th>Quantity of Sequences $10^6$</th>
<th>Average length of sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>50</td>
<td>298</td>
<td>2.124 930</td>
<td>123</td>
</tr>
<tr>
<td>D2</td>
<td>100</td>
<td>563</td>
<td>4.514 568</td>
<td>156</td>
</tr>
<tr>
<td>D3</td>
<td>200</td>
<td>1267</td>
<td>9.462 360</td>
<td>107</td>
</tr>
<tr>
<td>D3</td>
<td>400</td>
<td>2321</td>
<td>21.654 201</td>
<td>141</td>
</tr>
</tbody>
</table>

Figure 3. Performance Testing of MR-BIDE Algorithm in Hadoop Environment
The experimental results in figure 3 show that with the increase in the quantity of data nodes and data sets, MR-BIDE algorithm have a better performance in extending and speeding up ratio. It is mainly caused by the mode in which MR-BIDE uses an approach of building up multi-layer trees. The Map tasks are assigned by Hadoop to build up trees. With the quantity of DataNodes increased, the DataNodes will make full use of the resources and executes the pruning judgment and the extended judgment to balance the loads. The curve D4 in the figure is nearly in a line. In dealing with large-scale dataset, MR-BIDE has more advantage. However, with the increase in the cost of communication between many nodes, the speed-up ratio of this algorithm becomes slowdown.

5. CONCLUSIONS

The algorithm of MR-BIDE is dependable and efficient in processing large-scale datasets; particularly, when it is applied to process intensive data, there exists an ideal linear relationship between the speed-up ratio and the data nodes. It can be relatively fast to get a global sequential pattern in a multi-node task. While processing relatively sparse data, the DataNode costs much time in network communication. Since this experiment was made in the lab with high speed but limited datasets, it is necessary to find out a better way to reduce the cost of time in the communication between different nodes, so that the algorithm could be optimized and deployed in practical distributed environment.

ACKNOWLEDGEMENTS

This work was supported by Scientific Research Programme for Higher Education of Shandong Province(NO.J15LN15) and Key statistical scientific projects of Shandong Province(NO. KT15217).

REFERENCES