The Research of Decision Tree Mining Based on Hadoop

B.Sivasandeep
M TECH(CSE)
K.I.T.S,DIVILI

B.Venkateswara Reddy
ASSISTANT PROFFESOR,
K.I.T.S,DIVILI

Abstract
For a single node massive data, the mining calculation of the decision tree is very large. In order to solve this problem, this paper proposes the HF_SPRINT parallel algorithm that bases on the Hadoop platform. The parallel algorithm optimizes and improves the SPRINT algorithm as well as realizes the parallelization. The experimental results show that, this algorithm has high acceleration ratio and good scalability.

Keywords- Hadoop, MapReduce, SPRINT

INTRODUCTION
Decision-tree is one of the key Data Mining technologies and categorization based on decision tree has always been research focus. However, current researches on decision tree mining algorithm mainly focus on improving the mining algorithm which only improves the efficiency of the mining system but not the data processing capability. With the rapid development of computer and networking technology, the mass of data increases exponentially, which makes the Single point data mining platform unsuitable for data analysis? To solve this problem, cloud computing is required. Cloud computing is the result of distributed processing, parallel processing and grid computing. Distributed and parallel massive data computing and processing are the keys of cloud computing. Thus, we can solve the massive data mining problem by parallelizing traditional decision-tree algorithms and then running them through cloud computing. In this article, taking into account the characteristic of decision-tree, we propose a data mining platform based on Hadoop. The effectiveness and efficiency of the platform are then evaluated through an improved parallelizing decision-tree algorithm.

II. HADOOP RELATED TECHNOLOGIES
Hadoop is an integral distributed software framework and is the most commonly used framework to build the Cloud computing platform. It uses Map Reduce as programming model. MapReduce framework supports distributed computing on large data sets on clusters of computers. MapReduce takes huge amount of data sets, partitions them into smaller sub-sets and distributes them to different computers to process paralleled. MapReduce views data as <key, value> pairs. The data processing is into two steps: Map and Reduce.

In the Map step, MapReduce takes the input, partitions them into independent datasets and processes them paralleled. The outputs of Map are then sorted and combined and are taken as the input of the Reduce step. The intermediate I/O results are stored in HDFS file system. JobTracker is responsible for scheduling and monitoring all tasks and rescheduling failed tasks. TaskTracker executes the designated Map and Reduce tasks. The flow chart of Hadoop is shown in Fig. 1.

Hadoop MapReduce abstracts the complicated parallel computing into two functions, Map and Reduce, and partitions huge data sets into Splits. Every Split is handled by one node in the clusters of computers and output intermediate results. All the intermediate results are combined by huge amount of nodes to output the final result. We can see that the parallel procedure based on Map/Reduce is very simple in the sense that we only need to take care of the implementation of Map and Reduce function, the partition of data sets and categorization of Map tasks vs. Reduce tasks. All other complicated parallel computing programming problems, such as distribute storage, task
scheduling, load balancing, error tolerance, and network communication, are taken care of by Map/Reduce framework.

![Hadoop execution flow](image)

III. DECISION-TREE ALGORITHM DESIGNS BASED ON HADOOP

A. The SPRINT algorithm

Decision-tree is one of the important branches of data mining algorithms. Most of the decision-tree algorithms, as ID3, C4.5, CART, and etc., require that the training sample datasets stay in memory, which is impractical for data mining involving with thousands and millions datasets. To address the problem of limited main memory, John Shafer proposed SPRINT to apply to very large scale training sets and create compact accurate decision-tree. SPRINT has good expansibility and parallelizability, does not limited by the size of memory, runs fast, and allows multiple processors create a decision-tree model at collaboratively. In this article we take SPRINT as an example and discuss the design of Hadoop based decision-tree algorithms.

SPRINT uses two data structures: attribute lists and histograms. Attribute record consists of a triplet of (attribute value, class label, index of the record). The attribute lists are partitioned as the tree is grown and associated with children nodes. Histograms are associated with decision-tree nodes to describe the capture distribution of certain attribute. Two histograms C_below and C_above are associated with a node to describe classification distribution of continuous attributes; C_below maintains the distribution for attribute records that have already been processed. C_above maintains those that have not. Both values are updated as the algorithm runs. Categorical attributes have one histogram associated with a node. Serial SPRINT uses Gini index to find the best split points. It uses depth-first strategy. The detail procedure is shown below [2].

1) Create root node N. Establish attribute lists for all attributes. Pre-sort all attribute lists with continuous attributes. Associate all the attribute lists with N.
2) Create the tree for node N as following CRE_Tree algorithm.

**The algorithm of CRE_Tree is shown below:**

1) If all attribute lists are of the same class, return, otherwise, go to b).
2) Scan attribute lists and update histograms
3) Compute the minimum Gini index for candidate split points to get the best split point
4) Use the best split point found to split into N1 and N2
5) Partition the attribute list and associate them with N1 or N2
6) Repeat CRE_Tree to create tree for N1 and N2.

B. Parallelizing SPRINT

Based on the major steps of SPRINT, the study of parallelizing SPRINT focuses on following three aspects [3].

1) Split training datasets into multiple processing units parallely.
2) Find the best splits parallely
3) Create children nodes and associate attribute lists with them parallely.
C. Enhancement for SPRINT

The most time consuming part of SPRINT is to compute the Gini indexes for candidate split point datasets. The amount of computation is especially large for finding the best split points for continuous attributes. SPRINT scans an attribute list from the first attribute recode to the last one, computes Gini index for every candidate split points \(v\) (\(v\) is the average value of current record and the next record), and find the best split point with the minimum Gini index.

In this method, if node \(S\) has \(N\) training samples, the number of candidate split points will be \(N-1\). Although the attribute records are independent with each other and the computation can be done parally, the amount of computation to find the best split point for continuous attributes is still very large due to the huge amount of data involved in data mining. Thus, it is necessary to reduce the computation and improve the parallel computation efficiency.

We apply the node-partition method from [4] to parallelizing SPRINT and implement this algorithm in the Hadoop framework. When computing the best split point for continuous attributes, it does not require calculating all the candidate split points but only for those adjacent attribute records belong to different classes, which reduces the amount of computation significantly. We call the improved algorithm F_SPRINT.

D. Implementation of F_SPRINT in the Hadoop framework

We implement the improved SPRINT algorithm in the MapReduce framework of Hadoop parallyly and name it HF_SPRINT. HF_SPRINT consists of two parts. The first part has two JobTrackers which are used to process data and create root node. The second part is for node split iteration. The flow chart of HF_SPRINT integrated with MapReduce is shown in Fig. 2.

JobTracker1 is responsible for data processing. It uses map to partition the dataset into attribute list and then assign Reduce function according to different attributes. Reduce function computes the best split point and Gini index for every attribute and write the result into files. JobTracker2 takes the output files of Reduce function as its input. It sends different output from JobTracker1 through map to different reduce. Reduce finds the minimum Gini index and best split attributes, splits the splitting attribute lists according to the split point and splits the non-splitting attribute list with hash table.
The second parts also use two JobTracker to implement the node split iteration. The map function of JobTracker3 is to map different attribute lists of different nodes into different Reduce. Reduce is responsible for computing the best split point and Gini index. The map of JobTracker4 takes best split points and Gini index as input and distributes different attribute lists associated with different nodes to Reduce. The function of Reduce is the same as JobTracker2. Reduce also labels the best split point as leaf node or non-leaf node.

MapReduce treats data as a serials of <key, value> pairs. For example, for the Reduce of JobTracker1, the input is <<current node, attribute name, attribute value, attribute class>, <class, index of the record>>, and the output is <<current node, attribute name, attribute value, attribute class>, <class, the index of the record, Gini index, split point>>. For JobTracker4, the Map function is only responsible for assigning attribute lists associated with different nodes to different Reduce, and the input of Reduce is <<current node>, <attribute name, attribute value, attribute class, class, index of the record>>. We can see that, the key of HF_SPRINT algorithm is to design the operations of MapReduce and corresponding <key, value> pairs.

IV. EXPERIMENT ANALYSIS

In order to verify the advantages and disadvantages of the algorithm we built a simulation platform consisting of six server clusters. One of them is Jobtracker node; one is a TaskTracker node. Six servers can all be used as computing nodes and data storage nodes. At the mean time, Xen virtualization technology is used to enable concurrent execution of multiple MapReduce operations on the same node. Six servers are all installed with the Hadoop-0.20.0 and the JDK. The platform program is completed using eclipse IDE. Test data set is selected from the UCI. Glass data set (Referred to as A), the Audit data set (Referred to as B), and the Converttype data set (Referred to as C). Set A is a Sample of nine attributes, a total of 214 samples. Set B is a data set of 14 attributes, including six continuous attributes totaling 48,842 samples. Set C is a data set of 54 attributes, nine continuous attributes totaling 581,012. In this study, we focus on analyzing the performance such as accuracy, acceleration ratio and scalability of the parallel H_SPRINT algorithms based on the Hadoop platform and HF_SPRINT.

### TABLE I. ALGORITHM RESULTS

<table>
<thead>
<tr>
<th>Number Of nodes</th>
<th>Data set</th>
<th>H-SPRINT</th>
<th>HF-SPRINT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>accuracy</td>
<td>Time</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>91.67</td>
<td>0.31</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>91.61</td>
<td>0.22</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>91.34</td>
<td>292</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>91.23</td>
<td>211</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>91.36</td>
<td>139</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>78.23</td>
<td>471</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>76.25</td>
<td>310</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>79</td>
<td>153</td>
</tr>
</tbody>
</table>

From Table 1 we can tell that HF_SPRINT parallel algorithm shortens the computation time and still achieves the same accuracy. For the parallel computation point of view, as the number of nodes increases, the complexity of the algorithm reduces while achieving certain level of accuracy, which implies that parallelization of HF_SPRINT is feasible.

A. Acceleration Ratio

Acceleration Ratio \( S = \frac{T_{2}}{T_{1}} \) is an important index of evaluating the performance of parallel computing. It describes the reduction of computing time achieved from parallelism computing. \( T_{1} \) define as the time spent to solve a problem by single node. \( T_{2} \) defines as the time spent to solve the problem by m nodes paralleled. In the experiment, data set B is used to generate sets of 50000, 100000, 200000 and 500000 records randomly. The generated sets are named D1, D2, D3, and D4 respectively.
From Fig. 3 we can see that the slope acceleration ratio tends to decrease as the number of nodes increases. The main reason for this is that as the number of nodes increase, the communication overhead between nodes increases. For comparatively larger data set, the acceleration ration increases linearly and the algorithm is more efficient, which means that the HF_SPRINT algorithm based on Hadoop platform can be used to solve the data mining problem for huge amount of data.

B. Scalability

To show the utilization efficiency of parallel computing algorithms, we define algorithm efficiency as the ratio of the acceleration ratio of the algorithm to the number of nodes, that is, $E=S/N$, where $S$ is the acceleration ratio and $N$ is the number of nodes. From Fig. 4 we can tell that the algorithm efficiency decreases in general. The curve for D4 is flatter compared to those of D1 and D2, which means that as the size of the data set increases, the efficiency of HF_SPRINT algorithm tends to be stabler, thus the algorithm has better scalability.

The parallelization of decision-tree algorithms is the research focus of solving the classification problem for huge size data sets. SPRINT algorithm is one of the decision-tree algorithms with good parallelizability, but finding the split point for continuous attributes is very time consuming. During our study of parallelizing the SPRINT algorithms, we propose the HF_SPRINT algorithm based on Hadoop platform. The simulation results show that compared to H_SPRINT, HF_SPRINT has good parallel computing capability and significantly reduces the computing time.
CONCLUSION

The parallelization of decision-tree algorithms is the research focus of solving the classification problem for huge-size data sets. SPRINT algorithm is one of the decision-tree algorithms with good parallelizability, but finding the split point for continuous attributes is very time consuming. During our study of parallelizing the SPRINT algorithms, we propose the HF_SPRINT algorithm based on Hadoop platform. The simulation results show that compared to HF_SPRINT, HF_SPRINT has good parallel computing capability and significantly reduces the computing time.

REFERENCES


