The Strategy of Classification Mining Based on Cloud Computing

Lijuan Zhang
College of electrical and Electronic Engineering
Shanghai Institute of Technology
Shanghai 201418 P.R. China
zljsmx@126.com

Shuguang Zhao
College of Information Science and Technology
Donghua University
Shanghai 201620 P.R. China
sgzhao@dhu.edu.cn

Abstract: Cloud computing provides cheap and efficient solutions of storing and analyzing mass data. It is very important to research the data mining strategy based on cloud computing from the theoretical view and practical view. In this paper, the strategy of classification mining in cloud computing environment is focused on. Firstly, cloud computing, Hadoop, MapReduce programming model, Classification mining and SPRINT algorithm are introduced. Then the parallelization of SPRINT for cloud computing environment is designed. It includes improved SPRINT algorithm, and the implementation procedure of the improved SPRINT algorithm on MapReduce. Finally, the Hadoop platform is built and the experiment for testing performance of the strategy as well as the improved algorithm has been done. The results show that the strategy designed in this paper can archive higher efficiency when doing classification mining in cloud computing environment.

Keywords: Classification Mining; Cloud Computing; Parallel; SPRINT; MapReduce

I. INTRODUCTION

Cloud Computing [1,2] is a new business model. It distributes the computing tasks to the resource pool constituted of a large number of computers, so that a variety of application systems can obtain computing power, storage space and a variety of software services on demand. The novelty of the Cloud Computing is that it almost provides unlimited cheap storage and computing power. This provides a platform for the storage and mining of mass data.

Based on parallel computing conditions and parallel mining demand under the cloud computing environment, we study decision-tree-based classification mining, SPRINT algorithm. We improve SPRINT algorithm in order to combine it with the MapReduce programming model of cloud computing and classify the mass data in parallel.

Hadoop[3,4] is an open source software of Apache. It composes of the storage modules HDFS, computing modules MapReduce, HBase, Hive, Pig, ZooKeeper, Cascading and other modules. They work together to solve the problems of storing TB/PB-level data, and the efficient, reliable, scalable computing on such a large amount of data.

Classification mining is an important research topic of data mining. The objective of classification is to build a model of the classifying attribute based upon the other attributes. Once a model is built, it can be used to determine the class of future unclassified records. Several classification models have been proposed over the years, e.g. neural networks [5], statistical models like linear/quadratic discriminants [6], decision trees [7][8] and genetic models[ll]. Among these models, decision trees are particularly suited for data mining [9][10]. SPRINT is a most widely used decision-tree-based classification algorithm, which removes all of the memory restrictions, and is fast and scalable. The algorithm has also been designed to be easily parallelized.

The rest of the paper is organized as follows: In Section 2 we discuss issues in building decision trees and describes the serial SPRINT algorithm. Section 3 describes the parallelization of SPRINT for cloud computing environment. In Section 4, we give a performance evaluation of the parallel algorithms on MapReduce.

II. SERIAL SPRINT ALGORITHM

A decision tree classifier is built in two phases: a growth phase and a prune phase. In the growth phase, the tree is built by recursively partitioning the data until each partition is either “pure” (all members belong to the same class) or sufficiently small (a parameter set by the user). The tree growth phase is computationally much more expensive than pruning, since the data is scanned multiple times in this part of the computation. Pruning requires access only to the fully grown decision-tree, and the pruning phase typically takes less than 1% of the total time needed to build a classifier. We therefore focus only on the tree-growth phase.

The well-known CART [7] and C4.5 [8] classifiers, for example, grow trees depth-first and repeatedly sort the data at every node of the tree to arrive at the best splits for numeric attributes. SPRINT, on the other hand, replaces this repeated sorting with one-time sort by using separate lists for each attribute. SPRINT uses two data structures called Attribute lists and Histograms.

A. Attribute lists

SPRINT initially creates an attribute list for each attribute in the data. Entries in these lists, which called attribute records, consist of an attribute value, a class label, and the index of the record (tid) from which these value were obtained. Initial lists for continuous attributes are sorted by attribute value once when first created. If

the entire data does not fit in memory, attribute lists are maintained on disk.

B. Histograms

For continuous attributes, two histograms are associated with each decision-tree node that is under consideration for splitting. These histograms, denoted as Cabove and Cbelowr are used to capture the class distribution of the attribute records at a given node. As we will see, Cblow maintains this distribution for attribute records that have already been processed, whereas Cabove maintains it for those that have not.

Categorical attributes also have a histogram associated with a node. However, only one histogram is needed and it contains the class distribution for each value of the given attribute. We call this histogram a count matrics.

Since attribute lists are processed one at a time, memory is required for only one set of such histograms.

III. PARALLELIZING CLASSIFICATION

We now turn to the problem of building classification trees in parallel. We again focus only on the growth phase due to its data-intensive nature.

In parallel tree-growth, the primary problems remain finding good split-points and partitioning the data using the discovered split points. In this section we will present how we parallelize SPRINT. It includes improvement of SPRINT Algorithm, the implementation procedure of the Parallelizing SPRINT algorithm on MapReduce and partition and distribution of Data Set.

A. Improvement of SPRINT Algorithm

Recall that the main data structures used in SPRINT are the attribute lists and the class histograms, we improved SPRINT algorithm to realize the parallelization, and the specific ideas are as follows:

According to certain principles, the training-set examples is horizontally divided into n data subsets which are sent to m nodes.

Each node then generates its own attribute-list partitions in parallel by projecting out each attribute from training-set examples it was assigned. Lists for categorical attributes are partitioned and require no further processing.

m nodes respectively scan the attribute lists either evaluating split points for continuous attributes or collecting distribution counts for categorical attributes.

Having determined the winning split points, each node is responsible for splitting its own attribute-list partitions.

recursively partitioning the sub set data until each partition is either “pure” (all members belong to the same class) or sufficiently small (a parameter set by the user).

B. Combination of the Improved SPRINT algorithm and MapReduce

Cloud computing uses MapReduce [9] programming model. MapReduce is the emerging parallel programming system invented by Google. It puts parallelization, fault tolerance, data distribution, load balancing into a library, and the system sums up all operations on the data to Map stage and Reduce stage. To submit handling procedures of all operations to the MapReduce, programmer only need to define Map function and Reduce function. Based on the size of the input data and the job configuration information, MapReduce system can automatically initialize the job to multiple same Map Tasks and Reduce tasks, and they respectively read different input data blocks and order the Map function and Reduce function to process.

Fig. 1 shows the architecture of MapReduce, which mainly consists of three modules. Client submits parallel processing job written by the user to the Master node; the master node automatically decomposes user's job into Map tasks and Reduce tasks, and assigns the task to the worker; Worker requests to the Master for tasks, while distributed file system consisting of multiple Workers stores input/output data of MapReduce.

We implement the improved SPRINT algorithm based on MapReduce programming model of cloud computing environment as follow steps:

1) Partitioning and distributing data:

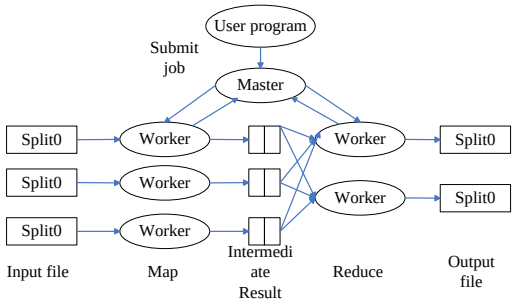

Figure 1. Aarchitecture of MapReduce

MapReduce library horizontally divides transaction database into n data subsets which are sent to m nodes executing Map tasks.

2)Formatting the data subset:

Format n data subsets as <key1,value1>, which is <<Ai, AC, Vj>,<idj, Cj>>, Ai is the ith attribute name of the training set; AC is the class of the attribute (continous or catagori); idj is the index value of the attribute redord in the training set; Cj is the class value of the attribute record.

3)Executing Map task

The task of Map function is scanning each record <<Ai, AC, Vj>,<idj, Cj>> of the input data subset and distributing the <key, value> with same Ai to the same

reduce node. Map function generates and outputs intermediate <key2, value2>, defined as <<Ai, AC, Vj>,<idj, Cj>>, which is ordered by Ai.

4)Executing Reduce task

The nodes assigned Reduce tasks read <<Ai, AC, Vj>,<idj, Cj>> submitted by Map function and build its own attribute lists and category histograms. After scanning each of the node's attribute lists (category histograms) and evaluate splits based on that attribute, the worker nodes compute the gini index respectively. Finally, the Reduce task outputs <key, value> as <<Ai, AC, Vj>,<idj, Cj, gini, split>>, gini is the least gini index of the corresponding attribute list, and split is the split point with the lowest value for the gini index.

5)Splitting the node

Once the best split point has been found, we execute the split by creating child nodes and dividing the attribute records between them. This requires splitting the node's lists for every attribute into two. Partitioning the attribute list of the winning attribute is straightforward. We scan the list, apply the split test, and move the records to two new attribute lists - one for each new child. Contemporarily, we insert the rids of each record into a hash table, noting to which child the record was moved. Once we have collected all the rids, we scan the lists of the remaining attributes and probe the hash table with the rid of each record. The retrieved information tells us with which child to place the record.

6)Repeating the above steps for the child node, until each partition is either "pure" or sufficiently small.

C.Partition and Distribution of Data Set

The cluster system under the cloud computing environment emphasizes high scalability and capability of parallel processing. Data partition and distribution impact load balancing and communication granularity. It is needed to partition and distribute data reasonably. There are two methods of data distribution:

1)Static data distribution

The process of data distribution is set at the stage of the program design. While executing the program, it distributes the data to nodes doing calculation using the prearranged distribution method. The static data distribution can deal with the problem of load balancing effectively under the light network load and machine load environment.

2) Dynamic data distribution

The process of data distribution is dynamic in the process of calculation. The master node doesn't distribute all the data once. It distributes the rest data when certain worker node

completes the calculation and transmits the result to the master node until all the data is processed.

In this paper, we use the method of data set distribution proposed in [12].

IV. EXPERIMENT AND ANALYSIS

In order to test the performance of the strategy designed by us, experiment has done on the Hadoop platform. In the experiment, the public data set provided by California University is chosen, which records information of blood donors in a blood center. The class attribute of the data set is whether the blood donor will give blood on March, 2007. There are four attributes: R, F, M, T. Their meanings are as follows: R expresses the months from the last denotes to now; F represents the total times of blood donation; M is the total amount that the donor has given(CC); T represents the months from the first denote to now.

In the experiment, Ubuntu + eclipse + hadoop are used as software environment, 10 PCs with not exactly the same configuration are used as hardware resources.

Fig. 2 shows the time effect of the improved algorithm and the data set distribution method.

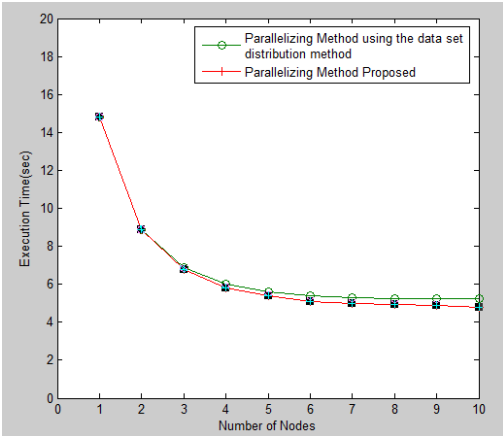

Figure 2: Experimental results

The curves reflect the changes of the time complexity with different number of nodes, and of the improved algorithm when using and not using the data set distribution method proposed in this paper. Apparently,

with the increase of the number of nodes in the cluster, the running time of our algorithm shows a more pronounced decline. And under the cloud computing environment, the improved algorithm has better performance using the data set distribution method.

ACKNOWLEDGMENT

This paper is supported by the Major Project of Shanghai Committee of science and technology(Grant no: 11510502700) , Major Project of Shanghai Board of Education(Grant no: 12ZZ189) and a grant from the science and technology development funds of Shang Institute of Technology(Grant no: KJ2012-06) .

This paper is recommended by Professor Xiaobin Li of College of electrical and Electronic Engineering of Shanghai Institute of Technology.

REFERENCES

[1] A Weiss. Computing in Clouds. ACM Networker, 11(4):18-25, Dec.2007.

[2] R Buyya, CS Yeo, S Venugopal, Market-Oriented Cloud Computing:Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications. 10(1): 5-13, 2008.

[3] Apache, Hadoop, http://lucene.apache.org/hadoop/, 2006.

[4] Zhu Zhu. Research and Application of Massive Data Processing Model Based on Hadoop [D]. Beijing: Beijing University of Posts and Telecommunications, 2008.

[5] R. Lippmann. An introduction to computing with neural nets. IEEE ASSP Magazine, 4(22), April 1987.

[6] M. James. Classificaton Algorithms. Wiley, 1985.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, 1984.

[8] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufman, 1993.

[9] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database mining: A performance perspective. IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December 1993.

[10] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. SLIQ: A fast scalable classifier for data mining. the Fifth Int'l Conference on Extending Database Technology (EDBT), Avignon, France, March 1996.

[11] Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters, Communications of the ACM, 51 (1):107 – 113, 2008.

[12] Lingjuan Li, Min Zhang. The Strategy of Mining Association Rule Based on Cloud Computing. 2011 International Conference on Business Computing and Global Informatization:475-478, 2011.