© 2014 IJIRT | Volume 1 Issue 5 | ISSN : 2349-6002

DATA CLASSIFYING LARGER DATASETS BY USING SPRINTS

K. Veeraprathap Reddy[1], T. Chandra Sekhara Reddy[2]

[1]M.Tech, CSE Dept, MLRIT, Hyderabad

[2]M.Tech(S.E), Associate. Professor, MLRIT, Dundigal, Ranga Reddy

***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.**

***Index Terms*- Hadoop, Map Reduce, SPRINT**

I. INTRODUCTION

Decision-tree is one of the key Data Mining technologies and categorization based on decision-tree has always been a 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, taking into account the characteristic of decision-tree, we propose a data mining platform based on Hadoop. The efficiency and efficiency of the are then evaluated through an improved platform parallelizing decision-tree algorithm.

Data Classification

Classification is an important data mining problem. Although classification is a well studied problem, most of the current classification algorithms require that all or a portion of the the entire dataset remain permanently in memory. This limits their suitability for mining over large databases. We present a new decision-tree-based classification algorithm, called SPRINT that removes all of the memory restrictions, and is fast and scalable. The algorithm has also been designed to be easily parallelized, allowing many processors to work together to build a single consistent model. This parallelization, also presented here, exhibits excellent scalability as well. The combination of these characteristics makes the proposed algorithm an ideal tool for data mining.

Challenges in Data Classification:

Classification is an important data mining problem. Although classification is a well studied problem, most of the current classification algorithms require that all or a portion of the the entire dataset remain permanently in memory. This limits their suitability for mining over large databases. The new decision-tree-based classification algorithm, called SPRINT that removes all of the memory restrictions, and is fast and scalable. The algorithm has also been designed to be easily parallelized, allowing many processors to work together to build a single consistent model. This parallelization, also presented here, exhibits excellent scalability as well. The combination of these characteristics makes the proposed algorithm an ideal tool for data mining

Discussion of problem:

Decision-tree is one of the key Data Mining technologies and categorization based on decision-tree has always been a 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.

II. PROBLEM STATEMENT

Decision-tree is one of the key Data Mining technologies and categorization based on decision-tree has always been a research focus. However, current researches on decisiontree 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, 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. Classification is an important data mining problem. Although classification is a well studied problem, most of the current classification algorithms require that all or a portion of the the entire dataset remain permanently in memory. This limits their suitability for mining over large databases. We present a new decision-tree-based classification algorithm, called SPRINT that removes all of the memory restrictions, and is fast and scalable. The algorithm has also been designed to be easily parallelized, allowing many processors to work together to build a single consistent model. This parallelization, also presented here, exhibits excellent scalability as well. The combination of these characteristics makes the proposed algorithm an ideal tool for data mining.

Classification has been identified as an important problem in the emerging field of data mining. While classification is a well-studied problem, only recently has there been focus on algorithms that can handle large databases. The intuition is that by classifying larger datasets, we will be able to improve the accuracy of the classification model.

Classification is an important data mining problem. Although classification is a well studied problem, most of the current classification algorithms require that all or a portion of the the entire dataset remain permanently in memory. This limits their suitability for mining over large databases. The new decision-tree-based classification algorithm, called SPRINT that removes all of the memory restrictions, and is fast and scalable. The algorithm has also been designed to be easily parallelized, allowing many processors to work together to build a single consistent model. This parallelization, also presented here, exhibits excellent scalability as well. The combination of these characteristics makes the proposed algorithm an ideal tool for data mining

III. SYSTEM DEVELOPMENT

Data Classification Algorithms:

Classification is an important data mining problem. Recent researches are on the data classification algorithms that can handle large databases. The intuition is that by classifying larger datasets, we will be able to improve the accuracy of the classification model.

The Classification problem:-

The classification problem may be informally stated as follows: We are given a training set consisting of many training examples, taken by uniformly sampling the instance space we wish to 'classify'. Each training example consists of a tuple with multiple attributes, one of which is the class label. The aim of classification is to process the training set and produce a classifier/model which 'accurately' describes each class. This model can then be used to classify data whose class label is unknown.

Decision-Tree Algorithm Designs

A decision tree classifier is usually built in two phases: A construction phase and a pruning phase. The construction phase is computationally more expensive than the pruning phase, since it involves multiple scans over the data, while the pruning phase only requires access to the fully grown decision tree.

Decision-tree is one of the important branches of data mining algorithms. Most of the decision-tree algorithms, such 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 we take SPRINT as an example and discuss the design of Hadoop based decision tree algorithms.

The SPRINT Algorithm:-

A decision tree based classifier called SPRINT (acronym for Scalable Parallelizable Induction of decision Trees) is presented, which, as the name suggests, is both scalable w.r.t. size of dataset as well as parallelizable. Let us look at some of the main ideas involved in this algorithm.

Decision-tree is one of the important branches of data mining algorithms. Most of the decision-tree algorithms, such 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 we take SPRINT as an example and discuss the design of Hadoop based decision tree algorithms.

Overview of SPRINT:

A decision tree classifier is built in two phases [3] [2]: 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). This process is shown in Figure 2. The form of the split used to partition the data depends on the type of the attribute used in the split. Splits for a continuous attribute A are of the form value(A) < c where t is a value in the domain of A. Splits for a categorical attribute A are of the form value(A) E X where X C domain(A). We consider only binary splits because they usually lead to more accurate trees; however, our techniques can be extended to handle multi-way splits. Once the tree has been fully grown, it is pruned in the second phase to generalize the tree by removing dependence on statistical noise or variation that may be particular only to the training set. 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. Our experience based on our previous work on SLIQ has been that 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. For pruning, we use the algorithm used in SLIQ, which is based on the Minimum Description Length principle.

Consider, for example, the credit rating problem, wherein a credit rating company wishes to classify customers based on a training database containing information about them. The classi cation tree is generated in a top down fashion as follows: The data is recursively partitioned until either each partition is su ciently 'pure' (parameterized by a user specified confidence), or is too small to yield statistically significant results. If neither of the above two criteria hold, the best possible split is chosen (For example, education level (e-level) at root node in and data is partitioned according to that split. We shall see in section 2.2.4 how the 'goodness' of a split is evaluated. As shown in figure , only binary splits are performed. For a continuous attribute A, it is of the form: value(A) < a, where a is a value in domain(A), while for a categorical attribute A, splits are of the form: value(A) 2 S, where S domain(A).

E-level in

{graduate, postgraduate}

Yes **No**

Rating=good **Salary > 60K**

Rating=good **Rating=Poor**

The Algorithm:-

The recursive algorithm is as given in Algorithm above. Below explains how split points are evaluated and how splits are performed.

Algorithm :-Flow of SPRINT algorithm

```
Partition(Data D)
begin
    if more than % (=confidence) of the elements in D belong to the same class OR size
        of D < minsize then
        return;
    end if
    for each attribute A do
        evaluate splits on attribute A (Section 2.2.4)
    end for
    Use the best split to partition D into D 1 and D2
    Partition(D 1)
    Partition(D 2)
end
```

The well-known CART [3] and C4.5 [2] 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. SLIQ, on the other hand, replaces this repeated sorting with one-time sort by using separate lists for each attribute (see [5] for details). SLIQ uses a data structure called a class list which must remain memory resident at all times. The size of this structure is proportional to the number of Figure 3: Example of attribute lists input records, and this is what limits the number of input records that SLIQ can handle. SPRINT addresses the above two issues differently from previous algorithms; it has no restriction on the size of input and yet is a fast algorithm. It shares with SLIQ the advantage of a one-time sort, but uses different- data structures. In particular, there is no structure like the class list that grows with the size of input and needs to be memory-resident. We further discuss differences between SLIQ and SPRINT in Section 2.4, after we have described SPRINT.

Data Structures:-

The two principal data structures used by SPRINT are attribute lists and histograms. An attribute list is a vertical projection of the training set, consisting of three columns: An attribute value, the class label and the id of the record from which these values were obtained. At the root of the classification tree, the attribute lists are got by projecting out the required attributes from the training set. At each node of the tree, whenever a split is performed, the attribute list corresponding to that node is partitioned as per the split condition, and each partition is associated with the corresponding child. The attribute lists for numerical attributes are sorted initially, and continue to remain sorted throughout the tree construction phase.

For categorical attributes, the histogram (also called count matrix) contains the class distribution for each value of the attribute. For numerical attributes, two histograms are associated: Cabove and Cbelow. These , as the notation suggests, keep track of the class distributions above and below certain values of the attribute.

IV. RELATED WORK

Data Placement and Workload Balancing

Recall that the main data structures used in SPRINT are the attribute lists and the class histograms. SPRINT achieves uniform data placement and workload balancing by distributing the attribute lists evenly over N processors of a shared-nothing machine. This allows each processor to work on only l/N of the total data. The partitioning is achieved by first distributing the training-set examples equally among all the processors. Each processor 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 therefore evenly partitioned and require no further processing. However, continuous attribute lists must now be sorted and repartitioned into contiguous sorted sections. For this, we use the parallel sorting algorithm given in [8]. The result of this sorting operation is that each processor gets fairly equal-sized sorted sections of each attribute list.

Finding split points

Finding split points in parallel SPRINT is very similar to the serial algorithm. In the serial version, processors scan the attribute lists either evaluating split points for continuous attributes or collecting distribution counts for categorical attributes. This does not change in the parallel algorithm - no extra work or communication is required while each processor is scanning its attribute-list partitions. We get the full advantage of having N processors simultaneously and independently processing l/N of the total data. The differences between the serial and parallel algorithms appear only before and after the attribute-list partitions are scanned.

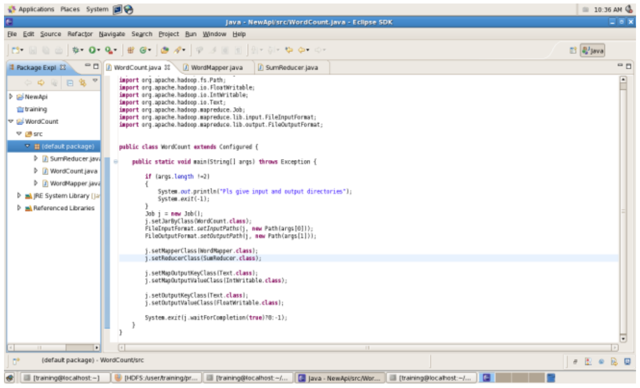
Continuous attributes

For continuous attributes, the parallel version of SPRINT differs from the serial version in how it initializes the C&low and C,,bove class-histograms. In a parallel environment, each processor has a separate contiguous section of a “global” attribute list. Thus, a processor’s Cbelou, and Cabove histograms must be initialized to reflect the fact that there are sections of the attribute list on other processors. Specifically, C&l- must initially reflect the class distribution of all sections of an attribute-list assigned to processors of lower rank. The C&,,,Ve histograms must likewise initially reflect the class distribution of the local section as well as all sections assigned to processors of higher rank. As in the serial version, these statistics are gathered when attribute lists for new leaves are created. After collecting statistics, the information is exchanged between all the processors and stored with each leaf, where it is later used to initialize that leaf’s C&we and Cb&vr class histograms.

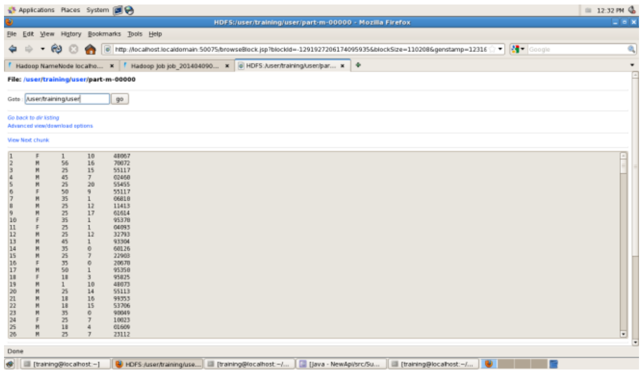
Categorical attributes

For categorical attributes, the difference between the serial and parallel versions arises after an attribute-list section has been scanned to build the count matrix for a leaf. Since the count matrix built by each processor is based on "local" information only, we must exchange these matrices to get the "global" counts. This is done by choosing a coordinator to collect the count matrices from each processor. The coordinator process then sums the local matrices to get the global count-matrix.

RESULTS



```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class WordCount extends Configured {

    public static void main(String[] args) throws Exception {

        if (args.length !=2)
        {
            System.out.println("Pls give input and output directories");
            System.exit(-1);
        }
        Job j = new Job();
        j.setJarByClass(WordCount.class);
        FileInputFormat.setInputPaths(j, new Path(args[0]));
        FileOutputFormat.setOutputPath(j, new Path(args[1]));

        j.setMapperClass(WordMapper.class);
        j.setReducerClass(SumReducer.class);

        j.setMapOutputKeyClass(Text.class);
        j.setMapOutputValueClass(IntWritable.class);

        j.setOutputKeyClass(Text.class);
        j.setOutputValueClass(FloatWritable.class);

        System.exit(j.waitForCompletion(true)?0:-1);
    }
}
```



File: /user/training/user/part-m-00000

```
1       F       1       10      48067
2       M       56      16      70072
3       M       25      15      55117
4       M       45      7       02460
5       M       25      20      55455
6       F       50      9       55117
7       M       35      1       06810
8       M       25      12      11413
9       M       25      17      61614
10      F       35      1       95370
11      F       25      1       04093
12      M       25      12      32793
13      M       45      1       93304
14      M       35      0       60126
15      M       25      7       22903
16      F       35      0       20670
17      M       50      1       95350
18      F       18      3       95825
19      M       1       10      48073
20      M       25      14      55113
21      M       18      16      99353
22      M       18      15      53706
23      M       35      0       90049
24      F       25      7       10023
25      M       18      4       01609
26      M       25      7       23112
```

V. 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 H_SPRINT, HF_SPRINT has good parallel computing capability and significantly reduces the computing time

REFERENCES

[1] Lei Wan-yun. Cloud computing technologies, platforms and application cases[M].BeiJing. Tsinghua University Press 2011.5:pp222~224

[2] Yang Chen-zhu. The Research of Data Mining Based on HADOOP [D]. Chong Qing. Chongqing University 2010.11:pp42~43

[3] Zhu Min,Wan Jian-yi,Wang Ming-wen. Design and implementation of parallel decision tree classification algorithmbased on MR[J]. Journal of Guangxi Normal University Natural Science.Vol.29 No.1 2011.3 : pp84

[4] Peng Cheng,Luo Ke. Improving the Method Used by SPRINT Algorithm to Find the Best Split Point of Continuous Attribute[J]. Computer Engineering and Applications 2006.27:pp155~156

[5] http://archive.ics.uci.edu/ml/datasets.html

[6] Li Ying-an. Research on Parallelization of Clustering Algorithm Based on MapReduce[D]. Zhongshan University 2010:pp30~33

AUTHOR DETAILS:

First Author: ***K. VeeraPrathap Reddy*** received B.Tech Computer Science and Engineering from Abdul Kalam Institute of Technological and Sciences, Kothagudem, in the year 2011. He is currently M.Tech student in Computer Science and Engineering Department from MLR Institute of Technology. His research interested areas are in the field of Cloud Computing, Mobile Computing, Networking and Information Security.

Second Author: T. Chandra Sekhara Reddy working as an Asst. Professor & Convener & Co-Ordinator for Industry Institute Partnership (IIPC) in MLR Institute of Technology. He has completed his M.Tech (S.E) from JNTU, Hyderabad and he has 7 years of teaching experience. His research interested areas are Wireless Sensor Networks, Data Mining, Data base.

Third Author: G Kiran Kumar is working as Associate Professor & HOD-CSE in MLR Institute of technology. He did M.Tech from Osmania University, Hyderabad, and submitted Ph.D from Nagarjuna University. His research areas include Data Mining, Spatial data mining, Software Engineering.