Parallel K-Medoids Clustering Algorithm Based on Hadoop

Yaobin Jiang
Department of Computer Science and Technology,
East China Normal University, ECNU
Shanghai, China
jybecnu@163.com

Jiongmin Zhang
Department of Computer Science and Technology,
East China Normal University, ECNU
Shanghai, China
jmzhang@cs.ecnu.edu.cn

***Abstract*: The K-Medoids clustering algorithm solves the problem of the K-Means algorithm on processing the outlier samples, but it is not be able to process big-data because of the time complexity[1]. MapReduce is a parallel programming model for processing big-data, and has been implemented in Hadoop. In order to break the big-data limits, the parallel K-Medoids algorithm HK-Medoids based on Hadoop was proposed. Every submitted job has many iterative MapReduce procedures: In the map phase, each sample was assigned to one cluster whose center is the most similar with the sample; in the combine phase, an intermediate center for each cluster was calculated; and in the reduce phase, the new center was calculated. The iterator stops when the new center is similar to the old one. The experimental results showed that HK-Medoids algorithm has a good clustering result and linear speedup for big-data.**

Keywords: Clustering Analysis; K-Medoids; MapReduce; Hadoop; Big-Data

I. INTRODUCTION

With the rapid development of the information society, the amount of data collected for analysis has a sharp rise, so the computing tasks of clustering analysis is more onerous. K-Means and K-Medoids clustering algorithm are two simple partitioned clustering algorithms, which were widely used in scientific research and practice, and researchers have made a lot of improvement. PAM (Partitioning Around Medoids) algorithm is the first implemented K- Medoids algorithm [1], the CLARA (Clustering LARge Applications) and CLARANS are optimal algorithms by reducing the sample size [2], and other optimized mode by pre-analysis to get the initial centers or using the distance matrix [3]. However, those improved algorithms still have large time complexity and space complexity for big-data.

MapReduce programming model was proposed for distributed system by Google Labs [4], the Apache Hadoop is an open source platform, which has implemented this model, has been widely studied and used in the industry [5]. The parallel K-Means algorithm based on MapReduce has been implemented in Mahout [6].

After studying K-Medoids clustering algorithm and MapReduce programming methods, the parallel K-Medoids clustering algorithm HK-Medoids was proposed and has the related experimental verification.

978-1-4799-3279-5 /14/$31.00 ©2014 IEEE

II. MAPREDUCE PROGRAMMING MODEL

In each request to MapReduce is called job. To accomplish this job, in the map phase, the master node takes the input files, divides it into smaller sub-problems, and distributes them to the worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node; in the reduce phase, the master node collects all of the sub-problems answers and combines them in some way to form the output, which is the answer to the job which was originally trying to solve[7]. Fig.1 summarizes the MapReduce task execution process [8]. At first, the input file is divided into N parts, usually 16MB to 64MB. The master node is responsible for scheduling, allocate the jobs to the worker nodes; the work node reads the corresponding input files slice, and gets the key-value pairs. The key-value pairs are passed as parameters to map function and written to the local disk periodically. The intermediate Reduce worker node sorted the results of key-value pairs, and generated output file.

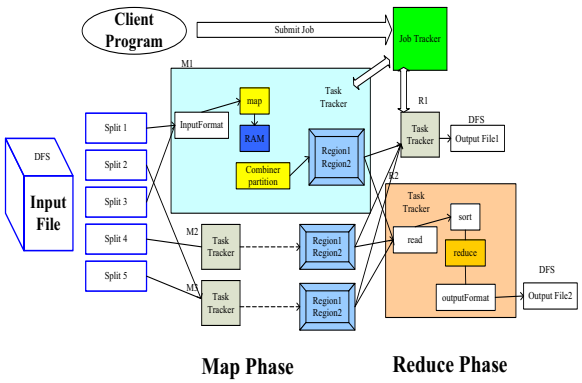

Figure 1: MapReduce mechanism

The cores of MapReduce programming model are Map and Reduce functions, which should be implemented by programmer to the actual use, the function is converting the

input key-value pairs (key, value) to another group of key-value pairs (key, value) by certain rules. Map function is parse the input file into a number of intermediate results List (<key2, value2>); Reduce function is using (key2, List <value2>) as input data, and combining them to get the final result. For all of the actual using, what the programmer only needs is implement the specific process of these two functions only; the other complex problems such as parallel computing task scheduling, machine communication, fault-tolerant operation can be handled automatically by Hadoop platform [8].

III. K-MEDOIDS ALGORITHM

Became of using the average value as the new center, the K-Means algorithm is inaccurate especially when there are some outliers in the clustering samples. K-Medoids algorithm selects the sample point, which has the minimum cost, as new center for the cluster, and can get rid of the impact of outliers.

A. PAM Algorithm

PAM is a simple K-Medoids indirect clustering algorithm based on the similarity of the samples, the ideas are as follows: it uses an iterative, greedy approach to solve the problem, which traverses all of the sample points in each clusters, and select the point OT which has the minimum distance Et (in current cluster) to instead of the original cluster center, and continue the iterative clustering until the standard measure function began to converge and the cluster center not change [1].

PAM algorithms use the Euclidean distance between samples in the similarity constant typically, the distance is defined as follows:

$$\mathrm{dist}_{(x,y)} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_m - y_m)^2} \quad (1)$$

Where: $x = (x_1, x_2 \ldots x_m)$ and $y = (y_1, y_2 \ldots y_m)$ is the m-dimensional data sets of the two samples for clustering.

The clustering effect measure function E for PAM algorithm is defined as follows:

$$E = \sum_{i=1}^{k} \sum_{o_i \neq o_t} \mathrm{dist}_{(o_i - o_t)}^{2} \quad (2)$$

Where: n is the total number of data samples; k is number of clusters, $\mathrm{dist}_{(o_i - o_t)}^{2}$ is the Euclidean distance of the center o_i and the each sample o_t in every cluster.

B. PAM Design

PAM algorithm is described as: the clustering data set $O(O_1, O_2 \ldots O_i \ldots O_n)$, where $O_i = (o_{i1}, o_{i2} \ldots oim)$ is the m-dimensional object, the algorithm is to find a partition $P_k = (C_1, C_2 \ldots C_k)$, where the measure function E is the most minimum [1].

PAM is as follows: [9]

1. Initialize: randomly select *k* of the *n* data points as the Medoids
2. Associate each data point to the closest medoid.
3. For each medoid *m*
   a) For each non-medoid data point *o*
   b) Swap *m* and *o* and compute the total cost of the configuration
4. Select the configuration with the lowest cost.

Repeat steps 2 to 4 until there is no change in the medoid.

The PAM algorithm is not sensitive to the effective for abnormal data, but the calculation is very large, the time complexity of serial implement algorithm is very large, as $O(t * k * (n - k)^2)$, where t is the number of iteration, n is the total number of data set, k is the number of clusters. For big-data, n will be greater than k and t, then the elapsed time will increase quickly with the rapid growth of the n and decrease the efficiency of the algorithm, so the common K-Medoids clustering PAM algorithm only was used for small dataset. Therefore, the parallel K-Medoids algorithm, by using parallel processing model and increasing the computing nodes to speed up linearly, is a new idea in practical application.

IV. HK-MEDOIDS DESIGN

The idea of HK-Medoids clustering algorithm: Hadoop HDFS file system storage the samples file and the initial cluster center file, after reading the initial cluster center, the other samples were divided to the most similar cluster parallel by the Euclidean Distance in Mapper function, the temporary center for each clusters were calculated through Combiner function, and clustering again in Reducer function to identify the final center, which will be used in next iteration. And the iteration is running or not, is judged by the result of comparing with the new center and the old one for each cluster [10, 11].

The iteration can be divided into three steps, pseudo-code is as follows [12, 13]:

A. Map Function

Map function divide each sample to the nearest cluster, the algorithm time complexity of is $O(k * n_i)$, k is the number of clusters, n_i is the number of samples for each slice.

Algorithm1：map(Writable Clusters, Text lines)
input：all of Clusters information, lines of input files
output：<K, V> pair，K is the cluster， V is the sample

```
while(line = fs.readLine(lines)){
        Sample s = GetSampleFromString(line);
        minDist = Double.Max_Value;
        index = -1;
        for i=0 to clusters.length do{
            dist = ComputeDist(s,Clusters[i] .getCenter[i]);
             if(dist< minDist){
               minDist = dist;
               index = i;
            }
        }
        Output< Clusters[index], s> pair;
}
```

B. Combine Function

The computer node, which completes the map phase, runs the K-Medoids clustering with the clustered data to get the temporary center. There is the main time-consuming step of the algorithm, the time complexity of

about O (n_i^2 / k), which k is the number of clusters; n_i is the number of samples for each slice.

Algorithm2： combiner(Writable Cluster, Iterator< Writable > Samples)
input：The intermediate cluster information, all samples of the intermediate cluster
output: <K, V> pairs, K is the intermediate cluster, V is the intermediate center and the number of the samples which belong to the intermediate cluster

```
foreach(c in clusters){
    foreach(s in Samples){
        minDist = Double.Max_Value;
        index = -1;
        for i=0 to Samples.length do{
            distTemp += ComputeDist(s, Samples [i]);
        }
    If(distTemp< minDist){
        minDist = dist;
        index = i;
    }
    midValue = Samples [i] + Samples.length;
}
  Output< c, midValue > pair;
}
```

C. Reduce Function

Before every iteration stops, those intermediate clusters were clustering again, and calculate the new center for every cluster, the complexity of the algorithm is O(k*s), k is the number of clusters, s is the number of data split.

Algorithm3：reducer(Writable Clusters, Iterator< Writable >midValues)
input：The intermediate cluster information, the intermediate center and the number of the sample
output：<K, V>pairs，K is the cluster information， V is the new center for every cluster

```
foreach(c in clusters){
    Number [] nl;
    Samples [] sl;
    For i=0 to midValues.length do{
      sl[i] = midValues.getSample();
      nl[i] = midValues.getNumber();
    }
newCenter  nc = getNewCenter(nl, sl);
output<c, nc>;
}
```

V. Results and Analysis

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper.

A. Environment

Seven personal computers were used, one for master node, and the last six for work node. The CPU is Intel Dual E2200, the hard disk is 160GB/7200rpm/SATA, the memory is RAM 2G DDR2 800, and the network card is 100Mbps. Hadoop version 1.0.0, java version Java 1.6_31.

B. Experiment

Multiple sets of data were used to verify the implemented HK-Medoids clustering algorithm from both of the algorithm validity and the algorithm speedup.

1) validity

Experimental data is the classical and authoritative iris feature dataset, which can be downloaded from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/iris/, this dataset can be divided into three categories and each contains 50 data. Table 1 gives the average results of 10 clustering result using HK-Medoids algorithm, the overall results show that HK-Medoids clustering algorithm has good clustering effect even though Versicolor and Virginica have some crossover.

Table 1：HK-Medoids clustering results for Iris data

True	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	45	5
Virginica	0	2	48

2) Speedup

Speedup generally refers to the ratio of running time-consuming for the same task in a single processor system and parallel processor systems, and is used to measure the performance and effectiveness of parallel system [14]. Speedup is defined by the following formula:

$$S_p = T_1/T_p \quad (3)$$

Where:

p is the number of work node in Hadoop
T_1 is the execution time using one work node in Hadoop
T_p is the execution time using p work nodes in Hadoop

When $S_p = p$, this speedup is called linear speedup. To reflect the parallel processing of HK-Medoids algorithm, file split size was reduced so the number of blocks task is increased and the algorithm speed accelerate when sub-block record reduced. Experiments were conducted using three datasets of 60 dimension, Sample1: file size 9.3MB, 20000 records, a map block; Sample2: File Size 46.5MB, 100000 records, three map blocks; Sample3: File Size 232MB , 500,000 records, 15 map blocks. Random initial cluster centers were used to generate six clusters. Fig.2 is an average speedup of 10 times task.

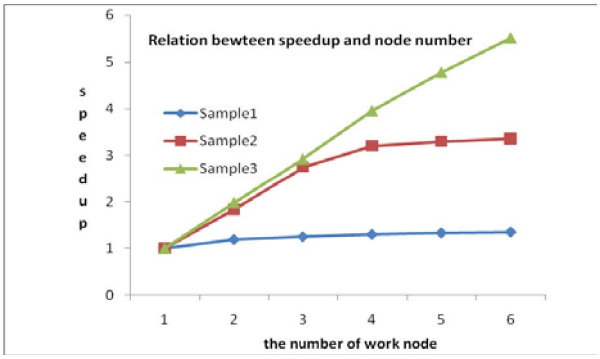

Figure 2： Relation of speedup and work node for HK-Medoids

C. *Analysis*

The results showed that the acceleration of the HK-Medoids clustering algorithm increases along with the expanded number of the work nodes. The growth rate of speedup decreases owing to the increased machine communication. And the speedup reach bottleneck easily when dealing with small-sizes dataset. For big-data, the algorithm HK-Medoids can obtain a linear speedup.

VI. CONCLUSION

Parallel K-Medoids clustering algorithm can be used for big-data, when based on MapReduce programming model on Hadoop cloud platform. The study shows that Parallel K-Medoids clustering algorithm HK-Medoids can obtain good clustering result for big-data. The following work will be focused on optimizing algorithm scheduling on Hadoop platform and selecting the initial cluster centers.

REFERENCES

[1] Tan, Pang-Ning; Steinbach, Michael; and Kumar, Vipin; Introduction to Data Mining [M] 2005

[2] Ng R T, Han J. CLARANS: A method for clustering objects for spatial data mining [J]. Knowledge and Data Engineering, IEEE Transactions on, 2002, 14(5): 1003-1016.

[3] Hae-Sang Park; Chi-Hyuck Jun. A simple and fast algorithm for K-Medoids clustering. H.-S. Park, C.-H. Jun / Expert Systems with Applications 36 (2009) 3336–3341

[4] MapReduce. wiki［EB /OL］[2014-02-10] http://en.wikipedia.org

[5] Apache. Hadoop［EB/OL］. http://hadoop.apache.org

[6] Apache. Mahout［EB/OL］[2014-02-10] http://mahout.apache.org

[7] 董西成 Hadoop 技术内幕:深入解析 MapReduce 架构设计与实现原理 [D] 机械工业出版社 (2013)

[8] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

[9] Sergios Theodoratos & Konstantinos Koutroumbas (2006). Pattern Recognition 3rd ed. p. 635.

[10] Wen J X L C X, Haitao Z X Y. Parallel implementing k-means clustering algorithm using MapReduce programming mode [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2011.

[11] 江小平, 李成华, 向文, 等. K-means 聚类算法的 MapReduce 并行化实现[J]. 华中科技大学学报: 自然科学版, 2011, 39(1): 120-124.

[12] Esteves R M, Pais R, Rong C. K-means clustering in the cloud--a Mahout test[C]//Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on. IEEE, 2011: 514-519.

[13] 聚类分析 K 中心点算法［EB/OL］[2014-02-10] http://blog.sina.com.cn

[14] Speedup.wiki［EB/OL］[2014-02-10] http://en.wikipedia.org