Research on Efficient K_Means Parallel Algorithm Based on Hadoop Distributed Architecture

Lin Qian[a], Lin Wang[b], Zhu Mei[c], Jun Yu[d], Guangxin Zhu[e], Debing Song[f] and Mingjie Xu[g]

State Grid Electric Power Research Institute (SGEPRI) Nanjing, China

[a]qianlin@sgepri.sgcc.com.cn, [b]wanglin18@qianlin@sgepri.sgcc.com.cn, [c]meizhu2016@aliyun.com, [d]yujun@sgepri.sgcc.com.cn, [e]zhuguangxin@sgepri.sgcc.com.cn, [f]songdebing@sgepri.sgcc.com.cn, [g]xumingjie@sgepri.sgcc.com.cn

Abstract. Focusing on the problems of K-means algorithm that has high time complexity, slow convergence, lower clustering accuracy, slow operating speed, an efficient K-means parallel algorithm based on Hadoop system and MapReduce framework is proposed. Firstly, the algorithm uses K selective sorting algorithm to improve the sampling efficiency; Secondly, the iterative center is updated by using the weight replacement policy; finally, the initial center point is obtained based on the sample pretreatment strategy. Experimental results show that the proposed algorithm not only has good convergence, accuracy and speedup, but also can improve performance of the algorithm.

1. Introduction

With the promotion of big data [1] and the emergence of various new applications, the data scale continues to expand from TB level up to PB level. At the same time, how to correctly analyze and efficiently mine representative data is the current research trend. Using clustering algorithm for information classification is an important method of processing these data.

K-means [2] is a common data mining algorithm, but the algorithm cannot meet the increasing data requirements in a single-machine serial environment, and the results of clustering depends on the initial value selection. For the shortcomings of the K-means algorithm, the literature [3] combined the Hadoop distributed platform to parallelize the K-means algorithm and proposed the PK-means algorithm. To reduce the transmission of intermediate data by designing the Combine function, and to deal with the problem of insufficient memory when the K-means algorithm processes massive data. However, the selection of the initial value has not been improved. In [4], the idea of Canopy-Kmeans algorithm is proposed by using the idea that the center points are as far apart as possible. The problem of initial value selection dependence is solved, but the central point is easy to fall into local optimum. On this basis, Ref [5] uses the dichotomy to improve it and proposes the BCkmeans algorithm, it avoids Canopy local optimal problems, but for the processing of massive data, the cost of center point selection is higher. Improved with intelligent algorithms, such as ant colony algorithm [6], genetic algorithm [7-10], etc., these improved algorithms can obtain good convergence. However, although the above algorithms for initial value improvement can obtain better initial values, the iterative center update uses mean replacement, which is susceptible to singular value points. In addition, these algorithms are preprocessing of global data. In the face of massive data analysis, the preprocessing cost is higher, which affects the overall performance of the algorithm.

Considering the above problems in the big data environment, this paper proposes an efficient K-means parallel algorithm based on MapReduce. The K-selection sorting algorithm is combined with the MapReduce framework for parallel sampling. The sample data preprocessing strategy is used to obtain the initial center, and the weight replacement strategy is used to update the iterative center. Experiments show that the algorithm not only improves the clustering quality, but also improves the efficiency of the algorithm in the face of massive data processing.

2. K-means algorithm

K-means algorithm is a clustering algorithm based on partition. Set data x {x_1 , x_2 ,..., x_n }, where each vector x_i in the data set represents a data object, and use Euclidean distance to judge the similarity between data objects. The larger the distance, the smaller the similarity. The Euclidean distance between data objects is shown in (1):

$$d(x_i, y_j) = \sqrt{\sum_{r=1}^{m} (x_{ir} - x_{jr})^2} \qquad 1 \le i \le n, 1 \le j \le n \tag{1}$$

Where m is the data set attribute dimension, x_{ir} , x_{jr} are the rth attribute values of the data objects x_i , x_j.

The K-means algorithm is described as follows:

Input: cluster number k and data set x output: k clusters that are iteratively terminated.

Step 1: For the data set x, k data are randomly selected as the initial cluster center.

Step 2: Calculate the Euclidean distance between the data points in the dataset and the k center points.

Step 3: Assign the data object to the closest cluster category.

Step 4: Find the mean of each cluster to determine a new center point.

Step 5: Determine whether the new and old center points have changed. If they do not change, the cluster ends. Otherwise, skip to step 2.

3. Efficient K-means parallel algorithm design

3.1. K-means algorithm improvement strategy

(1) K-means algorithm in the single-serial serial environment, in the face of massive data processing, often due to memory overflow problems cannot be clustered. Therefore, this paper uses the MapReduce computational model in the Hadoop platform to parallelize the improved K-means algorithm to adapt it to the processing of massive data.

(2) The selection of the initial value determines the effect of clustering. The selection of the initial value of the K-mean algorithm is random, resulting in unstable clustering results and poor accuracy. There are usually two ways to improve the initial value: combining intelligent algorithms with the method of maximizing the minimum distance. Both improvements are global preprocessing of the data, and the initial value selection is costly. Therefore, combined with the literature [11], literature [12] proposed a sample data preprocessing strategy to obtain the initial value. The data set sampling uses parallel random sampling of 3.1. According to the law of statistical large numbers [13] Bernoulli's law of large numbers and the central limit theorem, no matter what kind of distribution the whole obeys, as long as the mathematical expectation and variance exist, the capacity is extracted from it. For a sample of K, and when K is sufficiently large, tending to be normally distributed, the sample capacity formula is defined as follows:

$$\frac{N \times t^2 \times \delta^2}{N \times \varepsilon^2 + t^2 + \delta^2} \tag{2}$$

The total data set is *N*, the sample size is *K*, the error limit is , the probability is *t*, and the standard deviation is .

When random sampling is performed in various survey activities, the confidence is usually 0.95 and the probability is 1.96. Therefore, the experiment used a confidence of 0.95 and a probability t of 1.96.The data preprocessing is calculated as follows:

$$K_j = \sum\nolimits_{i=1}^{n} d_{ij} - d_{ij} \qquad j = 1,2,...,n \tag{3}$$

Where K_j is the distance between each data point in the sample and other data points, and d_{ij} is the Euclidean distance between the sample data points.

(3) Weight substitution strategy. The K-means algorithm center point iterative update is replaced by the mean method. This method makes the center point greatly affected by the singular value, and finally affects the clustering effect. Therefore, the weight substitution strategy is used to update the center point in the iterative process. Experiments show that this method has better clustering accuracy than the mean iterative update.

The idea of weight substitution [14]: In the iterative process, a weight is assigned to the data points in each cluster, the data points near the center point have larger weights, and the distances away from the center point are smaller. Finally, the data points in each cluster are accumulated to find a new cluster center, and replaced. The formula is defined as follows:

$$K_j = \frac{d_{jh}}{D} x_{j1} + \frac{d_{j(h-1)}}{D} x_{j2} + ... + \frac{d_{j2}}{D} x_{j(h-2)} + \frac{d_{j1}}{D} x_{jh} \tag{4}$$

3.2. Algorithm Description

The K-means optimization algorithm is described as follows:

Input: Cluster number k, data set D with N data objects.

Output: k clusters that meet the minimum distance criteria.

Step 1: Calculate the sample size using equation (2).

Step 2: K Select Sort Parallel Sampling.

Step 3: Sample preprocessing to obtain the initial center point, pre-clustering:

Step 3-1: Calculate the distance dij between each data point in the sample data by using the MapReduce distributed programming model, and save it in the distance relationship file. The distance used is measured as the Euclidean distance of the formula (1).

Step 3-2: For the distance relationship file saved in the previous step, calculate the value of using Equation (3), and select the first K values as the initial center point of the cluster, and save it in the cluster center file.

Step 3-3: Assign the data points to the cluster to which they belong according to the minimum distance criteria.

Step 4: Center iterative replacement.

Using the MapReduce distributed programming model, the new center point is calculated using equation (4) and the old center point is replaced.

Step 5: Data point allocation, iteration terminated.

Step 5-1: Reassign the data points in the data to the cluster to which they belong.

Step 5-2: Calculate whether the new and old center points converge. If it converges, iteratively terminates. Otherwise, jump to step 4 to recalculate.

The core code for weight distribution is as follows:

(1) KWMapper: map stage assigns each data point to the cluster center nearest to it, performs pre-clustering, and saves the result in the pre-cluster file.

```
setup(){center=Cluster center file} for(int I = 0; i<k; i++){
for(int j=0; j<value; dimension; j++){
dis+=Math.pow(center.get(i).get(j)-value.get(j));
}
If(dis<min){
min=dis;
Index = center.get(i);
}
Context.write(new Text(Index), new Text(value));
    }
```

(2) KWReducer: The reduction stage performs weight distribution on the result of pre-clustering. First, calculate the distance between each data point and the cluster center to which it belongs, and then assign weights to the data points in each cluster to generate a new center point.

```
setup(){cluster=Pre-clustered file} for(Text val:value){
for(int j=0;j<val.dimension;j++){
sum += Math. pow((key.get(j)-val.get(j)),2); sum=Math.sqrt(sum);
}
}
for(Text val:value){
for(int j=0;j<val. dimension;j++){
k=Math.pow((key.get(i)-val.get(j)),2); k+=Math.sqart(k)*val.get(j)/sum;
}
}
Context.write(new Text(null), new Text(k));
  }
```

The algorithm first obtains the sample size according to the sample capacity determination formula, and uses the K-selection sorting algorithm to perform parallel random sampling on the basis of the MapReduce framework, and saves the collected sample data to the sample file. Then, the initial value of the cluster is selected from the sample file through the sample preprocessing strategy, and pre-clustering is performed. The MapReduce job is started during the iteration, and the MapReduce task is performed once per iteration, and the new initial value is obtained by using the weight iteration replacement method. The perturbation of singular value point clustering results is reduced by mean iteration. When the initial value of the cluster satisfies the set threshold deviation, the iterative process ends, and the clustering result is saved in the final cluster file.

4. Experimental results and analysis

The experiment consists of 6 PCs, one of which acts as the master node for resource scheduling and allocation, and the remaining 5 as slave nodes, responsible for the task. The machines use the same configuration: 1 4-core CPU, 4G memory, 500G hard drive, CPU clocked at 2.9HZ, model Pentium(R) Dual-Core E6600, operating system Ubuntu 14.04LTS, JDK 1.7.0, The cluster is built using the Hadoop 2.2.0 version.

4.1. Hadoop cluster tuning

The number of data blocks allocated in the Hadoop cluster (the default size is 64M) determines the number of concurrent graphs. Therefore, the size of different data blocks and the number of map concurrency affect the efficiency of the algorithm.

The experiment content is in a Hadoop cluster with 64 cores and 4G memory, one of which is used as the NameNodes, and the other 5 are used as the DataNode to participate in the calculation. The data block size is adjusted by modifying the dfs.block.size setting of hdfs-site.xml. Set data blocks of different sizes for the three sets of data and adjust the number of map concurrency. The experimental data is based on Table 1 data. The specific allocation is shown in Table1. The running time of the algorithm is shown in Fig.1.

Table 1. Data block allocation

Dataset size	size				
	64	128	256	512	1024
0.62	9	5	3	2	1
1.2	20	10	5	3	2
1.8	29	15	8	4	2

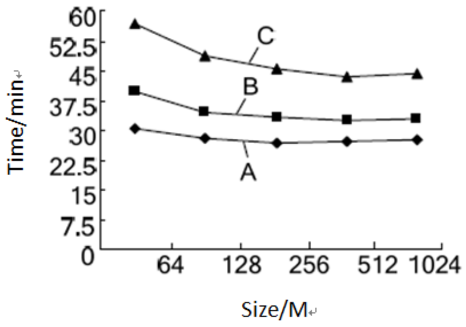

Figure 1. Running the graph of the algorithm for different map concurrency

It can be seen from Fig 1 that as the data block increases, the number of concurrent graphs decreases, and the running time of the algorithm does not decrease linearly with the decrease of the number of concurrent graphs. On the contrary, each data set has its most suitable map concurrency number.

Since the experimental configuration uses a 4-core CPU, 4 threads of parallel computing; excessive map concurrency will increase the map task assigned to each CPU core, and each CPU core runs the assigned map job, addressing The number of times is also increasing, increasing the overhead of the system. As the number of map concurrency decreases, this overhead is also reduced. However, when the number of map concurrency of the three sets of data is reduced to 2, the running time is increased. This is because each machine is doing double-threaded or single-threaded computing, and the computing resources are idle, and each CPU core is added. The amount of calculation does not fully reflect the advantages of cluster computing. From the above experiments, it can be concluded that when the number of map concurrency is close to the number of CPU cores, the efficiency of the algorithm can be improved.

4.2. Comparison of algorithm performance

In order to improve the performance of the algorithm, the data block files in the six Hadoop clusters are set to the most suitable map concurrency numbers of the three data sets, and the cluster performance test is repeated. Compare the performance of this algorithm with the algorithm and cluster tuning in the default environment. The specific task assignment is shown in Table 2, and the experimental results are shown in Fig 2.

Table 2. Block file settings

Data set size	**Block size**	**Map concurrency**
0.62	64	9
1.2	64	20
1.8	64	29
0.62	256	3
1.2	512	3
1.8	512	4

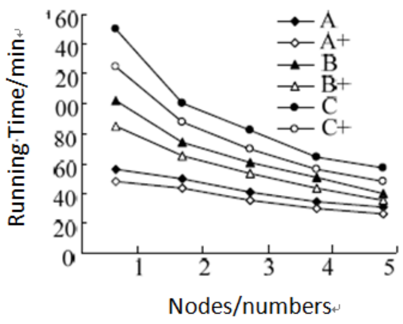

Figure 2. Comparison of algorithm performance

It can be seen from Fig 7 that the performance of the algorithm has been further improved after cluster tuning. Since the data block size is 64M in the default environment, in a parallel computing of a 4-core CPU, each core will be assigned multiple map tasks, and each map job will be addressed when running, so that the addressing time is the whole. The running time ratio is large; after cluster tuning, find the most suitable map concurrency number for each data set. Compared with the default environment, the number of concurrent graphs is reduced a lot, so the proportion of addressing time will be large. In addition, the advantages of cluster computing to process large data sets are fully utilized. Therefore, the performance of the optimized algorithm is further improved compared to the default environment.

5. Conclusion

(1) Comparison by stand-alone experiment, the algorithm in this paper has better and more convergence than the PK-means algorithm. UCI experiments show that the algorithm has better clustering accuracy than Canopy-Kmeans and BCkmeans.

(2) In the cluster performance test, by adjusting the number of cluster nodes and calculating the speedup ratio, the algorithm is adapted to the analysis and processing of big data.

(3) In the cluster tuning experiment, the performance of big data processing is further improved by adjusting the number of maps and the way of cluster memory.

Acknowledgments
This work was financially supported by the State Grid Corporation of Science and Technology (WBS number: 521104170019).

References

[1] OCHIAN A, SUCIU G, FRATU O, et al. Big data search for environmental telemetry [C] // IEEE International Black Sea Conference on Communications and NETWORKING. 2014: 182-184.
[2] ICHIKAWA K, MORISHITA S.A simple but powerful heuristic method for accelerating k-means clustering of large-scale data in life science [J]. IEEE/ACM Transactions on Computational Biology & Bioinformatics, 2014, 11 (4): 681-692.
[3] ZHAO W, MA H, HE Q. Parallel k-means clustering based on mapreduce [M] // CloudComputing. Springer Berlin Heidelberg, 2009: 674-679.
[4] 2014 (2): 29-31. ZHAO Qing. Efficient Algorithm of Canopy-Kmeans Based on Hadoop Platform [J]. School of Electronic Engineering, 2014 (2): 29-31.
[5] 2016 (5): 26-30, 25. XIAO Xueping, NI Jiancheng, CAO Bo. A BCkmeans parallel clustering algorithm based on Map-Reduce model [J]. Electronic Technology, 2016 (5): 26-30, 25.
[6] YU Qianqian, DAI Yueming, LI Jingjing. Parallel clustering algorithm ACO-K-means based on MapReduce [J]. Computer Engineering and Application, 2013, 49 (16): 117-120.
[7] 2014 (2): 657-660. JIA Ruiyu, GUAN Yuyong, LI Yalong. Parallel genetic K-means clustering

algorithm based on the MapRdeuce [J]. Computer engineering and design based on MapReduce, 2014 (2): 657-660.
[8] ZHU J, LI J, HARDESTY Eetal. GPU-in-Hadoop: Enabling MapReduce across distributed heterogeneous platforms [C] //Ieee/acis, International Conference on Computer and Information Science, 2014: 321-326.
[9] DAHIPHALE D, KARVE R, VASILAKOS A V, et al. An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications [J]. IEEE Transactions on Network & Service Management, 2014, 11 (1): 101-115.
[10] 2014 (4): 813-825. CI Xiang, MA Youzhong, MENG Xiaofeng. Top-K query method for large data in cloud environment [J]. Journal of Software, 2014 (4): 813-825.
[11] Güngör Z, Ünler A.K -harmonic means data clustering with simulated annealing heuristic [J]. Applied Mathematics & Computation, 2007, 184 (2): 199-209.
[12] Subramaniyaswamy V, Pandian S C.A Complete Survey of Duplicate Record Detection Using Data Mining Techniques [J]. Information Technology Journal, 2012, 11 (8): 941-945.
[13] KANARIS L, KOKKINIS A, FORTINO G, et al. Sample Size Determination Algorithm for fingerprint-based indoor localization systems [J]. Computer Networks, 2016 (101): 169-177.
[14] ZHANG J M.An Improved K-means Clustering Algorithm [J]. Journal of Information & Computational Science, 2013, 10 (1): 193-199.