Parallel Clustering Optimization Algorithm Based on MapReduce in Big Data Mining

Huajie Zhang, Lei Song, Sen Zhang

Abstract—**Traditional data mining algorithms have such defects as low computational efficiency and high memory usage, increasingly unsuitable for the current situation of big data processing. This article investigates Hadoop platform characteristics on the basis of the MapReduce framework mode, adopting the Top-K algorithm for parallel random sampling. To overcome the deficiency of conventional K-Medoids method in data processing and to optimize traditional algorithms, internal replacement strategy and horizontal performance expansion are adopted. Through the experimental test of the improved K-Medoids algorithm, a conclusion was obtained that the optimized parallel clustering K-Medoids algorithm based on the MapReduce framework has been significantly improved in terms of clustering accuracy, running time, speedup ratio and convergence, which meets the requirement of big data mining, analysis and processing.**

***Index Terms*—Clustering Algorithm, Big Data, Mining Algorithm, Optimal Design, Parallelization**

I. Introduction

DATA mining can be used to find valuable information from a large number of databases. By applying various methods to analyze the potential relationship between data and models in the database, the researchers better understand the relationship among the data so as to discover contents that are more useful but are easily overlooked [1-5]. Therefore, data mining is an important method of solving current rapid increase in the amount of data and hidden information. As a data processing link in data mining, it is necessary to preprocess various data in real time in the first place so as to perform redundant processing on the original data and convert them. The results are evaluated and optimized, and finally the effective data mined is output to the user [6-8].

With the advancement of data mining algorithms, the amount of data that needs to be processed on a daily basis has also increased sharply, which has higher requirements for data processing technology. If there is no good way to deal with these data, they will not play their due role and become garbage occupying storage space [9]. Both K-Medoids and K-Means, relatively classical clustering analysis algorithms, are comparatively simple to operate, are strongly sensitive to initial cluster centers, and easy to implement, and widely used in various fields of scientific research [10-15]. Compared with K-Means, K-Medoids in spite of the higher accuracy and higher time complexity, is more difficult to adapt to the clustering calculation in the data mining environment [16]. Thus, many scholars have optimized its algorithm. For example, it has been stated in the literature [17] that clustering in clusters can reduce the overall computational complexity, thereby improving the effect of clustering analysis, but it fails to solve the problem of high-dimensional datasets and the data with irregular shapes; Reference [18] adopts kernel-based adaptive clustering, which overcomes the sensitivity of K-Medoids to the initial value, but it does not reduce the time complexity. Some scholars also propose to combine K-Medoids and K-Means to effective exert the advantages of such two algorithms to make up for their shortcomings [19], with the latter used as the initial center and the former to calculate the new cluster center; however, due to the fact that the additional components increase the complexity of the algorithm, it thus could not adapt to the real context of big data.

To ensure the accuracy and stability of the outcomes of clustering calculation in the current big data environment, this article uses MapReduce and Hadoop platforms, adopts the parallel random sampling based on the Top-K algorithm, and implements the optimal design for K-Medoids. Experiments prove that the optimized parallel clustering algorithm combined with the MapReduce framework is considerably improved in terms of clustering accuracy, running time, speedup ratio and convergence, and can better adapt to the clustering calculation in the context of big data mining.

II. Related Works

A. Hadoop Platform and MapReduce Framework

As an open source ecosystem platform, Hadoop provides parallel calculating for data across nodes. It can easily expand nodes and run parallel programs, which has super big data storage and processing capabilities, and is quick and efficient for users to write [20-22]. The Hadoop platform is mainly implemented with HDFS (distributed file system), MapReduce (distributed calculating framework) and Hbase (distributed database) as the core. These three core components schedule and process each other, and jointly build an ecosystem that can process massive data, and realize parallel clustering algorithm processing for massive data. The specific system composition of the ecosystem is shown in Fig.

Manuscript received April 06, 2022; revised January 03, 2023.

H. J. Zhang is a lecturer at the Engineering Training Centre, Zhengzhou University of Technology, Zhengzhou 450044, China. (corresponding author, phone: +86-13733858110, fax：+86-371-60610279, e-mail: 20146047@zzut.edu.cn).

L. Song is a lecturer at the Modern Education Center, Kaifeng Vocational College of Culture and Arts, Kaifeng 475004, China. (e-mail: songlei@kfwyxy.edu.cn).

S. Zhang is a Ph.D. student at the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China. (e-mail: napoleonz@163.com)

1.

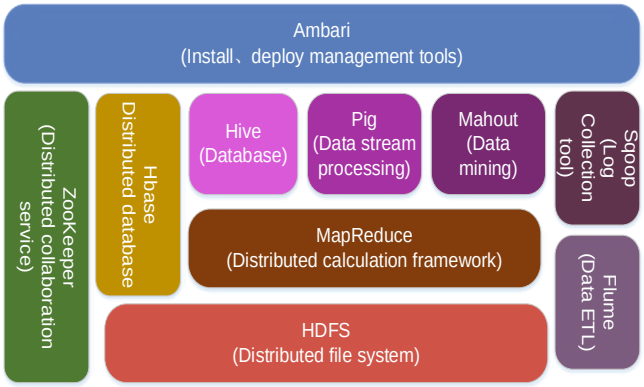

Fig. 1. Hadoop System Components

MapReduce used in this paper is a distributed computing framework model that can efficiently handle parallel operations on large-scale datasets [23, 24]. Map is a one-to-one mapping relationship of functions, and Reduce represents a many-to-one rule contract relationship between functions. MapReduce programming only needs to write Map and Reduce functions. Deployed on the Hadoop system platform, it can easily process parallelized big data. Fig. 2 shows the process of MapReduce processing data.

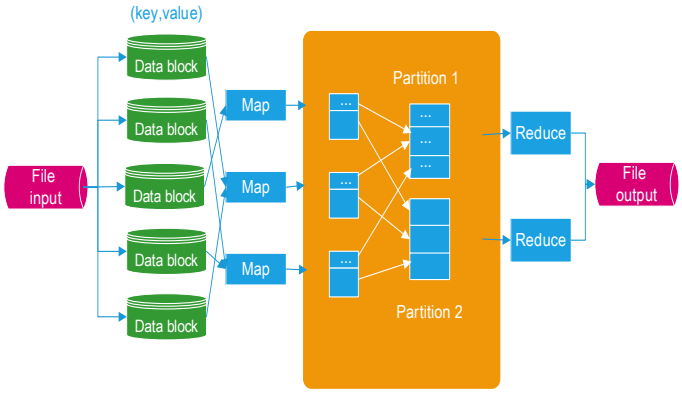

Fig. 2. Data Processing Procedure of MapReduce

B. Cluster Analysis Algorithms

The algorithm of cluster analysis is to classify the unknown dataset into certain categories on the basis of data similarity and the principle that the distance between the data points inside the cluster is the smallest and the distance between the external data points is the largest without giving a specific target. The method of transitioning the latter data objects to known effective measures [25-27]. According to the principle of similarity between data points, the original data consisting of the point set is reflected in the formula $V=\{V_i | i=1,2,...,n\}$, which is further divided into K groups $C_1, C_2, ..., C_k$, satisfying:

$C_i \cap C_j = \Phi, i \neq j$ and $\cup C_i = V$.

Among them, C_i is the class cluster, and V_i is the data object.

Both K-Means and K-Medoids belong to classic clustering analysis algorithms, with plain algorithm, simple programming, and strong sensitivity to the initial clustering center. The pseudo code of the K-Means algorithm execution process is as follows:

Algorithm: K-Means

Input: cluster quantity, the set K of points $D=\{D_1, D_2, ..., D_n\}$ to be clustered

Output: K set of clusters

Method:

1) D Arbitrarily select K a data point in the dataset as the initial cluster center;
2) Repeat;
3) For data points D_i, $i=1,2,...,n$;
4) Calculate D_i the Euclidean distance to all cluster centers;
5) Divide data points into clusters D_i that are closest to them;
6) For the clusters that have been divided C_j, $j=1,2,...,k$;
7) Solve the new cluster center again, that is, the mean of all points in each cluster in each dimension;
8) Calculate the change value of the cluster center twice J;
9) Until the change value J is within the threshold or the number of iterations reaches the upper limit.

III. Design and Optimization of Parallel Clustering Optimization Algorithm

A. The Traditional K-Medoids Algorithm Needs to Be Enhanced

The traditional K-Medoids algorithm has certain shortcomings in terms of storage, time complexity, initial center strategy, cluster center replacement, and computing performance:

1) Storage: In the event of big data, the storage of a single computer is hard to satisfy the computing demands of the K-Medoids method, which will cause a common problem with data: it fails to complete the expected clustering operations;
2) Complexity: The complexity of the K-Medoids method relatively exceeds that of algorithms such as K-Means. The time complexity is $O(n^2)$ that when processing a large amount of data, the traditional serial calculation process will cause a big CPU usage and the time required for calculation increases accordingly. Therefore it is difficult to adapt to the clustering operation in the big data environment;
3) Initial center strategy: The initial center selection will directly impact clustering effect of the entire algorithm. Most clustering algorithms use a random strategy for selecting the initial center by default, making the final result of the clustering difficult to guarantee. Although there are currently some selection strategies based on density and combined with artificial bee colonies, these methods only suit small amounts of data, and it is time-consuming by dealing with massive data. Therefore, the initial center selection scheme is also a problem that K-Medoids algorithm needs to solve;
4) Cluster center replacement: Although the traditional global sequential replacement strategy can ensure the reliability of the clustering results, it will lead to

excessive time loss in each iteration in the case of big data. In turn it will affect the time complexity and efficiency of this kind of algorithm. Accordingly, this research aims to find a new cluster center replacement strategy that can replace the current one;

5) Computing performance: The serial computing method implemented by the K-Medoids algorithm mainly uses vertical performance expansion to enhance the algorithm operation efficiency and increase the amount of algorithm data calculation.

When the data reaches a massive level, the vertical expansion can't satisfy the demands any longer, and it is easy to cause memory overflow and server warning. Therefore, when the parallel clustering optimization design of the K-Medoids method is implemented, this situation needs to be improved.

B. Improvement and Optimization of K-Medoids Algorithm

The traditional K-Medoids method has become increasingly incapable of processing big data. Sampling, as a bridge between traditional algorithms and massive data, should draw as many samples as possible. At present, there are two main forms of random sampling for data mining. One is byte-based random offset sampling. This sampling method is relatively simple, but due to repeated traversal, the time complexity will increase as the amount of data increases; the other is random traversal sampling based on global data. The method has high sampling efficiency in the case of a small amount of data, but cannot guarantee the format and content of the original data, which adds to the sampling time and complexity with the increase of sampling number.

With reference to the traditional K-Medoids method sampling, this study uses the Hadoop platform to establish a parallel sampling algorithm on the basis of the Top-K algorithm. The specific process is: first, input the sample size N and output the N number of samples; next in the map process, a unique random number is generated for each piece of data, the value is N output, and then the key is sorted and output according to the internal Shuffle of Hadoop; finally, the previous N / Rn piece of data is filtered and output Rn in the reduce process.

Improvement of K-Medoids algorithm

After the random sampling method of the K-Medoids method is determined, targeted improvements can be made to treat the issues in the conventional K-Medoids algorithm in terms of time complexity, storage, and computing performance; thus, it will improve the algorithm operating efficiency in big data environment.

1) In terms of storage: by using HDFS distributed storage, HDFS can partition big data and store it on different storage nodes. HDFS adopts a copy mechanism. For all data copies on the DataNode, the NameNode can monitor all the time. Once a computing node is down, the NameNode can immediately start the DataNode that stores the data copy. The advantages of using this method of storage are mainly reflected in: ① low cost: HDFS distributed storage does not require high storage devices, and only multiple ordinary devices can be used, which can effectively reduce equipment expenses; ② high security: HDFS has a perfect copy mechanism, and the system will automatically copy the stored data and store it on multiple storage nodes, which can better ensure data security; ③ high efficiency: HDFS stores data in blocks. So by simply accessing the directory of the management node, the operator will find the data storage location, with high data reading efficiency.
2) In terms of time complexity: the MapReduce programming framework can be used. In the past, the use of serial computing would increase the time complexity of it. Therefore, applying parallelization for optimizing the algorithm can transform the original time complexity into $O\left(n^2\right)/k$, with k standing for the number of nodes. Thus, in theory, using MapReduce for distributed computing can not only reduce time complexity, but also improve the computing efficiency of K-Medoids method.
3) In terms of the initial center strategy: the sample-based initial center selection scheme can be used for optimization. Parallel random sampling on the basis of the Top-K algorithm is used. The sample quantity is determined according M to the N number of data objects and quantity of cluster centers k. The computation formula for the samples number is below:

$$N = \frac{M}{500} + 100k \tag{1}$$

After data preprocessing, the formula below is used for calculation:

$$V_j = \sum_{j=1}^{n}((\sum_{l=1}^{n} dil) - dij),\ j = 1,2,...,n \tag{2}$$

The proportion of the sum of the distances between each point and the remaining points to the distances sum of all points can be calculated, and the point with the smallest k proportion is used as beginning cluster center. By this way, a better initial cluster center can be selected, which reduces the uncertainty and high consumption caused by random selection, and can makes the algorithm process more concise.

Clustering is unknown for the number and attributes of the divided classes. Usually, data objects need to be divided into multiple clusters, and the similarity of inner cluster objects is high. In most cluster analysis, the similarity between samples is mainly judged by distance. The distance is negative to the similarity. Algorithms like K-Medoids are common algorithms for calculating similarity based on distance. The most commonly used distance algorithms are as follows.

① Ming's distance:

$$d(x_i, j_i) = (\sum_{k=1}^{p}\left|x_{ik} - x_{jk}\right|^{q})^{1/q} \tag{3}$$

② Euclidean distance:

$$d(x_i, x_i) = (\sum_{k=1}^{p}\left|x_{ik} - x_{jk}\right|^{2})^{1/2} \tag{4}$$

When the Ming's distance $q = 2$ represent Euclidean distance.

③ Mahalanobis distance:

$$d_m^2(x_i, x_j) = (x_i - x_j)^T \sum^{-1} (x_i - x_j) \quad (5)$$

④ Range distance:

$$d_L(x_i, x_j) = \frac{1}{p} \sum_{k=1}^{p} \frac{|x_{ik} - x_{jk}|}{x_{ik} + x_{jk}} \quad (6)$$

Among them, the similarity of different object points in this paper is mainly calculated by Euclidean distance.

⑤ Full connection distance: Clustering requires objects in same cluster to be most similar. The more similar objects within a class, the better the clustering effect will be. For example, in a class set $A = \{a_1, a_2, \ldots, a_k\}$, the calculation formula of the similarity between classes is:

$$S_{inner} = \frac{\sum_{i=1}^{k} k \sum_{j=j+1} S(a_i, a_j)}{0.5 \times k \times (k-1)} \quad (7)$$

The a_i, a_j similarity between $S(a_i, a_j)$ classes is represented by, for a class a_i, the formula for calculating the similarity within a class is:

$$S_{inner}(a_i) = \frac{\sum_{i=1}^{k} k \sum_{j=j+1} S(x_i, x_j)}{0.5 \times n \times (n-1)} \quad (8)$$

4) Cluster center replacement, using the random replacement strategy within the cluster based on the cluster center, can not only meet the accuracy requirements of the clustering results, but also reduce the time needed for center replacement, which can treat the problem of low clustering efficiency.

In terms of computing performance, vertical performance expansion cannot satisfy the demands of massive data clustering, while horizontal performance expansion can improve different server and storage applications according to requirements. In comparison, Hadoop can dynamically change the number of computing nodes according to algorithm requirements, the computing performance of the cluster can thus be significantly improved.

Optimization of K-Medoids Algorithm by Hadoop

After improving five deficiencies of K-Medoids method, the Hadoop platform is applied for implementing secondary optimization of the algorithm, including:

The size of data shards can directly determine the number of maps, so the shard size has a decisive effect on the parallel efficiency of the algorithm. By adjusting the shard size of each map operation, higher clustering efficiency can be obtained, while analyzing the data results can lead to the best proportional link between the size of the dataset and data shard. In MapReduce, when dealing with massive data, the Map process can generate a large amount of data. Although horizontal expansion can improve computer performance, the limited bandwidth available between nodes limits the actual amount of data transmitted by MapReduce jobs. Hadoop outputs the specified combining function Combiner, which can be used as a Reduce task. By using Combiner, the processing burden of Reduce can be reduced and the operation efficiency of the entire algorithm can be improved. Fig. 3 demonstrates the specific output process.

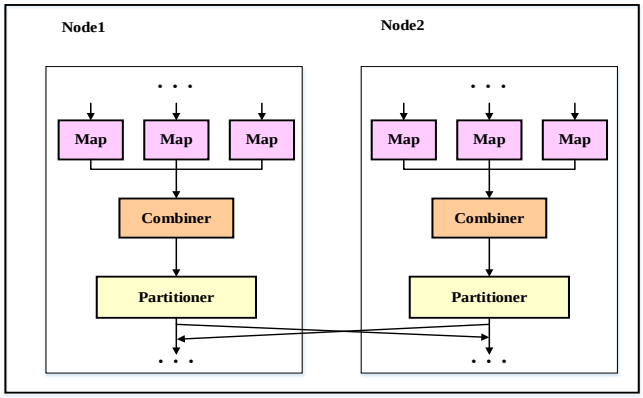

Fig. 3. Combiner Flow chart

In enhancing the performance bottleneck of conventional K-Medoids method in the process of big data processing, the algorithm is improved accordingly. The specific course of the enhanced K-Medoids method as follow: input the quantity of cluster centers k, and output k a cluster → Sampling and assigning object points based on Top-K calculation → Calculate the distance between object points and store them in the corresponding file → Calculate the corresponding V_j for each point j → Arrange V_j in descending order to determine the initial cluster center → Allocate the object points to the corresponding clusters according to the assignment principle →Calculate the sum of the distance between the cluster center and the object points → Replace the object points of the cluster center → Assign nodes, and terminate the judgment.

The main code of the driver class is:

```
Job job = new Job (conf, "KMedoidsDriver");
Job.setJarByClass(KMedoidsDriver.class);
Job.setOutputKeyClass(Text.class);
Job.setOutput ValueClass(Text.class);
Job.setReduceClass(KReduce.class);
Job.setMapperClass(KMapper.class);
FileSystem fs=Filesystem.get(conf);
Delete(new Path(args[2]), true);
```

It can be seen from the above that the processing process of the Hadoop-based K-Medoids parallel optimization algorithm is divided into two parts. The first part is to initialize the cluster center point set file, and divide the data set into several data blocks of a certain size, which are the inputs of the Map function for the system to process in parallel; while the second part is to start the Map task, execute the parallel processing by Reduce task, and finally produce the final clustering result.

IV. Experimental Results of Parallel Clustering Optimization Algorithm Based on MapReduce

A. Preparation of Experimental Environment

This research test uses six computers to form a Hadoop cluster, with Hadoop-2.2.0 version, Ubuntu14.04 operating system, NameNode node; besides, the CPU model of the master node configuration is Intel core i5-460M, and the memory/hard disk is 8GB and 1T respectively. Fig. 4 presents the cluster structure relationship composed of computers.

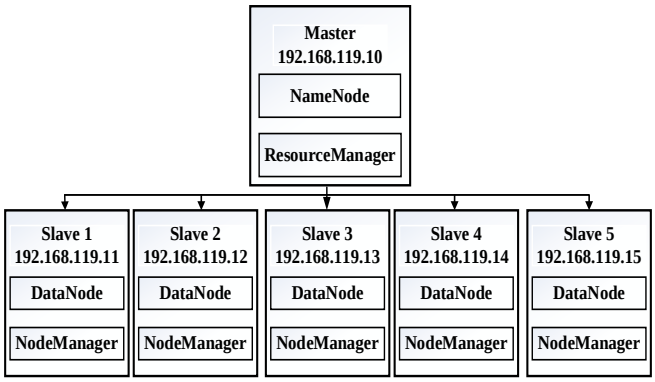

Fig. 4. Cluster structure diagram

B. Hadoop Platform Construction

Planning the entire Hadoop cluster is actually planning the individual components of the platform, such as HDFS. For the underlying HDFS components of the ecosystem, one of the nodes can be selected as NameNode to maintain the metadata information corresponding to the cluster system, and all the nodes in the cluster as DataNode can provide services for saving data files.

First, it needs to configure related files. The related configurations of the Hadoop platform are mainly stored in multiple xml files. Different xml files manage different attributes of Hadoop. Then, Hadoop installation is performed, including SSH login, host file modification and so on. For the underlying HDFS component of the Hadoop platform, a node needs to be chosen as the metadata point of the cluster to supply data storage services for other nodes as DataNodes. After the above platform is built, the improved algorithm property can be tested.

C. Analysis of Experimental Results

Accuracy Test

To test the effectiveness of optimized method, it's necessary to compare the conventional K-Medoids algorithm with the optimized one, and the clustering effect with accuracy. Select the standard data sets Wine and Iris to randomly generate records and add a small amount of noise for testing, compare the K-Medoids algorithm, partial optimization algorithm and the algorithm in the article for obtaining the precision rate, and judge whether the parallelized algorithm is effective by the accuracy rate and convergence time. Among them, the data type of the dataset is an interval scale variable, which is realized by Euclidean distance for measuring the dissimilarity between objects in the database. The precision rate denotes the ratio of data points that can be correctly clustered to all data points.

Table I and II exhibit the comparison results. In conclusion, the clustering accuracy and convergence time of the algorithm in the research have been improved, the accuracy rate has reached more than 95%, the results of multiple experiments are relatively stable, the fluctuation range is almost controlled within 5%, and the clustering effect is better. Meantime, the algorithm convergence time in this paper is the least in the data set.

TABLE I
COMPARISON OF ALGORITHM ACCURACY AND CONVERGENCE TIME OF DATASET *WINE*

Algorithm	Worst accuracy	Best accuracy	Average accuracy	Convergence time
K-Medoids	52.8	61.7	57.3	9.2347
Reference [18]	88.7	96.9	91.0	7.8832
Reference [19]	85.2	94.6	89.1	5.2179
Our model	92.8	97.5	94.2	4.0018

TABLE II
COMPARISON OF ALGORITHM ACCURACY AND CONVERGENCE TIME OF DATASET *IRIS*

Algorithm	Worst accuracy	Best accuracy	Average accuracy	Convergence time
K-Medoids	49.6	66.9	55.3	27.1444
Reference [18]	89.8	92.9	91.0	18.6563
Reference [19]	91.7	97.3	93.1	10.9727
Our model	93.6	97.3	94.8	9.9918

Single-machine Convergence Test

To test the convergence efficiency of such algorithm, 9000 data object points are selected to form four-dimensional coordinates, the size is 0.3MB, and cluster center quantity is 4. The convergence effect of the K-Medoids method before and after improvement is compared, and 10 of them are selected. The results are recorded in Table III.

TABLE III
CONVERGENCE TEST RESULTS BEFORE AND AFTER THE IMPROVEMENT (ITERATION NUMBERS/TIME REQUIRED)

Serial number	Conventional K-Medoids algorithm	Improved algorithm
1	5times/43.157s	4times/227.41s
2	9times/72.334s	6times/256.83s
3	6times/41.359s	3times/203.57s
4	8times/65.172s	5times/192.37s
5	7times/53.125s	6times/203.41s
6	8times/71.358s	5times/173.42s
7	5times/49.723s	4times/201.43s
8	6times/53.814s	6times/203.59s
9	7times/51.796s	5times/189.37s
10	5times/47.835s	6times/298.33s

Analysis of the convergence effects of the two methods in the table above reveals that the upgraded K-Medoids method features fewer mean iterations than the conventional algorithm in a single-machine pseudo-distribution environment, and has better global convergence. In terms of time consumption, the time consumed by the traditional method is nearly an order of magnitude less compared to that of the improved algorithm. On the horizontally extended Hadoop platform, the improved algorithm can obtain better convergence than the traditional one through sample

preprocessing and center point adjustment in the cluster, but the time consumption is greater than that of the traditional algorithm. The results may be attributable to the internal operation mechanism of Hadoop platform.

The Hadoop programming framework is mainly implemented through MapReduce. In the computing process, Hadoop will start HDFS, DataNode and other processes so that each process can communicate normally. Yet, the startup of processes such as the Map process, Reduce process, shuffle data sorting and data fusion segmentation will inevitably consume more time, so less time is actually used for calculation. Therefore, the operational advantages of the upgraded K-Medoids algorithm can only be fully demonstrated when dealing with massive data.

Cluster Runtime Test

In order to understand the cluster load capacity of the upgraded K-Medoids method in a cluster condition, the article selects 5 data sets for testing, of which the HDFS data block size is 128MB and the cluster centers quantity is 3. Table IV exhibits specific conditions of the datasets.

TABLE IV
DATA DETAILS OF 5 DATA SETS

Data set	Size (MB)	Number	Number of blocks
A	92	9600000	1
B	160	12384569	3
C	500	45376562	5
D	1329	176538924	10
E	3072	202583756	24

In the experimental environment of this paper, a total of 5 nodes from 1 to 5 is used in the calculation. The running time of the upgraded K-Medoids method under various nodes is shown in Fig. 5.

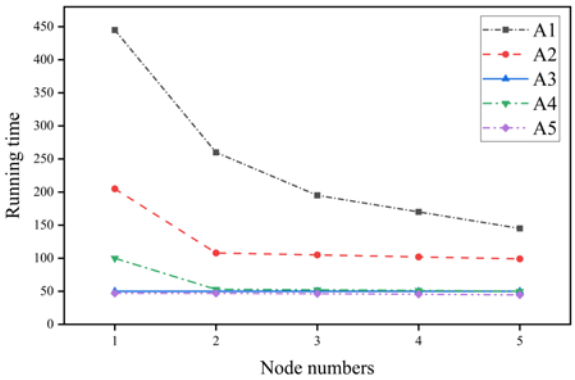

Fig. 5. Running time in different nodes of the improved algorithms

Through the analysis, a conclusion can be obtained that when the amount of data is small, the time required for clustering operations using different data nodes is not very different. Yet, with the continuous increase of the amount of data, the more nodes are used, the more time it takes. The reason for the analysis is that in the calculation process, as the dataset rises, the corresponding data will be evenly distributed to different nodes for calculation for maximizing calculation properties. Therefore, in the case of much data, the more computing nodes requiring more running time also shows a downward trend, indicating that the improved K-Medoids method has applicability in massive data clustering operations.

Cluster Scalability Test

Scalability is key to reflecting the impact of changes in the cluster nodes number on parallelized computing efficiency. The calculation formula is:

$$E = \frac{S}{P} \tag{9}$$

Among them, E refers to scalability, S denotes cluster parallelization speed ratio, and P denotes the quantity of parallelized computing nodes. With the above dataset, the nodes quantity is 1~7, and the scalability test outcomes of the K-Medoids algorithm are exhibited in Fig. 6.

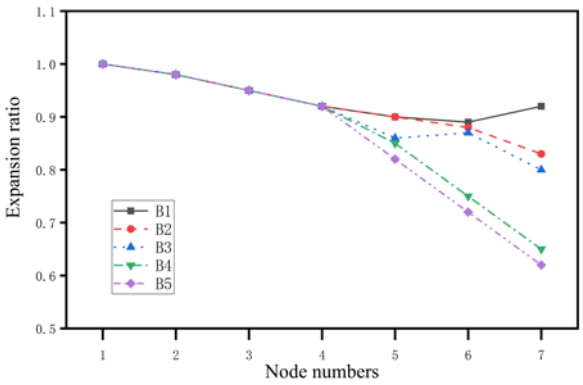

Fig. 6. Scalability test results of the improved algorithms

The above test results show that the entire upgraded K-Medoids algorithm features good scalability. As cluster nodes rise, the scalability of the K-Medoids algorithm presents a certain downward trend, especially for larger data sets. It may result from the maximum degree of parallelization. For example, after the nodes quantity in the A dataset reaches 5, the scalability reaches a horizontal state, indicating that the algorithm parallelization is maximized at this time. In addition, when nodes number and the size of the data increase proportionally, the improved K-Medoids algorithm maintains the same level of data processing and exhibits good scalability. Therefore, this algorithm is very valuable for large-scale datasets.

Cluster Speedup Test

Speedup, which represents the ratio of the execution time of the same task, is the most commonly used metric for parallel processing programs. The computing performance of cluster parallelization can be reflected by the speedup ratio, and the calculation formula is:

$$Sp = \frac{T_s}{T_p} \tag{10}$$

Among them is the time demanded for parallel execution, and means the time needed for serial execution. In the tests, the speedup ratio is positive correlated with the execution efficiency. The selected data packet size is 128MB, the number of clustering centers is 4, and 1-5 computing nodes are used for clustering respectively. Fig. 7 presents the

speedup test results.

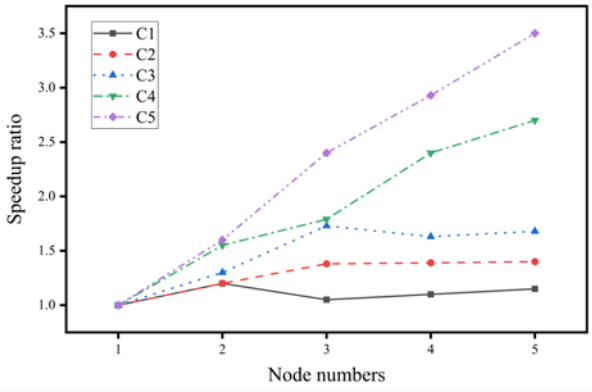

Fig. 7. Speedup test results of the improved algorithms

Analysis of the above figure tells that as the nodes quantity rises, the speedup ratio of the improved algorithm also features an upward trend. However, even in the event of big data, the speedup ratio of the upgraded algorithm is still high, indicating that the horizontal expansion of the cluster is utilized. It can significantly improve its operating efficiency.

V. Conclusions

With the rapid development of big data technology, the traditional clustering algorithms are increasingly unable to meet people's needs of data processing. This paper combines MapReduce and K-Medoids algorithm, and adopts parallel random sampling on the basis of Top-K algorithm, which avoids the low efficiency caused by random sampling. The conventional K-Medoids algorithm and its optimization can be improved by using intra-cluster replacement strategy, and horizontal performance expansion. Additionally, the upgraded K-Medoids algorithm is tested for accuracy, single-machine convergence, running time, scalability and speedup. The outcomes exhibit that the algorithm features good performance and high convergence and speedup. The experimental outcomes illustrate that the algorithm enhances data processing properties and can better meet the needs of current big data processing.

References

[1] A. Dogan and D. Birant, “Machine learning and data mining in manufacturing,” Expert Systems with Applications, vol. 166, pp114060, 2021
[2] G. Atluri, A. Karpatne and V. Kumar, “A survey of problems and methods,” ACM Computing Surveys (CSUR), vol.51, no.4, pp1-41, 2018
[3] M. K. Gupta and P. Chandra, “A comprehensive survey of data mining,” International Journal of Information Technology, vol. 12, no.4, pp1243-1257, 2020
[4] M. Y. Arafat, S. Hoque, S. Xu and D. M. Farid, “Machine learning for mining imbalanced data,” IAENG International Journal of Computer Science, vol.46, no.2, pp332-348, 2019
[5] A. Souri and R. Hosseini, “A state-of-the-art survey of malware detection approaches using data mining techniques,” Human-centric Computing and Information Sciences, vol.8, no.1, pp1-22, 2018
[6] M. Rodrigues, M. Y. Santos and J. Bernardino, “Big data processing tools: An experimental performance evaluation,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol.9, no.2, e1297, 2019
[7] M. I. U. Haq, Q. Li and S. Hassan, “Text mining techniques to capture facts for cloud computing adoption and big data processing,” IEEE Access, vol.7, pp162254-162267, 2019
[8] M. Uzun-Per, A. V. Gürel, A. B. Can and M. S. Aktas, “An approach to recommendation systems using scalable association mining algorithms on big data processing platforms: A case study in airline industry,” In Proceedings of the IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA), pp1-6, 2021
[9] H. Rangriz, Z. Bayrami Shahrivar, “The impact of E-CRM on customer loyalty using data mining techniques,” BI Management Studies, vol.7, no.27, pp175-205, 2019
[10] A. W. Rosyadi, N. Suciati, “Image segmentation using transition region and K-Means clustering,” IAENG International Journal of Computer Science, vol.47, no.1, pp47-55, 2020
[11] Y. Jeong, J.Lee, J. Moon, J. H. Shin and W. D. Lu, “K-Means data clustering with memristor networks,” Nano letters, vol.18, no.7, pp4447-4453, 2018
[12] P. Fränti and S. Sieranoja, “How much can K-Means be improved by using better initialization and repeats?,” Pattern Recognition, vol.93, pp95-112, 2019
[13] M. M. Fard, T. Thonet and E. Gaussier, “Deep K-Means: Jointly clustering with k-means and learning representations,” Pattern Recognition Letters, vol.138, pp185-192, 2020
[14] D. Yu, G. Liu, M. Guo and X. Liu, “An improved K-Medoids algorithm based on step increasing and optimizing medoids,” Expert Systems with Applications, vol.92, pp464-473, 2018
[15] M. Adepeju, S. Langton and J. Bannister, “Anchored K-Medoids: a novel adaptation of K-Medoids further refined to measure long-term instability in the exposure to crime,” Journal of Computational Social Science, vol.4, no.2, pp655-680, 2021
[16] W. Budiaji and F. Leisch, “Simple K-Medoids partitioning algorithm for mixed variable data,” Algorithms, vol.12, no.9, pp177, 2019
[17] E. Schubert and P. J. Rousseeuw, “Faster K-Medoids clustering: improving the PAM, CLARA, and CLARANS algorithms,” In Proceedings of the International conference on similarity search and applications, Springer, Cham, pp171-187, 2019
[18] T. N. Mau and V. N. Huynh, “Kernel-Based K-Representatives Algorithm for Fuzzy Clustering of Categorical Data,” In Proceedings of the IEEE International Conference on Fuzzy Systems, pp1-6, 2021
[19] S. Mousavi, F. Z. Boroujeni and S. Aryanmehr, “Improving customer clustering by optimal selection of cluster centroids in K-Means and K-Medoids algorithms,” Journal of Theoretical and Applied Information Technology, vol.98, no.18, pp3807-3814, 2020
[20] B. H. Husain and S. R. M. Zeebaree, “Improvised distributions framework of Hadoop: A review,” International Journal of Science and Business, vol.5, no.2, pp31-41, 2021
[21] O. Azeroual and R. Fabre, “Processing big data with Apache Hadoop in the current challenging era of COVID-19,” Big Data and Cognitive Computing, vol.5, no.1, pp12, 2021
[22] R. C. Taylor, “An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics,” BMC bioinformatics, vol.11, no.12, pp1-6, 2010
[23] H. Li, R. Liu, J. Wang and Q. Wu, “An enhanced and efficient clustering algorithm for large data using MapReduce,” IAENG International Journal of Computer Science, vol.46, no.1, pp61-67, 2019
[24] K. Kalia and N. Gupta, “Analysis of Hadoop MapReduce scheduling in heterogeneous environment,” Ain Shams Engineering Journal, vol.12, no.1, pp1101-1110, 2021
[25] G. Sanchez-Torres, A. Ceballos-Arroyo and S. Robles-Serrano, “Automatic measurement of fish weight and size by processing underwater hatchery images,” Engineering Letters, vol.26, no.4, pp461-472, 2018
[26] Y. Meng, J. Liang, F. Cao and Y. He, “A new distance with derivative information for functional K-Means clustering algorithm,” Information Sciences, vol.463, pp166-185, 2018
[27] V. Bindhu and G. Ranganathan, “Hyperspectral Image Processing in Internet of Things model using Clustering Algorithm,” Journal of ISMAC, vol.3, no.2, pp163-175, 2021