Hindawi
Complexity
Volume 2021, Article ID 9446653, 10 pages
https://doi.org/10.1155/2021/9446653

Research Article

k-Means Clustering Algorithm and Its Simulation Based on Distributed Computing Platform

Chunqiong Wu,[1,2] **Bingwen Yan**,[1,2] **Rongrui Yu**,[1,2] **Baoqin Yu**,[1,2] **Xiukao Zhou**,[1,2] **Yanliang Yu**,[1,2] **and Na Chen**[2,3]

[1]*Business College, Yango University, Fuzhou, Fujian Province 350015, China*
[2]*Big Data Business Intelligence Engineering Research Center, Fujian University, Fuzhou, Fujian Province 350015, China*
[3]*Software Engineering Institute, Lanzhou Institute Technology, Lanzhou, Gansu Province 730050, China*

Correspondence should be addressed to Chunqiong Wu; cqwu@ygu.edu.cn

Received 29 April 2021; Revised 17 May 2021; Accepted 10 June 2021; Published 19 June 2021

Academic Editor: Zhihan Lv

At present, the explosive growth of data and the mass storage state have brought many problems such as computational complexity and insufficient computational power to clustering research. The distributed computing platform through load balancing dynamically configures a large number of virtual computing resources, effectively breaking through the bottleneck of time and energy consumption, and embodies its unique advantages in massive data mining. This paper studies the parallel *k*-means extensively. This article first initializes random sampling and second parallelizes the distance calculation process that provides independence between the data objects to perform cluster analysis in parallel. After the parallel processing of the MapReduce, we use many nodes to calculate distance, which speeds up the efficiency of the algorithm. Finally, the clustering of data objects is parallelized. Results show that our method can provide services efficiently and stably and have good convergence.

1. Introduction

Data consumption and services have become the mainstream of today's information age [1]. Because of the huge amount of data and the complexity of iterative calculations in operation, traditional computing models have already become more difficult to deal with exponentially increasing data volumes. Therefore, the clustering algorithm for distributed cluster platforms has become an urgent problem to be solved [2, 3].

However, data analysis and knowledge discovery face greater challenges. Data mining usually uses algorithms to find the deep meaning hidden under the explicit features from massive data [4]. Most of the existing big data platforms are built based on distributed computing and distributed storage components. The computing resources of the big data platform are used to support big data analysis; most of the common practices are to parallelize the algorithm [5, 6]. It can be seen that in a big data environment, the distributed parallel computing framework can optimize related data-mining algorithms, parallelize them in the distributed computing environment, and provide more time for big data analysis and knowledge discovery analytical services to expand the scope and timeliness of big data applications.

Whether it is traditional data mining or data analysis in a big data environment, clustering, as a basic process of automatically categorizing unknown data, can be used in the data preprocessing stage as well as in data-mining processing. However, in the big data environment, cluster analysis faces many challenges. Some of these challenges are inherent to the clustering algorithm, while others are caused by the complex data environment. These challenges have brought new difficulties to cluster analysis in the big data environment, including the ability to handle diversified data types, ultra-high-dimensional data, and uneven data; the iterative execution efficiency of clustering algorithms; the algorithm extensible ability and clustering effect evaluation model; and many other issues. The *k*-means [7] can handle the clustering problem.

In summary, in a big data environment, data has characteristics such as massiveness, sparseness, and high dimensionality. Moreover, the big data processing platform based on the distributed system provides abundant computing and storage resources for the processing of massive information. How to effectively use the computing power of the distributed parallel computing framework to effectively improve the mining effect of traditional data mining algorithms and provide more timely data analysis services in the complex big data environment has become an urgent problem to be solved. In this paper, in view of many problems in *k*-means, combined with the MapReduce, the optimization strategy of *k*-means is studied. How to effectively apply traditional data mining algorithms to data analysis in a big data environment has universal reference significance. Our method was improved in three directions. We show this in detail in Section 3. The results prove that our method can provide accurate matching rate and real-time performance.

2. Related Technology

With the development of the Internet in recent years, the information that people can access has also increased exponentially. How to obtain knowledge from massive amounts of information is one of the current research focuses on computer information theory. As an important branch of data mining, clustering has gradually attracted widespread attention in recent years. Compared with other data mining methods, clustering has the advantage of not requiring prior knowledge, and knowledge can be obtained based on the natural distribution of data [8]. Clustering algorithms are divided into types based on partition, density, stratification, grid, and model.

Cluster analysis can divide the data set into several clusters [9]. The *k*-means is suitable for data sets with large amounts of data and high feature dimensions, and its dependence on data is low. Therefore, *k*-means has become a widely used clustering method [10]. However, *K* of the traditional *k*-means that needs to be determined in advance when it is initialized is determined only by the experience of the developer, and such subjectivity will affect the clustering efficiency and the credibility of the results [11]. The random selection of the initial clustering center will cause the instability of the clustering results [12, 13].

In recent years, based on the big data platform, there has been a lot of research work to implement the traditional data mining algorithm in parallel on the distributed platform and optimize the algorithm according to actual needs. Therefore, in response to the problem of selecting the initial center, Kumar et al. [14] performed a *k*-means analysis on the massive book circulation data on the Hadoop platform. From the beginning of *k*-means, the initial centroid selection was improved, and MapReduce was used to complete the parallel design of *k*-means for clustering book circulation data. Geng and Zhang [15] provided a data-mining method to solve the flood of weblog information. The algorithm was implemented on the Hadoop platform and used to analyze huge weblog information, which was verified from the three aspects such as effectiveness, speedup, and optimization rate. Xu and Ma [16] designed parallelized Bayes and Canopy algorithms based on the Hadoop platform and made a comparative analysis on the efficiency and scalability of classification and clustering algorithms. In order to solve the uncertainty and contingency of the selection of the central *K* value better, Yang et al. [17] made a density-based method. It can effectively eliminate orphan points and parallelize them. In response to the problem of too many iterations of the *k*-means clustering algorithm, Gopalani et al. [18] based on the *k*-means algorithm's characteristics of too many iterations and too low execution efficiency proposed a distributed computing framework based on Spark and applied it in the parallel algorithm of *k*-means text clustering. Then, according to RDD, *k*-means requirements for complex operations that need to be iterated are solved. Yu et al. [19] applied the MPI parallel computing framework to the wavelet clustering algorithm and proposed the MPI-wave cluster algorithm. Cui et al. [20] proposed a parallel genetic *k*-means to improve the efficiency of the overall operation.

However, *k*-means randomly selecting the initial center will lead to a local optimum. Values are unstable and iterative and time-consuming. In order to overcome the above problems, Sardar and Ansari [21] used a MapReduce computing framework combined with the *K*-selection sorting algorithm for parallel sampling to improve sampling efficiency and adopted a sample-based preprocessing strategy to obtain the initial center point to obtain a higher accuracy rate. Zhao et al. [22] designed map and reduce functions to realize the parallelization of the *k*-means algorithm. Xia et al. [23] proposed *k*-means local optimality. The algorithm uses the intersection between the subclusters of different clustering results to construct the weighted connected graph of the subclusters and then merges the subclusters through their connectivity. This algorithm improves the accuracy and efficiency of clustering to a certain extent, but because the algorithm does not have enough grasp of the clustering elements, the clustering accuracy still needs to be improved. Jin et al. [24] proposed a *k*-means initial clustering center selection algorithm based on optimal partitioning. The algorithm first divides the data samples and then determines the initial cluster centers according to the characteristics of the sample distribution. This algorithm improves the accuracy and efficiency of clustering, but for ultra-high-dimensional sample spaces, the algorithm will increase the number of recursions, making the calculations too complex and reducing the efficiency.

In summary, *k*-means has achieved results on how to select the initial center *K* value and reduce the number of iterations [25–30]. However, due to the blindness of the initial center *K* value selection, the cluster number needs to be determined in advance, and there are problems such as local optimization. Therefore, on the basis of the existing work, we conducted research on the more efficient and accurate *k*-means to bring sparseness and high latitude.

3. Methods

3.1. Distributed Computing Platform. Map reduce is one of the parallel programming models. Map reduce programs are

often used to parallelize massive amounts of data. The design idea is to convert the problem of divide and conquer, which is usually the processing of flood data sources, into the processing of multiple small data sources at the same time [31–33]. Finally, the intermediate results of each parallelization process are summarized to obtain the final result. It is one of the core distributed computing models and also a simple and easy-to-use distributed programming model. Therefore, this paper selects MapReduces as the distributed computing platform (Figure 1).

There are four entities at the top of the whole model, and the client is mainly responsible for submitting jobs to the MapReduce framework. JobTracker is solely responsible for scheduling the operation of the job. TaskTracker is responsible for running the input slice data and executing specific tasks. The distributed file system (HDFS) provides actual storage services and is used to share the resources required for operations with all nodes.

3.1.1. Traditional k-Means Algorithm. As a typical algorithm for calculating clusters, k-means is relatively efficient [34–36]. Therefore, it is used on a large scale in both theoretical research and actual production, with high status and influence; our method follows the methods of Li and Li [37].

To facilitate the description, this paper introduces a symbol $D = \{y_i \in R^n, \quad i = 1, \ldots, n\}$. The symbol $\{k_1, \ldots, k_k\}$ represents K cluster centers. The symbol $\{c_1, \ldots, c_k\}$ represents K different classes. The distance between data is as follows:

$$\operatorname{dis}(y_1, y_n) = \lim_{\lambda \longrightarrow 0} \sqrt{\sum_i \left(y_i(p), y_{i+1}(p)\right)^2}, \lambda = \max\left\{\left|y_{i+1}(p) - y_i(p)\right|\right\}. \tag{1}$$

The center point can be defined using the following equation:

$$k_j = \lim_{n \longrightarrow \infty} \frac{\sum y_i(p)}{n_j}, \tag{2}$$

where n_j refers to the number of same class.

The convergence flag can use the following formula to compute

$$l = \sum \sum \lim_{\lambda \longrightarrow 0} \sqrt{\sum_{i=1}^{n-1} \left(y_i(p), y_{i+1}(p)\right)^2}. \tag{3}$$

k-means [38, 39] is an iterative solution. We set cluster value K. Then the clustering center is constantly updated. However, there are still many problems with the process:

(1) Due to the influence of initial value and outliers, the results are not stable every time

(2) easy to converge to the local optimal solution

(3) The number of clusters needs to be preset

(4) Clustering center U does not necessarily belong to the data set

(5) k-means is easily affected by noise due to the use of the L2 distance function

In order to solve these problems, we have improved k-means.

3.1.2. Parallel Random Sampling. Traditional k-means uses all the data for clustering. This process takes a lot of time. Therefore, we first preprocess the data sources to reduce the amount of data used during the algorithm operation, thereby reducing the time consumption of the algorithm.

At present, there are two kinds of random sampling methods. One is traversal sampling, and the other is byte offset sampling. The characteristic of traversal is that the initial data is still selected in the sampling process without any operation, which is time-consuming, especially if the data set is large. It is a random sample, but the operation is still huge. Therefore, this method cannot be used for the data in this article. Although byte migration can process large quantities of data, the algorithm is not efficient.

In order to obtain more efficiency, we propose a parallel random sampling method on the basis of the above methods. Because the method is to operate on the parallel unit, it is more efficient and less time-consuming. The sampling procedure is as follows:

(1) First, assign values to all data. At the same time, the unified processing is carried out in the format of keyword.

(2) Sort the above data in order from the largest to the smallest.

(3) Select the smallest data after sorting as the center point of the initialization class cluster. The calculation formula is as follows:

$$\begin{aligned} \text{pre\_1} &= \sum_{b=1}^{n} \operatorname{dis}(d_{a1}, d_b), \\ \text{pre\_2} &= \sum_{a=1}^{n} \sum_{b=1}^{n} \operatorname{dis}(d_a, d_b), \\ \text{pre} &= 0.35 \cdot \text{pre\_1} + 0.65 \cdot \text{pre\_2}. \end{aligned} \tag{4}$$

3.1.3. Distance of Parallelization. Traditional k-means achieves the purpose of clustering by carrying out the cyclic calculation on all the data. However, this process takes a lot of time. Therefore, parallelizing it is a very good approach. The way parallelization is done is to take advantage of the independence of data from one data to another.

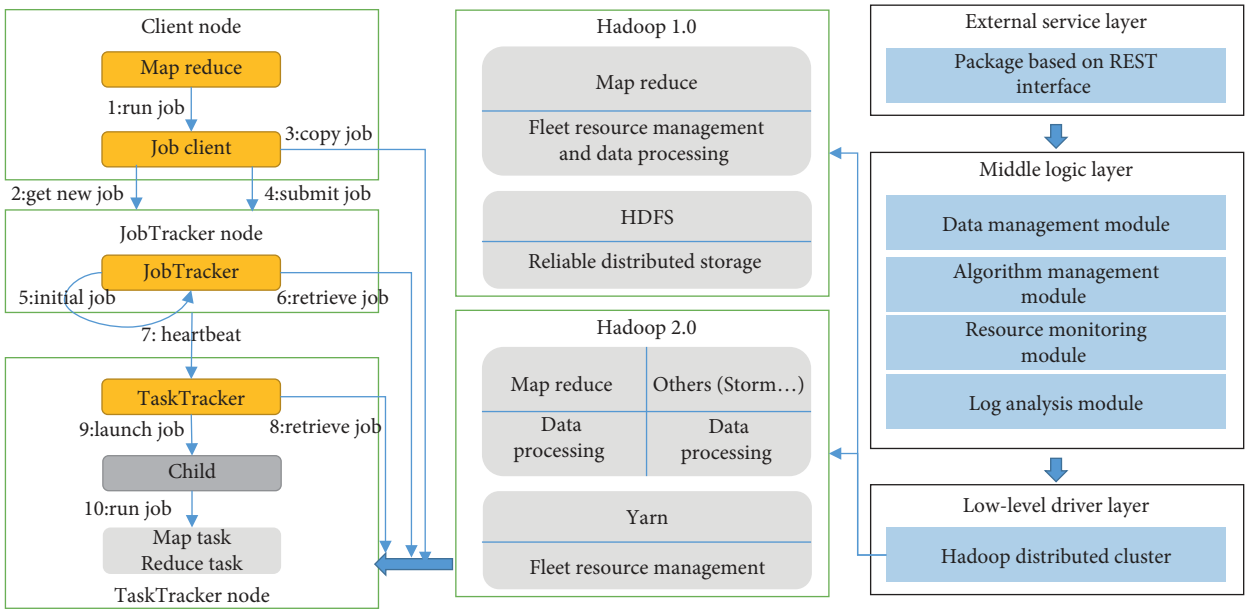

Figure 1: Operation flow chart of MapReduce.

The map function in MapReduce is used to map the data. First, the independence of data is used to map the data to different Reducer units in the form of keywords. Then, parallel clustering computation is carried out in these different units.

In this way, the independence between data can be effectively utilized. At the same time, the efficiency of clustering can be accelerated using multiunit simultaneous processing.

3.1.4. Parallelization of Data Object Clustering. Generally speaking, data can be mapped to the Reducer of their respective class clusters by the Mapper process according to the length of the distance. In order to adaptively obtain the corresponding Reducer of each class cluster, we set the value of parallelism as k. In the Reducer, the value of the initial center point of our method is calculated using random sampling and the sum of squares of Euclidean distance. Sort them, select the smallest value as the center point of the next round, and so on. The improved parallel structure is shown in Figure 2.

First, each cluster has its own Reducer. Therefore, we need to execute the parallelization strategy for all data. Then, we take all chosen data to process and set the focus dot of the class cluster. Then, Euclidean distance between many data and the current focus dot is meant. Finally, the minimum dot of the sum of squares is selected as the new center dot.

We adopt characteristics of MapReduce to optimize the calculation of minimum Euclidean distance. Because the comparison function and the key in the comparison function can calculate the value between the keys for sorting processing. The sort mechanism not only simplifies the problems faced by the algorithm but also facilitates the computing power of distributed clusters and speeds up the execution efficiency of selecting the minimum Euclidean distance sum (Figure 3). Combining the above features, k-means uses element number as value to realize the Euclidean distance sorting function. The structure of our method is shown in Figure 4. Our method's complexity is $O(n)$.

4. Results and Discussion

4.1. Experimental Data. To imitate the real environment, a total of 6 PCs are used.

Hardware configuration: AMD Athlon (TM) X4 with 3.10 GHz CPU, 4 GB memory, and 500 GB disk space.

Software environment: Linux operating system is CentOS, and JAVA, ZooKeeper, Hadoop, and dBase are also used.

To test the performance, a data set is used, and the modified data set is divided into 5 groups. Each group has 50 samples, which have 4 attributes. This experiment generated a total of 5 parts. The detail of the random data set can be seen in Table 1.

4.1.1. Convergence Performance. The effectiveness of our method is verified by comparing the convergence between the traditional k-means method and the proposed method. We use the same data set in the single machine experiment environment and calculate the number of iterations when the convergence state is reached. Two methods are used in a single computer environment. At the same time, data set 1 is used as the raw data set. The result is shown in Figure 5. Our method achieves fewer iterations in a single computer environment. It is further explained that the reason why our method converges fast is that preprocessing can have focus dot better than traditional k-means.

4.1.2. Correct Rate Comparison. To verify the accuracy of the primitive k-means, density k-means, MPI-wave, and our

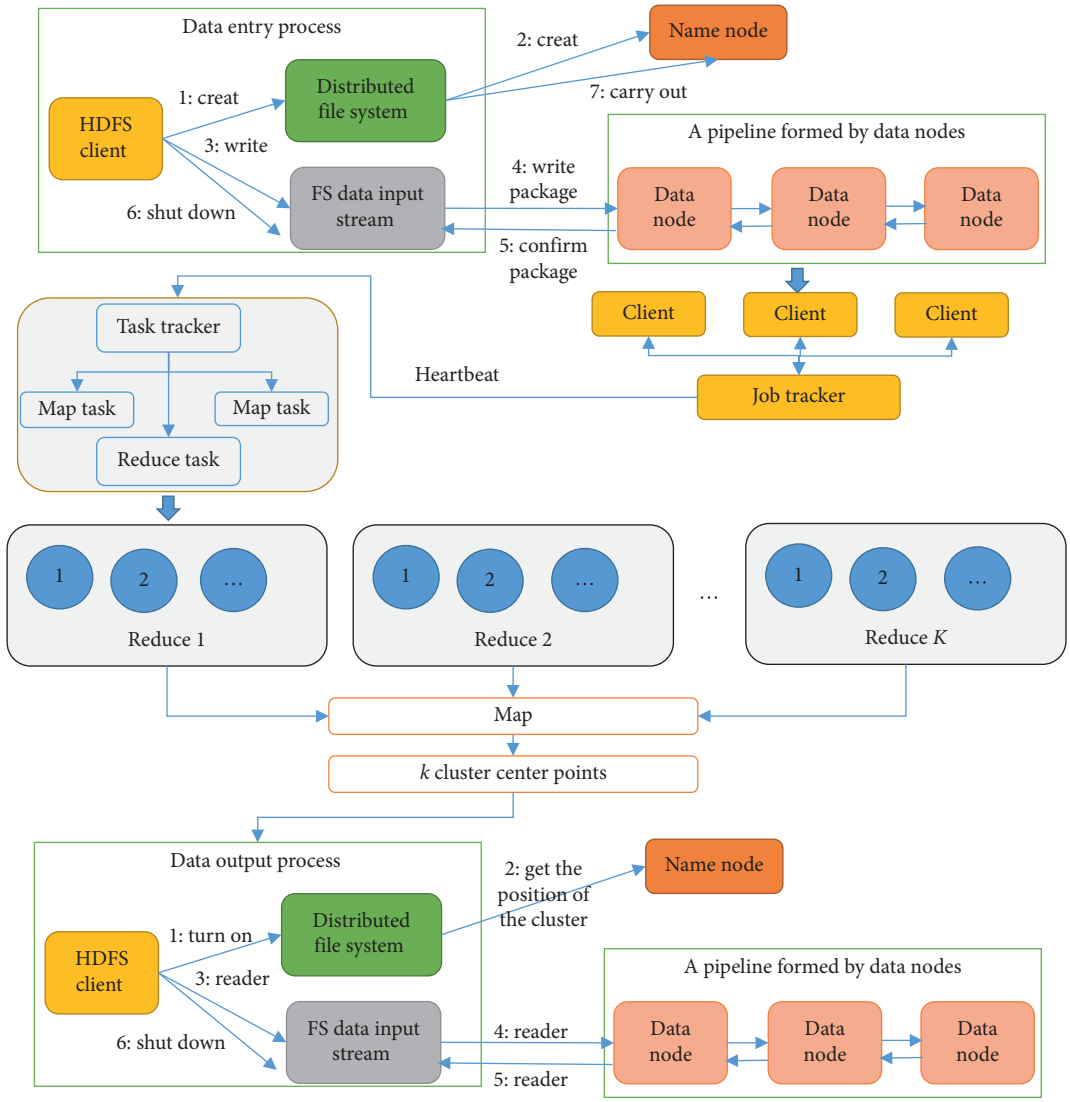

Figure 2: Reducer parallelization structure.

method, we design the following experiments. The effects of the four methods are shown in Figures 6 and 7 .

Accuracy and recall are higher than the comparison as the size of the data continues to expand on different data sets. Primitive k-means is prone to produce local optimal results, so its accuracy is significantly lower than other comparison algorithms. The k-means algorithm selects the two points with the largest distance as the two initial points selected for the first time. The initial point selection is too fixed, which is not the best clustering center that easily leads to inaccurate clustering results. Compared with the MPI-wave cluster algorithm, when there are 100 to 150 data sets, the overall algorithm has a trend of decreasing accuracy. The reason is that the k-means algorithm has defects in its algorithm, and there is a local optimum. As the amount of data continues to increase, the clustering effect will improve, and the local optimal situation will be improved. This is because our method effectively reduces the randomness of the selection of the initial center point by removing the interference data at first. Therefore, the accuracy of clustering can be improved.

4.1.3. Operational Efficiency Comparison. In order to verify the advantages of MapReduce distributed clusters over Spark in iterative computing, this experiment was performed on MapReduce and Spark clusters. The primitive k-means and our method are in the same machine configuration environment. The running time of the two with the same number of computing nodes is compared to reflect their respective operating performance.

In Figure 8, in primitive k-means and our method, the running time of Spark is slightly higher than that of MapReduce. However, as the amount of data increases,

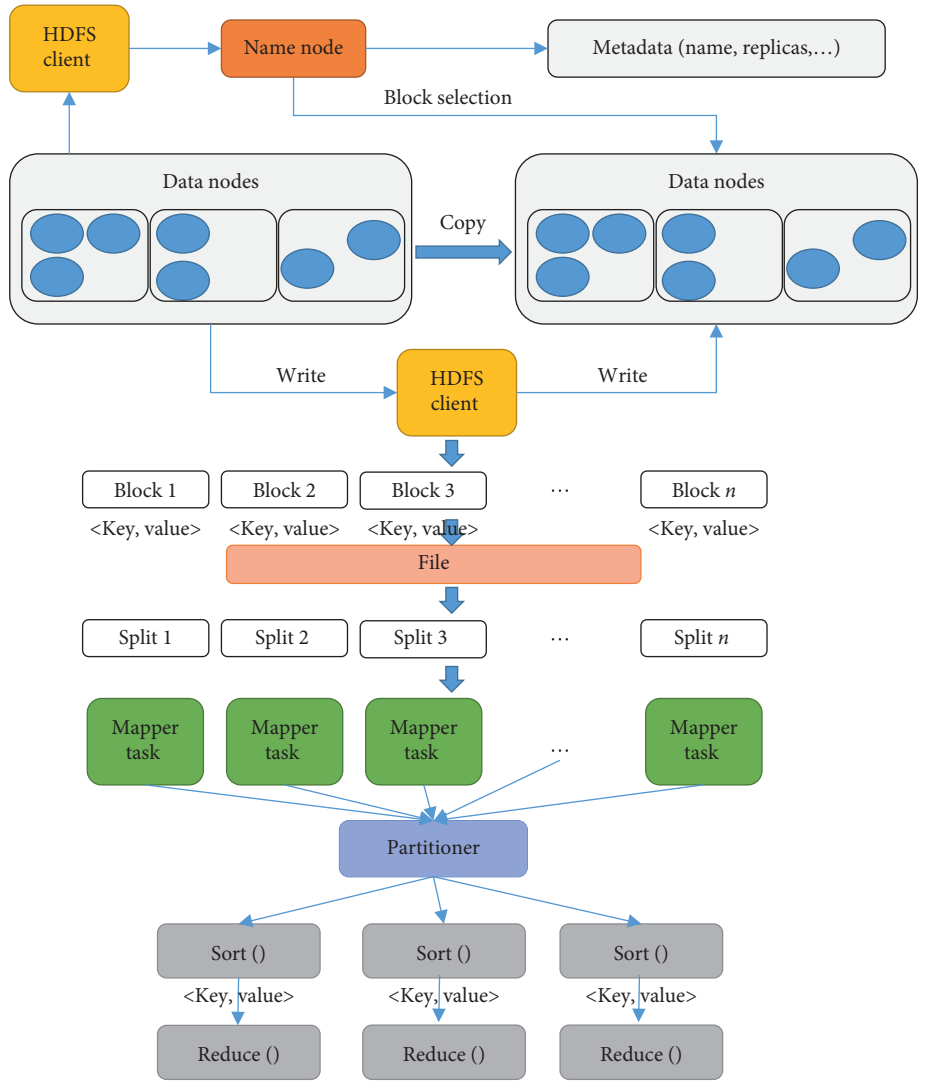

Figure 3: Sort mechanism of MapReduce.

starting from 150 pieces of data, the running time of the same amount of data is shorter on the MapReduce platform than on the Spark platform. Moreover, as the scale of data processing increases, MapReduce has a higher processing efficiency than Spark and a more obvious upward trend. This is because Spark must revisit HDFS when it reads a set of data objects into local memory during each iteration of the calculation process. Map reduce is based on memory calculations, which greatly reduces the time overhead rate of data I/O in iterative calculations. Therefore, in the MapReduce platform environment, the running time is significantly reduced compared with the Spark platform.

4.1.4. Cluster Environment Speedup Verification. Our algorithm uses a parallel structure. We use the speedup ratio to verify the real-time performance. By testing the acceleration of our algorithm, the real-time performance of our algorithm is verified. Its calculation formula is as follows:

$$\text{ratio} = \frac{\text{Time}_{\text{single}} + 1}{\text{Time}_{\text{multi}} - 1}, \tag{5}$$

where $\text{Time}_{\text{single}}$ is the run time in one unit, and $\text{Time}_{\text{multi}}$ is the run time in many units. The greater the value of the acceleration ratio is, the greater the efficiency can be effectively improved in the distributed cluster environment. Speedup experiment data is shown in Figure 9.

The acceleration ratio can increase with many data. This condition indicates that our method can improve accuracy. At the same time, it can be used in big data sets.

5. Conclusions

The random selection of the focus dot by k-means will lead to local optimization and unstable iteration time of clustering results. To overcome the problems, this paper proposes a k-means with improvement. First, random sampling is

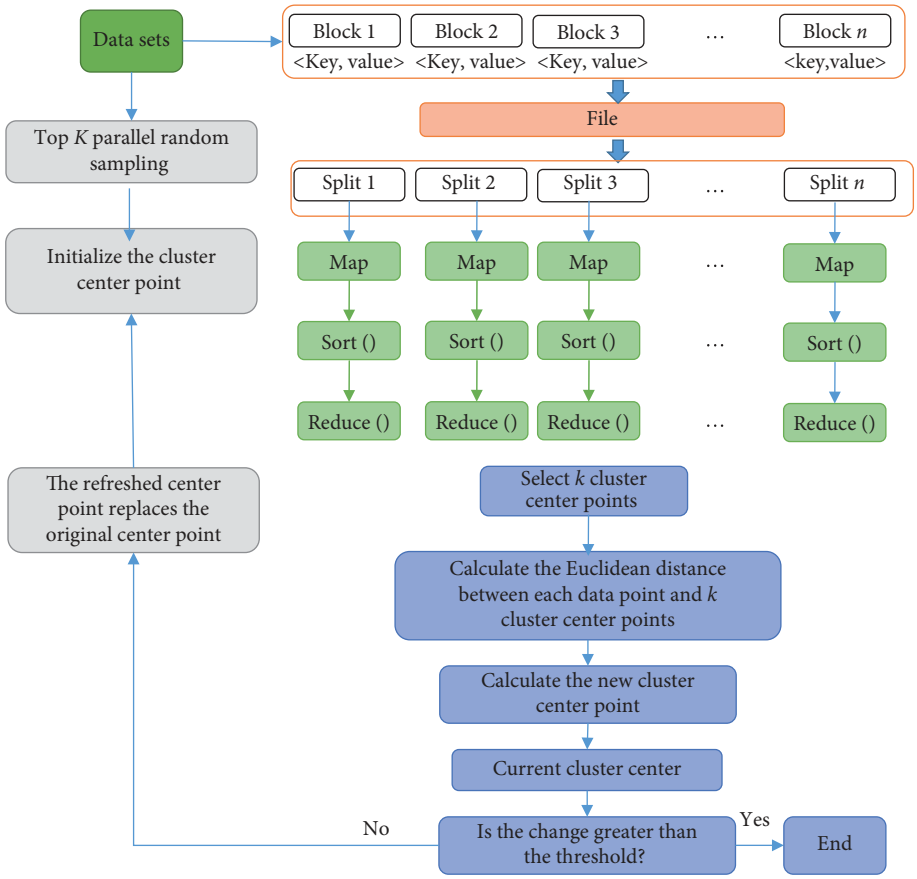

Figure 4: Structure diagram of our method.

Table 1: The data set we used.

Data sets	Size (MB)	Items	Dimension	Cluster center points
Data set 1	0.32	9,600	4	5
Data set 2	112	9,600,000	4	5
Data set 3	401	28,800,000	4	5
Data set 4	1,421	67,100,000	4	5
Data set 5	3,267	173,560,000	4	5

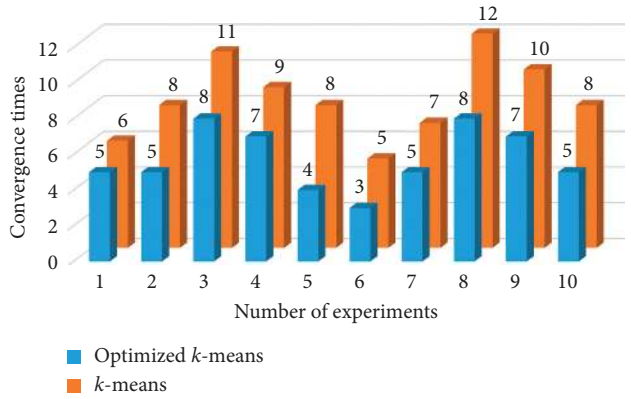

Figure 5: Convergence speed comparison.

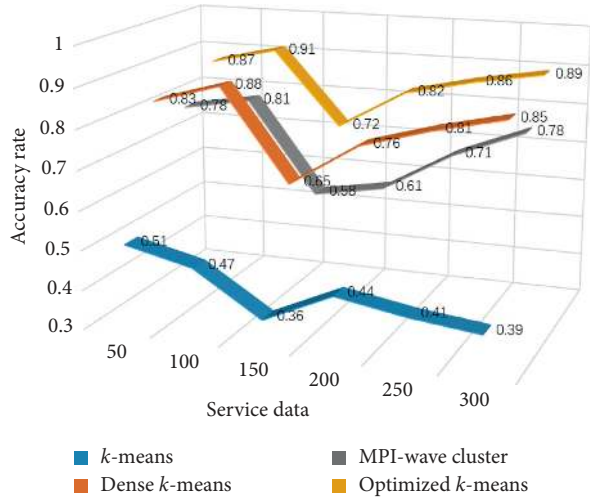

Figure 6: Accuracy rate comparison experiment.

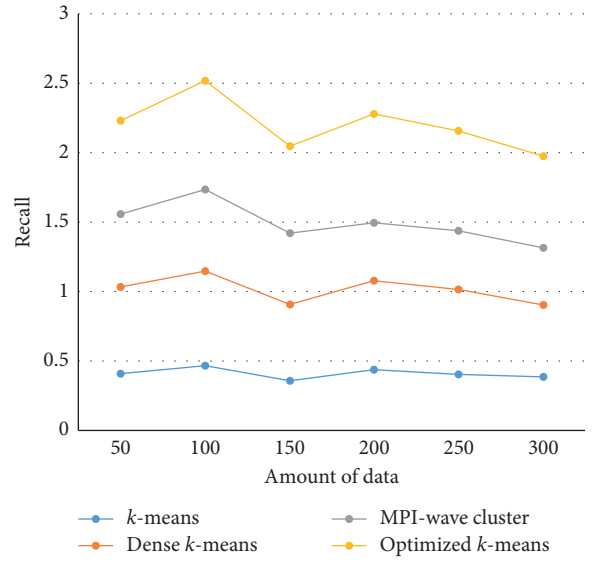

Figure 7: Recall rate comparison experiment.

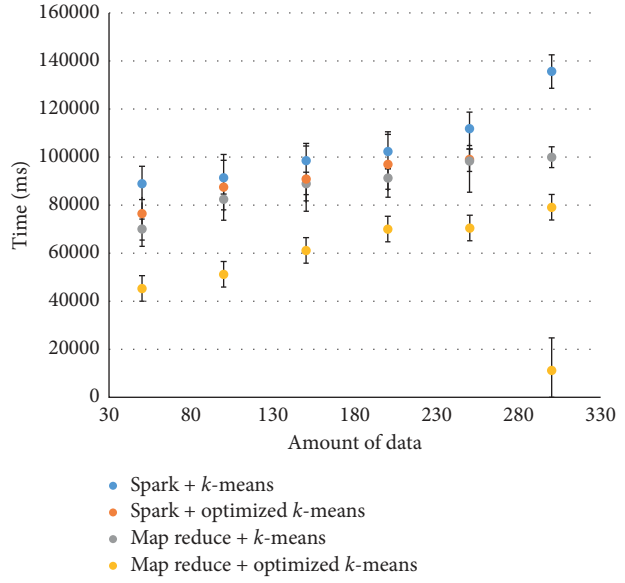

Figure 8: Time cost comparison experiment.

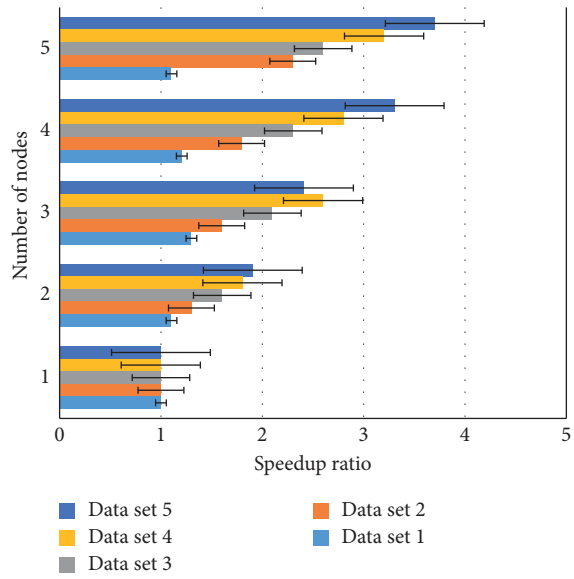

Figure 9: Speedup test.

initialized. Second, the distance of many data is calculated in parallel. Finally, data is clustered and parallelized. Results show that our method has high clustering accuracy.

Although our method can deal with large-scale data, it still has problems when dealing with high-dimensional data sets. Therefore, our next research plan is to further improve our algorithm so that it can adapt to high-dimensional data sets.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] W. Yi and J. Yan, "Energy consumption and emission influences from shared mobility in China: a national level annual data analysis," *Applied Energy*, vol. 277, Article ID 115549, 2020.

[2] S. G. Anton and A. E. Afloarei Nucu, "The effect of financial development on renewable energy consumption. A panel data approach," *Renewable Energy*, vol. 147, pp. 330–338, 2020.

[3] P. Pei, Z. Huo, O. S. Martínez, and R. G. Crespo, "Minimal green energy consumption and workload management for data centers on smart city platforms," *Sustainability*, vol. 12, no. 8, p. 3140, 2020.

[4] T. Enokido and M. Takizawa, "The power consumption model of a server to perform data access application processes in virtual machine environments, advanced information networking and applications," in *Proceedings of the International Conference on Advanced Information Networking and Applications*, pp. 184–192, Springer, Toronto, ON, Canada, May 2020.

[5] Q. Zhou, S. Guo, and H. Lu, "Falcon: addressing stragglers in heterogeneous parameter server via multiple parallelism," *IEEE Transactions on Computers*, vol. 70, no. 1, pp. 139–155, 2020.

[6] K. G. Miller, R. P. Lee, A. Tableman et al., "Dynamic load balancing with enhanced shared-memory parallelism for particle-in-cell codes," *Computer Physics Communications*, vol. 259, Article ID 107633, 2021.

[7] K. P. Sinaga and M.-S. Yang, "Unsupervised *K*-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020.

[8] A. K. Sangaiah, A. E. Fakhry, and M. Abdel-Basset, "Arabic text clustering using improved clustering algorithms with dimensionality reduction," *Cluster Computing*, vol. 22, no. 2, pp. 1–15, 2019.

[9] S. SoukainaMjahed, K. Bouzaachane, A. Taher Azar, S. El Hadaj, and S. Raghay, "Hybridization of fuzzy and hard semi-supervised clustering algorithms tuned with ant lion optimizer applied to Higgs boson search," *Computer Modeling in Engineering & Sciences*, vol. 125, no. 2, pp. 459–494, 2020.

[10] J. J. Jay, J. Eblen, and Y. Zhang, "A systematic comparison of genome-scale clustering algorithms," *Bmc Bioinformatics*, vol. 13, no. 10, pp. 1–12, 2012.

[11] M. S. Yang and K. P. Sinaga, "A feature-reduction multi-view *k*-means clustering algorithm," *IEEE Access*, vol. 9, p. 1, 2019.

[12] J. Song, X. Li, and Y. Liu, "An optimized *k*-means algorithm for selecting initial clustering centers," *International Journal of Security and Its Applications*, vol. 9, no. 10, pp. 177–186, 2015.

[13] H. B. Zhou and J. T. Gao, "An improved initial clustering center selection method for *K*-means algorithm," *Advanced Materials Research*, vol. 1022, pp. 337–340, 2014.

[14] A. Kumar, M. Kiran, S. Mukherjee, and G. R. Prakash, "Verification and validation of MapReduce program model for parallel *K*-means algorithm on Hadoop cluster," *International Journal of Computer Applications*, vol. 72, no. 8, pp. 48–55, 2013.

[15] Y. Geng and L. Zhang, "*K*-means clustering algorithm for large-scale Chinese commodity information web based on Hadoop," in *Proceedings of the International Symposium on Distributed Computing & Applications for Business Engineering & Science*, IEEE, Guangxi, China, November 2016.

[16] J. Xu and B. Ma, "Study of network public opinion classification method based on naive bayesian algorithm in Hadoop environment," *Applied Mechanics and Materials*, vol. 519–520, pp. 58–61, 2014.

[17] L. Yang, W. Jiang, and H. Ji, "Automatic brain tumor segmentation using cascaded FCN with DenseCRF and *K*-means," in *Proceedings of the 2019 IEEE/CIC International Conference on Communications in China (ICCC)*, IEEE, Changchun, China, August 2019.

[18] S. Gopalani, R. Arora, and S. Gopalani, "Comparing Apache Spark and MapReduce with performance analysis using *K*-means," *International Journal of Computer Applications*, vol. 113, no. 1, pp. 8–11, 2015.

[19] Q. Yu, S. Zhang, and X. Kong, "Converted-wave Kirchhoff prestack time migration parallel computation based on MPI+CUDA," *Geophysical Prospecting for Petroleum*, vol. 52, no. 1, pp. 60–64, 2013.

[20] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data *K*-means clustering using MapReduce," *The Journal of Supercomputing*, vol. 70, no. 3, pp. 1249–1259, 2014.

[21] T. H. Sardar and Z. Ansari, "An analysis of distributed document clustering using MapReduce based *K*-means

algorithm," *Journal of The Institution of Engineers (India) Series B*, vol. 101, no. 2, pp. 1–10, 2020.

[22] M. Zhao, J. Liu, and Z. Zhang, "A scalable sub-graph regularization for efficient content based image retrieval with long-term relevance feedback enhancement," *Knowledge-Based Systems*, vol. 212, no. 1, Article ID 106505, 2020.

[23] C. Xia, J. Hua, and W. Tong, "Distributed *K*-Means clustering guaranteeing local differential privacy," *Computers & Security*, vol. 90, pp. 101699.1–101699.11, 2020.

[24] T. K. Jin, J. Song, and C. S. Ah, "Optically readable waveguide-integrated electrochromic artificial synaptic device for photonic neuromorphic systems," *ACS Applied Electronic Materials*, vol. 2, no. 7, pp. 2057–2063, 2020.

[25] A. A. Aldino, D. Darwis, and A. T. Prastowo, "Implementation of *K*-means algorithm for clustering corn planting feasibility area in south lampung regency," *Journal of Physics: Conference Series*, IOP Publishing, vol. 1751, no. 1, pp. 012–038, 2021.

[26] J. Rejito, A. Atthariq, and A. S. Abdullah, "Application of text mining employing k-means algorithms for clustering tweets of Tokopedia," *Journal of Physics: Conference Series*, IOP Publishing, vol. 1722, no. 1, pp. 012–019, 2021.

[27] C. Li, F. Kulwa, J. Zhang, Z. Li, H. Xu, and X. Zhao, "A review of clustering methods in microorganism image analysis," *Advances in Intelligent Systems and Computing*, pp. 13–25, 2021.

[28] X. Chen and Y. Yang, "Hanson–Wright inequality in Hilbert spaces with application to $ *K* $-means clustering for non-Euclidean data," *Bernoulli*, vol. 27, no. 1, pp. 586–614, 2021.

[29] M. B. Gesicho, M. C. Were, and A. Babic, "Evaluating performance of health care facilities at meeting HIV-indicator reporting requirements in Kenya: an application of *K*-means clustering algorithm," *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, pp. 1–18, 2021.

[30] M. Ghadiri, S. Samadi, and S. Vempala, "Socially fair *k*-means clustering," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 438–448, Toronto, ON, Canada, March 2021.

[31] W. Bo, Z. B. Fang, L. X. Wei, Z. F. Cheng, and Z. X. Hua, "Malicious URLs detection based on a novel optimization algorithm," *IEICE Transactions on Information and Systems*, vol. E104.D, no. 4, pp. 513–516, 2021.

[32] W. Wang, F. Xia, H. Nie et al., "Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3567–3576, 2021.

[33] K. Sim, J. Yang, W. Lu, and X. Gao, "Blind stereoscopic image quality evaluator based on binocular semantic and quality channels," *IEEE Transactions on Multimedia*, p. 1, 2021.

[34] S. Yang, J. Wang, X. Hao et al., "BiCoSS: toward large-scale cognition brain with multigranular neuromorphic architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 2021, Article ID 3045492, 15 pages, 2021.

[35] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, "Accurate and fast URL phishing detector: a convolutional neural network approach," *Computer Networks*, vol. 178, Article ID 107275, 2020.

[36] S. Qi, Y. Zheng, X. Chen, and W. Wei, "Ants can carry cheese: secure and private RFID-enabled third-party distribution," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.

[37] X. Li and D. Li, "An improved *K*-means parallel algorithm based on cloud computing," *Communications in Computer and Information Science*, vol. 901, pp. 394–402, 2018.

[38] S. Yadav, R. Mohan, and P. K. Yadav, "Task allocation model for optimal system cost using fuzzy *C*-means clustering technique in distributed system," *Ingénierie des systèmes d information*, vol. 25, no. 1, pp. 59–68, 2020.

[39] A. Flores-Quiroz and K. Strunz, "A distributed computing framework for multi-stage stochastic planning of renewable power systems with energy storage as flexibility option," *Applied Energy*, vol. 291, no. 1, Article ID 116736, 2021.