Eng Int Syst (2021) 6: 411–418
© 2021 CRL Publishing Ltd

Engineering
Intelligent Systems

K-Means Parallel Algorithm of Big Data Clustering Based on Mapreduce PCAM Method

Yongyi Li[1,2,*], **Zhongqiang Yang**[1,2], **Kaixu Han**[1,2]

[1] *College of Electronic and Information Engineering, Qinzhou University, Qinzhou 535011, China*
[2] *Qinzhou Key Laboratory of Big Data Resource Utilization, Qinzhou 535011, China*

With the increase in the offshore industry in the Beibu Gulf, data clustering has become an important task of intelligent ocean monitoring. However, the traditional K-means algorithm is not suitable for large-scale marine data. Aiming at the characteristics of marine big data, a parallel K-means algorithm based on MapReduce big data clustering is proposed. First, according to the characteristics of the MapReduce framework, a partition, communication, combination and mapping model is established. A parallel K-means algorithm based on MapReduce big data clustering is then designed, and the execution process of the algorithm is analyzed. Finally, through data and experimental analysis, it is demonstrated that the MR K-means parallel algorithm reduces the time and space complexity and the data point missing rate compared with the traditional algorithm.

Keywords: Clustering, K-means, Parallel, MapReduce, PCAM

1. INTRODUCTION

Data clustering is an important part of information processing, and regular data can be clustered by general clustering algorithms. However, big data has large data volumes, complex data types, heterogeneousness, and highly dynamic and spatiotemporal characteristics. The use of traditional clustering algorithms can lead to excessive complexity, especially in monitoring data. The Beibu Gulf is rich in marine resources, leading to many industries in the region flourishing, including tourism, marine production, logistics, transportation, leisure sports, food culture, folk customs and crafts. Ocean data analysis is complicated with the volume of big data in ocean monitoring. The traditional data mining technology method using a single node has become a bottleneck in big data processing, which has lead to an imperfect information chain and resource isolation. In order to enable big data to serve other industries, effective clustering is the key to the problem. None of the many existing clustering algorithms can be widely applied to reveal the data structure of a variety of data sets. In general, we can choose different clustering algorithms according to data types, the purpose of clustering and its application. Clustering algorithms generally include five categories: partition clustering, hierarchical clustering, density-based clustering, grid-based clustering and model-based clustering (Yu, 2017). The general K-means partition clustering algorithm has the characteristics of being simple and efficient, but it needs large amount of memory and cannot solve the problem of a large amount of data. The K-means algorithm is one of the most common clustering algorithms (Jin, 2014). When the number of clusters and iterations are obviously smaller than the number of data points, K-means is effective, but it cannot guarantee the scalability and effectiveness of clustering for the processing of large data sets which leads to the drastic increase of time complexity. Moreover, when the clusters are non-spherical or have different sizes or densities the K-means algorithm finds it difficult to detect the corresponding clusters. Hierarchical clustering mainly uses hierarchically organized data, such as

*Corresponding Author e-mail: qzxylyy@163.com

document data with topic and subtopic structures. It cannot solve the big data clustering problem with complex types, strong dynamics and time series characteristics. Therefore, it is possible to transform the single node computing method into parallel computing to solve the problem of a large data volume and high complexity (Dayong et al., 2019; Meisam et al., 2019; Ou et al., 2019).

Parallel processing is a way to improve the utilization of computing resources, with the purpose of improving the execution performance of programs. In many technological fields, parallel processing is applied to solve practical problems. The initial stage of parallel processing requires programmers to have strong parallel programming skills in hardware and architecture, and there are still higher requirements in the aspect of complex and complicated parallel control logic. In the age of big data, a cluster environment based on system framework has been developed. The constraints of parallel processing are changing; the technology core becomes the establishment and application of technology models. Therefore, it is feasible to apply distributed parallel processing technology in the field of big data computing.

In this study, on the basis of the K-means algorithm, the idea of parallel computing is introduced, and the multi node data processing is completed using the framework of MapReduce, and the solution of a MapReduce K-means algorithm is proposed. The aim is to reduce the calculation amount of big data, speed up calculations and improve the efficiency of arithmetic operations. Combining the theoretical analysis and experimental evaluation is of extreme significance.

2. DESIGN OF A PARALLEL ALGORITHM IN BIG DATA BACKGROUND

Using multiple processors to solve the same problem is the basic idea of parallel computing, that is, to decompose large problems into multiple small tasks, and several independent processors perform corresponding small tasks at the same time. Parallel computing can be accomplished by a computer system with multi processor functions, and a cluster of independent computers can be implemented in parallel computing through a certain connection. The master/slave node mode is the most common form of clustering, in which tasks are generally allocated by the master node to the slave node, and then returned to users after result summarization. Therefore, the essence of a parallel algorithm and its implementation steps are to solve the problem process with multiple nodes.

In general, problems that can be solved by parallel computing generally have the following characteristics:

(1) The work can be separated into discrete parts to facilitate solving them simultaneously.

(2) Multiple instructions can be executed at any time and in time.

(3) The time consumed in solving problems under multiple computing resources should be less than the time consumed using a single computing resource (Li, 2006).

2.1 Strategy and Technology of Parallel Algorithm Design

Parallelization of serial algorithms is one of the most commonly used design strategies. Partitioning, communication, agglomeration, and mapping (PCAM) design methodology includes four phases of design parallelism. This method maximizes the concurrency of algorithms and satisfies the scalability of the algorithms. It can improve the efficiency of the whole algorithm by optimizing the local stage.

2.1.1 Partitioning

Domain decomposition, also known as data partition, is performed by partitioning the big data to be processed into small and uniform data blocks. The commonly used domain decomposition methods include the divide and conquer method, the domain decomposition method based on numerical solution of partial differential equations, the red/black coloring method in solving equations. (Fang, 2011).

2.1.2 Communication

Communication is mostly used for information exchange and monitoring in parallel computing, where the usual form is the data dependency attached to tasks. The commonly used communication modes include global-global communication, structured/unstructured communication, static/dynamic communication, synchronous/asynchronous communication, etc. (Guo 2008).

2.1.3 Agglomeration

Combining small tasks into large tasks to reduce the number of tasks is aimed at reducing communication and computing consumption in order to improve efficiency. The usual agglomeration methods are surface-volume effect and repeated calculation.

2.1.4 Mapping

Each task is allocated to a node to improve the algorithm performance and reduce the total execution time of the algorithm; this mainly involves load balancing and task scheduling. Commonly used load balancing methods include recursive pair partitioning, local algorithm, probabilistic algorithm, and cyclic mapping. Commonly used task scheduling strategies mainly include the manager/employee model and the non-centralized model. In a large-scale problem, the manager/employee model can prevent the manager from becoming a bottleneck.

3. ANALYSIS OF K-MEANS PARALLEL ALGORITHM BASED ON MAPREDUCE

MapReduce is a distributed computing engine of the Hadoop platform. By constructing a Mapper and Reducer, programmers can ignore the parallel computing and scheduling of the

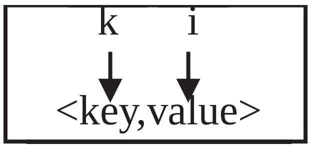

Figure 1 Correspondence of Communication Variables.

underlying program and concentrate on the specific execution tasks. The difficulty of writing parallel programs is reduced, and the writing efficiency of parallel programs is improved. Any programming language or script can be used to write MapReduce programs to achieve parallel computing strategy.

3.1 K-means parallel algorithm model based on MapReduce

(1) Let the data points set D be (Cui, 2012):

$$D = \{x_1, x_2, \ldots, x_n\} \tag{1}$$

From (1), the vector of the first i clustering data point is $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ir}\}$ where $i \le n$ and r is the number of attributes.

(2) Cluster center and distance model

c_j represents the j-th cluster, and the mean vector of the data points in c_j represents the cluster center of c_j:

$$m_j = \frac{1}{|c_j|} \sum_{x_i \epsilon c_j} x_i \tag{2}$$

Where $|c_j|$ represents the number of data points in c_j. The distance between data point x_i and cluster center m_j is:

$$\begin{aligned} \mathrm{dist}\left(x_i, m_j\right) &= x_i - m_j \\ &= \sqrt{\left(x_{i1} - m_{j1}\right)^2 + \left(x_{i2} - m_{j2}\right)^2 + \ldots + \left(x_{ir} - m_{jr}\right)^2} \end{aligned} \tag{3}$$

(3) Convergence condition

Convergence condition is minimizing the sum of square error:

$$\mathrm{SSE} = \sum_{j=1}^{K} \sum_{x \epsilon c_j} dist(x, m_j)^2 \tag{4}$$

Where k is the cluster number needed.

(4) Cluster center

Assume that the number of data points assigned to cluster c_j is n_j, the number of data points after clustering completes becomes:

$$n_j = n_j + 1 \tag{5}$$

According to (2), the next cluster center is obtained.

$$m_{j+1} = \frac{1}{|c_j|} \sum_{x_i \in c_j} x_i + \frac{1}{|c_j| + 1} s_j \tag{6}$$

(5) Division model

Before data is processed in parallel, the data set is first divided. Since big data has regularity and temporality, data can be grouped according to the first r attributes (such as time) (Luo, 2012). If the attribute has T values, the set of packets P is expressed by (7):

$$D_p = \{x_i | x_{ir} = x[p],\ \mathrm{p} \in [0, T]\} \tag{7}$$

Where $x[p]$ is a value of the r-th attribute.

In order to determine the corresponding relationship between nodes and data packets, array data $[k]$ can be used to save the data set sequence number to be processed by cluster node k. The division model is:

$$\begin{aligned} \mathrm{data}[k] = \Bigg\{ & i \mid x_i \in D_p, i = 1, 2, \ldots, n, k = 1, 2, \ldots, N, \\ & p \in \left[\frac{T}{N}(k-1), \frac{T}{N} \times k\right]\Bigg\} \end{aligned} \tag{8}$$

Where T is the number of groups, N is the number of nodes, and n is the number of data points.

(6) Communication model

The communication model realizes the functions of transmission, calculation and storage of each node from partitioned data blocks (Wang, 2010). The MapReduce system abstracts the communication function to the Map function. Generally, local operations are performed on data storage nodes, with the format of the function as <key, value>, where key is the node number, and value is used for storing data to be processed. Instead of storing data, the addresses of data are recorded in the form of arrays in partitions. Combining with (8), key is the communication interface variable, and the communication model is expressed as (9). The correspondence of communication variables is shown in Figure 1.

$$\begin{aligned} \mathrm{Input}(k, \mathrm{valuelist}) = \{\langle key, value\rangle \mid k \\ \to key, data[k] \to value\} \end{aligned} \tag{9}$$

(7) Combination model

After the Map stage, the resulting data is used as the input of Reduce. When repeated key values occur in the same node, the values are combined to reduce the size of the storage file. As the operation of MapReduce is affected by network bandwidth, the combine function can be used to reduce the data transmission between Map and Reduce as much as possible (Zhu, 2017). The main effect of this function is to reduce the number of intermediate result

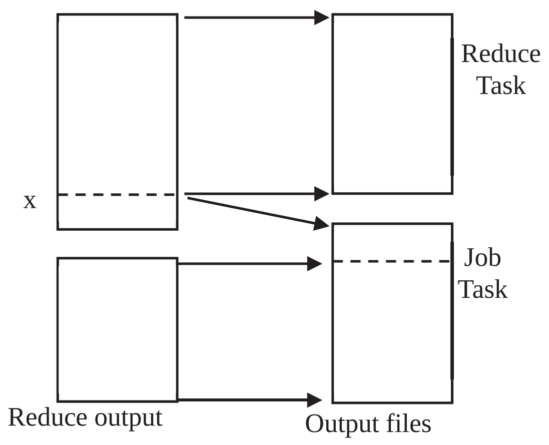

Figure 2 Task Scheduling Strategy.

<key, value> pairs, so that the system consumption can be reduced. The combination model is expressed as (10):

$$\text{Comb}(k) = \{\langle key, value\rangle \mid Input(k, valuelist) \rightarrow value, key = k\} \tag{10}$$

(8) Mapping model

After processing the data in the Reduce stage, the data results are stored in HDFS, and each result becomes a Reduce task (Li and Dong, 2012). The tasks are dynamically undetermined, which results in different sizes of the Reduce tasks. However, the block size of the HDFS distributed file system is fixed and finite. Therefore, two important problems need to be solved for Reduce output: load balancing and task scheduling. Assuming the HDFS block size is 64 MB, and the block name is F: output (k, value list) can be defined as a Reduce output function, where the value list is the output data list. According to the boundary policy, the scope of K is calculated and load balancing is achieved. The data set stored in F file block can then be expressed by (11):

$$\text{F}[k] = \left\{ valuelist(x) \mid x \leq count(valuelist), \sum_{i=1}^{x} du(i) \leq 64MB \right\} \tag{11}$$

Where count (valuelist) is the function of obtaining the data number, du (i) is the function of obtaining the data size, and x is the critical value of the boundary strategy. According to (11), the overall task scheduling strategy can be obtained. The data within the boundary is distributed to Reduce tasks, while the data outside the boundary is transmitted to idle file Job tasks. The task scheduling process is shown in Figure 2.

3.2 Steps of Parallel K-Means Algorithm Based on MapReduce

(1) Key points of improving MapReduce K-means algorithm

According to the characteristics of the MapReduce framework and K-means algorithm, the following problems need to be examined in order to realize the parallel algorithm. All data is distributed to different nodes, and each node only calculates its own data. Each node can read the cluster center generated by the previous iteration and determine which cluster of data points its own data should belong to. Each node calculates the relevant results, in each iteration, according to its own data points. Finally, the actual clustering results are calculated based on the relevant data calculated by each node.

(2) MapReduce K-means algorithm steps

According to the key points of algorithm improvement, data preprocessing and data normalization are performed. The algorithm steps of MapReduce K-means are shown in Figure 3.

1) Use (8) to divide all data points and send them to each node of a cluster.
2) Each node obtains all the central point information of the previous iteration result. If the first iteration is performed, the central points are randomly selected.
3) At the Map stage, each node obtains the data points divided by step 1, and calculates the similarity with each central point.
4) Each node outputs the clustering results of data points, including cluster ID and data points belonging to the cluster. The format is <cluster ID, data points >, and the Map stage ends.
5) In the Reduce phase, the whole <cluster ID, data points> of each node is sent to the Reducer to generate clustering results. The content includes the cluster ID, the mean value and the number of data points belonging to the cluster, and the format is <cluster ID, [number, (mean)]>.

3.3 Process Analysis of Parallel K-Means Algorithm Based on MapReduce

If there are four 2D co-ordinate data points, shown as the data set {A (1, 1), B (2, 3), C (3, 4), D (5, 5)} in Table 1, the preset

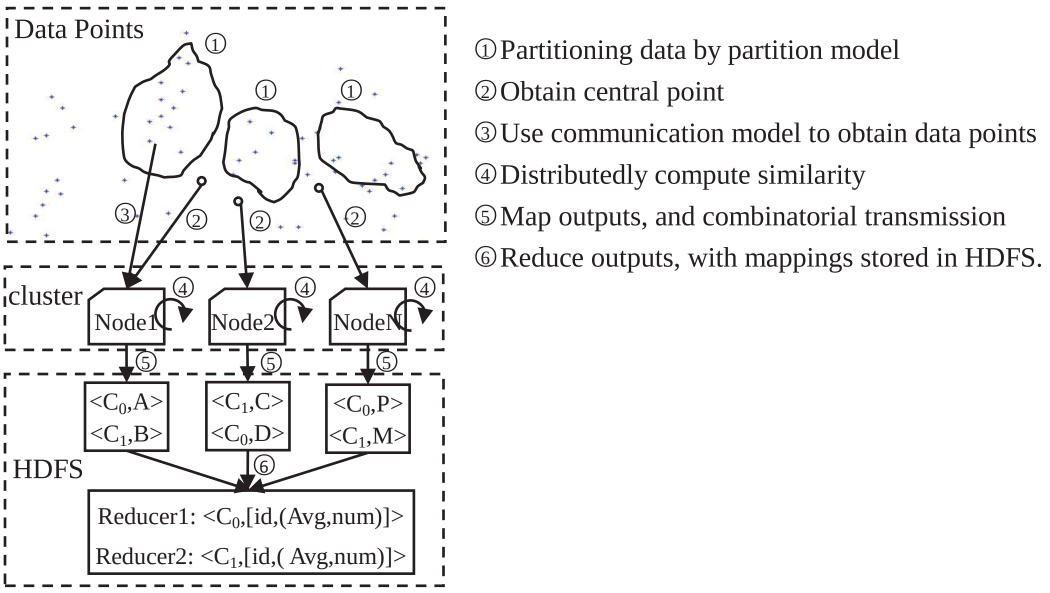
Data Points
①
②
③
④
⑤
⑥
cluster
Node1
Node2
NodeN
HDFS
<C0,A>
<C1,B>
<C1,C>
<C0,D>
<C0,P>
<C1,M>
Reducer1: <C0,[id,(Avg,num)]>
Reducer2: <C1,[id,(Avg,num)]>
①Partitioning data by partition model
②Obtain central point
③Use communication model to obtain data points
④Distributedly compute similarity
⑤Map outputs, and combinatorial transmission
⑥Reduce outputs, with mappings stored in HDFS.

Figure 3 MapReduce K-Means Algorithm Steps.

Table 1 Data Set to be Clustered.

Object	Attribute 1 (X)	Attribute 2 (Y)
A	1	1
B	2	3
C	3	4
D	5	5

Table 2 Initial Partition.

Cluster ID	Cluster center	Allocated point number
cluster0	A (1, 1)	0
cluster1	C (3, 4)	0

Table 3 Initial Distance Calculation on Map Stage.

Data points	Distance to each cluster center		Allocated cluster
	cluster 0	cluster 1	
A (1, 1)	0		cluster0
C (3, 4)		0	cluster1
B (2, 3)	$\sqrt{5}$.		cluster1
D (5, 5)	$4\sqrt{2}$.		cluster1

Table 4 Map Output Result.

node0 output	node1 output
<cluster 0, A (1, 1)>	<cluster 1, B (2, 3)>
<cluster 1, C (3, 4)>	<cluster 1, D (5, 5)>

number of iterations is $k = 2$, and cluster number is $n = 2$, with an execution process as follows.

Assuming there are two nodes, namely node0 and node1. According to the partition model and the communication model, node0: A (1, 1), C (3, 4), and node1: B (2, 3), D (5, 5). A (1, 1) and C (3, 4) are randomly selected as the centers of cluster0 and cluster1, respectively. The clustering situation before the start of the iteration is shown in Table 2.

At the Map stage, the distance from each data point to each cluster center is calculated. From the calculation, it is clear that the distances of point A and C to their clusters are 0, therefore they should be assigned to corresponding clusters. The distances between points B and D to A and C are analyzed separately. It is found that both B and D are closer to point C. Then, B and D should be temporarily classified into the cluster where C is located and marked as Cluster 1.

At this point, the data output on each node can be marked according to Table 4, and can then enter the combination stage. Table 4 is the combination input.

After the calculation of the combined model, the combined output format is <cluster ID, [point number, (mean)]>, and the result is shown in Table 5.

Table 5 Combination Output Result.

node0 output	node1 output
<cluster0, [1, (1, 1)]> < cluster1, [1, (3, 4)]>	<cluster1, [2, (3.5, 4)]>

Table 6 Reducer Result Data.

Reducer 0	Reducer 1
<cluster0, [1, (1, 1)]>	<cluster1, [1, (3, 4)]> <cluster1, [2, (3.5, 4)]>

Table 7 Comparison of Execution Time of 2 Algorithms.

No. of data	No. of clusters	Data block size	K-means	MR K-Means
10	2	4.47	8	11.79
2 100	8	129.61	6 720	4 303
2 900	10	170.29	11 600	7 420
3 300	11	190.53	14 520	9 284
3 700	12	210.71	17 760	11 352
4 500	14	251.00	25 200	16 101
4 900	15	271.11	29 400	18 781

Next in the Reduce stage, since the output of the Map stage has the same cluster ID in different nodes, all the data of each cluster will be sent to the same Reducer, with the format being <cluster ID, [point number, (mean)]>. The results of the two Reducers are shown in Table 6.

By calculation, two clusters, namely cluster0 and cluster1 are obtained. When the terminating condition is satisfied, the number of clusters can be output. In this parallel algorithm, the setting of termination conditions can be consistent with the original K-means, and the specific parameters include the iteration number, the initial center, the variation of square sum of the error, etc.

3.4 Algorithm Complexity Analysis

(1) Algorithm complexity analysis

Assume the sample data set number is n, and k clusters are expected to be generated. Combining partition model, if the total number of data blocks is T, the data block size is n/T, the node number is N, the iteration number is t, and the time consumption of similarity calculation is d, then the time complexity of the K-means algorithm is O (nktd) (Li, 2016). After parallelizing and optimizing the original algorithm, the huge amount of computation can be distributed to all nodes at the same time. The time consumption of the Map phase is O (kn/N), and in the Reduce phase, each cluster's data item is k (n/T)/N, and the time complexity is O (kntd/TN). Therefore, the time complexity of the parallel K-means algorithm based on MapReduce is O (kn/N+ kntd/TN). It is clear that the time complexity is largely influenced by the number of nodes and the size of data blocks. In order to improve load balancing and task scheduling efficiency in Reduce stage, it is necessary to make the data block size equal to the center number of clusters (Tang et al., 2017), i.e., kn/(n/T) = n/T. Therefore, the ideal size of the data blocks is $\sqrt{kn}$ and the complexity of input and output can be reduced.

(2) Experimental analysis of algorithm

In the single machine and MapReduce framework, different data was used to conduct experiments. A Hadoop cluster, including 1 master node and 6 slave nodes, with A8-7500B, 3.2 GHz AMD CPU, 4GB RAM, 2TB hard disk drive, and Ubuntu 14.04 operating system (Zhou et al., 2016). The development tool and platform is Eclipse 8.5 and Hadoop 2.5.2. The data set is Baibu Gulf marine monitoring data, mainly including meteorological, thermohaline and wave data. The meteorological data includes air pressure, air temperature, humidity, wind direction and wind speed. The running time of the two algorithms is compared and shown in Table 7.

From Table 7, it is clear that in the case of a very small amount of data, if the number of nodes is relatively small, the efficiency of the K-means algorithm is obvious. As shown in Figure 4, with the sharp increase of data volume, the advantage of a parallel algorithm based on MapReduce K-means is becoming more and more obvious. If other factors change, the result of the time complexity also varies. As shown in Figure 5, when the number of nodes reaches more than 3, the execution time of the MapReduce K-means algorithm has obvious advantages.

4. CONCLUSION

According to the characteristics of big data clustering, the feasibility of a parallel algorithm based on MapReduce K-means is analyzed, and the technical strategy of a parallel algorithm is studied. On the basis of the traditional K-means algorithm, the model of partition, communication, combination and mapping

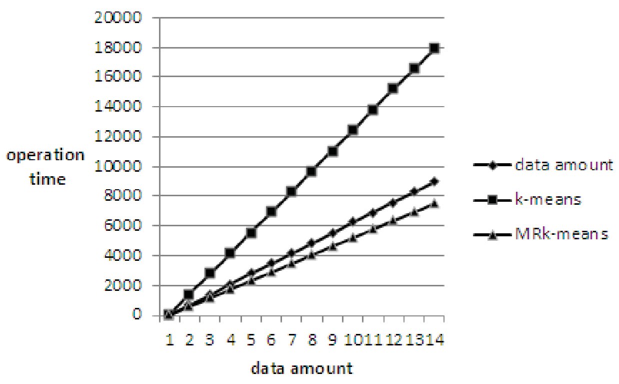

Figure 4 Relationship Between Data Amount and Operation Time.

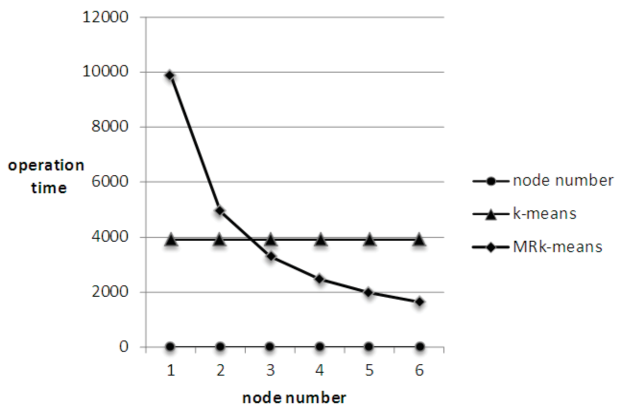

Figure 5 Influence of Different Node Number on Time Complexity.

is established, and the steps of a parallel K-means algorithm based on MapReduce are designed. The operation process and experimental results show that, compared with the K-means algorithm, the K-means parallel algorithm based on the MapReduce framework has more advantages in dealing with the ocean data clustering problem in time complexity. Setting an appropriate cluster number, data block size, iteration number and number of nodes can reduce the computation and data load space of the traditional K-means algorithm, improve the clustering efficiency, and reduce the time, space complexity and data point loss rate.

ACKNOWLEDGMENTS

The research is supported by: Natural Science Foundation of Guangxi in 2016(2016GXNSFAA380102); Natural Science Foundation of Guangxi in 2016(2016GXNSFAA380188); Junior Middle School Teachers Basic Ability Promotion Project of Guangxi in 2017(2017KY0797); Junior Middle School Teachers Basic Ability Promotion Project of Guangxi in 2017(2017KY0795).

REFERENCES

1. Cui, D. 2012. Research and improvement of K-means clustering algorithm [dissertation]. Hefei (AH); Anhui University.
2. Dayong X. (2019). Research on Supply Chain Management Strategy of Longtang Electric Engineering Co. Ltd. Acta Electronica Malaysia, 3(1): 10–13.
3. Fang, L. 2011. Research on key technologies of efficient land resource service platform based on cloud computing [dissertation]. Hangzhou (ZJ); Zhejiang University.
4. Guo, Z. 2008. A parallel genetic algorithm based on adaptive migration strategy [dissertation]. Ganzhou (JX); Jiangxi University of Science and Technology.
5. Jin, R. 2014. Research and application of clustering algorithm for large scale data [dissertation]. Shanghai; Donghua University.
6. Li, L., Dong, Y., Kong, Y. 2016. Parallelization study of improved K-means algorithm on MapReduce programming model. Journal of Harbin University of Science and Technology, 21(1): 31–35.
7. Li, S. 2006. Research and implementation of cluster parallel system based on Linux and MPI [dissertation]. Guangzhou (GD); Guangdong University of Technology.
8. Li, Y., Dong, J. 2012. Study and improvement of MapReduce based on Hadoop. Computer Engineering and Design, 33(8): 3111–3115.
9. Luo, S. 2012. An algorithm for university exam arrangement based on division method. Journal of Changchun University of Technology (Natural Science Edition), 33(2): 193–196.
10. Meisam D., Somayeh P., Masoumeh S. (2019). Webometrics Analysis of Iranian Universities about Medical Sciences' Websites between September 2016 AND March 2017. Acta Informatica Malaysia, 3(1): 07–12.
11. Ou Z., He B.X., Xiaohong Y., Bojia P.I., Reine D.C., R.A. Xingran Z. (2019). A Look at Millennial Attitudes Toward AI Utility in The Class. Information Management and Computer Science, 2(1): 07–09.

12. Tang, H., Yang, Y., Xin, Z. 2017. A single-pass K-means clustering algorithm with MapReduce. Computer Technology and Development, 1–6.
13. Wang, X. 2010. Optimization of high performance MapReduce system [dissertation]. Hefei (AH); University of Science & Technology China.
14. Yu, J. 2017. Research on radio resource allocation strategy based on context information in cellular networks [dissertation]. Hefei (AH); University of Science & Technology China.
15. Zhou, R., Li, Z., Chen, S. 2016. Parallel optimization sampling clustering K-means algorithm for big data processing. Journal of Computer Applications, 36(2): 311–315, 329.
16. Zhu, K., Huang, R., Zhang, N. 2017. Efficient frequency patterns mining algorithm based on MapReduce model. Computer Science, 44(7): 31–37.