Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 11, November 2019, pp. 2860-2870
DOI: 10.23940/ijpe.19.11.p4.28602870

Hadoop-based Parallel Algorithm for Data Mining in Remote Sensing Images

Yanhua Wang, Yaqiu Liu*, and Weipeng Jing

School of Information and Computer Engineering, Northeast Forestry University, Harbin, 150040, China

Abstract

As a typical distributed parallel computing model, cloud computing can greatly reduce the execution time of computing tasks. Remote sensing image data mining, an important part of data mining, plays a significant role in meteorological analysis and earthquake prediction. By constructing a Hadoop cloud computing platform, this paper studies the Hadoop-based parallel algorithm for remote sensing image data mining. In accordance with the Hadoop distributed computing framework, the parallel algorithm for remote sensing image data mining is realized through data preprocessing, image feature extraction, and clustering analysis. The main work of this paper includes image preprocessing, Hadoop-based parallelization of remote sensing image feature extraction, and a Hadoop-based parallel algorithm for remote sensing image data mining.

Keywords: cloud computing; data mining; feature extraction

(Submitted on September 15, 2019; Revised on September 30, 2019; Accepted on October 12, 2019)

© 2019 Totem Publisher, Inc. All rights reserved.

1. Introduction

Data mining is the process of digging out hidden yet previously known knowledge and information from a large amount of incomplete, noise-containing, fuzzy, and random data [1]. The purpose of data mining is to discover information or knowledge that cannot be found by intuition. The more unexpected the information is, the more valuable it may be. Data mining uses various analytical tools to determine the relationship between models and data. Through these models and relationships, it helps decision-makers to find the potential correlations among data and discover the neglected elements, and thus it is considered an effective means to solve the current problem of data explosion with a lack of useful information. Data mining technology has been successfully used in weather forecasting, meteorological analysis, earthquake prediction, communications, as well as military and financial areas [2-3].

As a challenging subdomain of data mining, image data mining is one of the most advanced research directions in the field of multimedia database and information decision-making, and it has one important branch: remote sensing image mining. Remote sensing image shares some main features of image data, including color, texture, shape, and spatial features [4-5]. It also has its own features: the multiple sources of remote sensing information and its space, spectrum, and intensity [6]. Due to the above characteristics of remote sensing images, the amount of data and information that need to be processed during remote sensing image data mining is massive. Therefore, it has become a major problem in remote sensing image data mining to extract effective information and achieve rapid data mining.

Cloud computing is a typical distributed and parallel computing model, and it can greatly reduce the execution time of computing tasks. As one of the most developed cloud computing platforms, Hadoop is an open source software framework that has been favored by big data researchers and successfully applied to image processing due to its reliability, scalability, and distributed computing and storage [7-8]. This open source cloud platform has massive storage capacity and computing power ,which can be used to effectively solve the major problems in remote sensing image data mining.

* Corresponding author.
E-mail address: Yaqiuliu@126.com

Based on the above-mentioned information, this paper studies the parallel algorithm for remote sensing image data mining on the Hadoop cloud computing platform. It focuses on the following areas:

(1) Image preprocessing. As image files cannot be simply divided or assembled and Hadoop is not powerful enough in image processing, there is no special interface class for images. Based on the binary file storage form SequenceFile provided by Hadoop, the image data is serialized into byte streams and stored in binary files. When performing the MapReduce task, it can be read directly with the SequenceFileInputFormat provided by Hadoop.

(2) Hadoop-based parallelization of feature extraction in remote sensing images. This study improves the parallel extraction algorithm for remote sensing image features through the custom partition strategy. It increases the number of Reduce tasks in the MapReduce program, which extracts the features of remote sensing images to improve the efficiency of parallel processing.

(3) Hadoop-based parallel algorithm for remote sensing image data mining. The initial clustering center and the number of clusters in the K_Means clustering algorithm need to be manually determined, causing the clustering results to fall into a local optimum. This paper uses the Canopy algorithm to improve the K_Means algorithm by roughly clustering the features of remote sensing images and then uses the result as the initial clustering point for the K_Means algorithm to complete the classification of remote sensing images.

2. Relevant Work

Since the middle of the 20th century, data mining has proposed many algorithms to solve practical problems and has developed into a discipline of multiple fields such as artificial intelligence, machine learning, natural language processing, and image processing. With the advent of the big data era, data information is becoming more complex and the amount of data is becoming ever larger. It is now a key research direction for experts in related fields to combine cloud computing and data mining [9].

At present, many practical data mining platforms have been developed abroad. Relevant literature [10] proposed an open source data mining platform WeKa, which is a collection of machine learning algorithms that can perform data mining tasks and has milestone significance. On the basis of WeKa, literature [11] developed a distributed data mining platform Weka4WS, which can effectively improve the data mining efficiency when dealing with large amount of data. As Hadoop develops, scholars and enterprises now focus on the application of Hadoop in data mining. To solve the problem of massive data processing, Microsoft has built business intelligence mining tools by combining Windows Server and Hadoop. MicroStrategy, IBM, and Oracle launched cloud computing platforms to provide enterprises with better data mining services. Google, YAHOO, and other companies have also built cluster structures to search for and analyze logs and big data files [12].

Data mining started late in China, but it has made some achievements in recent years with the rapid development of Internet technology. This field mainly focuses on data learning algorithms, applications of data mining, and its theoretical breakthrough. PDMiner (Parallel Distributed Miner), developed by the Institute of Computer Technology of CAS on the basis of Hadoop platform, has been used in projects of China Mobile to mine user behavior data [13]. Literature [14] proposed the application of Hadoop-based data mining technology in weather forecasts and studied the classification of weather data through the Bayes classification algorithm. Using the advantages of the Hadoop platform in dealing with massive data, literature [15] realized MapReduce parallelization in the K-Means algorithm and accelerated its processing. Then, literature [16] proposed a dynamic K-Means algorithm, in which the BP neural network algorithm was used to determine the clustering result through a dynamic split-and-merge iteration process, further improving the processing efficiency of data mining. Based on these previous studies, this paper adopts the Hadoop distributed computing framework to propose a parallel algorithm for remote sensing image data mining through data preprocessing, image feature extraction, and clustering analysis.

3. Image Pre-Processing

3.1. Introduction to SequenceFile

SequenceFile is a form of binary file storage provided by Hadoop. It represents various files as key-value pairs, serializes them into byte streams, and stores them as binary files [17]. It uses Hadoop's standard Writable interface to serialize and deserialize files.

SequenceFile is mainly composed of a "header" and many records. The header contains the version number, the class name of the key/value, the compression information, and the synchronization mark. For each record, the format varies depending on whether it is compressed or not. The stored record contains the key, value, record length, and key length. The storage structure of SequenceFile is shown in Figure 1.

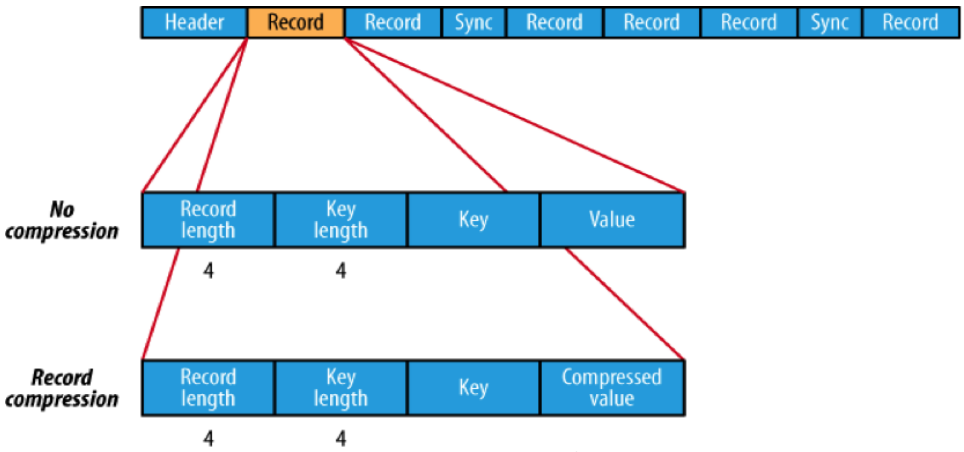

Figure 1. The storage structure of SequenceFile

SequenceFile has two important internal classes, SequenceFile.Reader and SequenceFile.Writer. SequenceFile.Reader performs the reading operation by creating an instance first and then repeatedly calling the next() method to iteratively read the records in the binary file saved by the writing operation. SequenceFile.Writer creates an object via the createWriter() static method and then returns an instance that can be used to add a key-value pair to the end of the file by calling the append() method.

3.2. Conversion of Remote Sensing Image Data into SequenceFile

Hadoop does not provide data types that represent images when it is used in image processing, while SequenceFile can represent various files as key-value pairs and serialize them into byte stream. Therefore, in this paper, remote sensing image data is converted into SequenceFile to accomplish the preprocessing.

Algorithm 1: Conversion of remote sensing image data into SequenceFile

```
FileSystem fs = FileSystem.get(new Configuration());
// Obtain input information of remote sensing image
Path  input = new Path(FileName);
// Obtain byte stream information from the input stream
FSDataInputStream in = fs.open(input);
byte[] b = new byte[in.available()];
in.readFully(b);
// Take the name of the input file as the key
Text key = new Text(input.getName());
// Convert the input format to BytesWritable as the value
BytesWritable value = new BytesWritable(b);
// Create SequenceFile.Writer object
SequenceFile.writer writer = SequenceFile.createWriter(fs,new Configuration(),
new Path(OutPut),Text.class,BytesWritable.class);
// Add the key-value pair to binary file
writer.append(key,value);
// Close the stream
in.close();
IOUtils.closeStream(writer);
```

In actual operation, the file names of remote sensing images are taken as the key because the images can be grouped according to their file names. The sorting efficiency is also higher as the file names are shorter. The content of the image, which represents information about the pixel, is taken as the value. Thus, the operation of remote sensing images becomes the operation of pixel values, which can be operated directly by the MapReduce program in Hadoop.

The specific process can be described as follows: first, read the data according to the input path of remote sensing images. Then, create input stream to get byte stream information of the file. Finally, call the createWriter method to

generate a SequenceFile.Writer object. The file name is key, and the data stream is value. It is then added to the binary file through the append () method of the SequenceFile.Writer object. The details are shown in Algorithm 1.

In the code above, FileName is the input information, and OutPath is the path information for serialized binary files. After the binary file is obtained, the conversion of the remote sensing image data format is completed. In subsequent distributed image processing, SequenceFileInputFormat can be used as the input format for data.

4. Hadoop-based Design for Parallel Feature Extraction in Remote Sensing Image

The extraction of image features provides a theoretical basis for image recognition and serves as the foundation for image data mining. This paper uses the gray histogram to extract image features and parallelizes feature extraction in remote sensing images through the MapReduce program in Hadoop. By customizing the partition classes in MapReduce, the number of Map and Reduce tasks executed in parallel can be increased to further improve parallel processing efficiency.

4.1. Feature Extraction Through Gray Histogram Arithmetic

As a key step in pattern recognition and classification, feature extraction measures the intrinsic and essential features or attributes of a research object to obtain numerical results, or it decomposes and symbolizes the object to form a feature vector, symbol string, and relation graph. Feature extraction based on gray histogram is a typical algorithm in that the gray histogram of an image contains rich information. It provides a global description of the image, and the extracted features are invariant in RST (rotation, scale, and translation).

A gray histogram is a statistical graph that represents the grayscale distribution of images. In mathematics, it is a function of statistical characteristics and gray values of images, and it counts the number of occurrences of each grayscale in an image. The gray histogram of an image is a discrete function of grayscale that can be defined by Equation (1).

$$H(i) = \frac{n_i}{N}, i = 0,1,\cdots,L-1 \tag{1}$$

Where i represents the grayscale, L represents the number of grayscale classes, n_i represents the number of pixels with grayscale i in the image, and N represents the total number of pixels in the image. The equation describes the percentage of pixels with this grayscale in all pixels of the image, which is also the frequency of pixels with grayscale i in the image.

The gray histogram is an important statistical property of images. Useful features that reflect image characteristics can be obtained through the analysis of gray histograms. Due to its calculation convenience and RST invariance, the gray histogram is used in feature extraction of remote sensing images to classify images in this paper.

4.2. Hadoop-based Parallel Extraction of Image Feature

When a gray histogram is used in feature extraction of remote sensing images in Hadoop clusters, the built-in partitioning strategy is adopted by default. This paper, however, implements the Partitioner interface in Hadoop and customizes the partition strategy to deal with image features. The number of Reducer tasks in the execution process of MapReduce is increased so as to further improve the efficiency of parallel extraction of image features.

4.2.1. The Partition Function Interface in Hadoop

The partition process by Partitioner is executed after the Mapper task and before the Reducer task. Its main function is to distribute the intermediate results of Mapper tasks to different Reducer tasks according to key values. Partition is achieved by dividing data in Mapper tasks, and the class responsible for implementing the partition is called Partitioner, an abstract class that has interface. This class contains only one abstract method, getPartition, which is used to return the partition number of the key value.

HashPartitioner is the default implementation class of Partitioner and uses the Hash function to partition the output of Mapper. In the getPartition method of HashPartitioner, there are three parameters: the first two are the key and value of Mapper output, and the third is the number of Reducer tasks. Suppose that the key value output of the Hadoop-based Mapper task is $\alpha_1, \alpha_2, \cdots, \alpha_n$, P_{α_i} represents the partition number corresponding to the key value $\alpha_i (i = 1,\cdots,n)$, and H_{α_i}

represents the hash of α_i. Then, Equation (2) can be obtained.

$$P_{\alpha_i} = \left(H_{\alpha_i} \& M\right) \% N, i = 1,2,\cdots,n \tag{2}$$

In this equation, *M* is the constant $(2^{31}-1)$ representing the maximum value of type int, and *N* represents the number of Reducer tasks. It should be noted that by default *N* is 1, and thus the return value of the getPartition method is always 0, which means that the output of the Mapper task is always sent to a Reducer task for processing.

4.2.2. Parallel Extraction of Image Feature based on Custom Partition

In the Hadoop cluster, the default partition class HashPartitioner is used when extracting the image features with no partitioning strategy specified, in which all processing tasks are carried out in one Reducer. Therefore, in order to improve the efficiency of parallel processing, this paper extracts image features by means of custom partition.

Custom partition can be achieved by inheriting the Partitioner class and implementing the partition method in the Partitioner class that serves as the base for all partition strategies. The specific operation is to define the partition class MyPartition by inheriting the Partitioner abstract class through extends keyword and rewrite the getPartition () method. When processing remote sensing images in the MapReduce program, the file name of the image, which has been given in the preprocessing stage, is taken as the key. Therefore, in the getPartition method, the matching of file names of remote sensing images is achieved by calling the equals () method of the incoming parameter key to divide the matched data into a partition. According to the total number of partitions, the preprocessed remote sensing image data is equally divided to ensure load balancing in the process of feature extraction.

After the custom partition is completed, the setPartitionerClass() method of the Job object in MapReduce needs to be called to input the defined MyPartition class as a parameter. The number of the desired partitions also needs to be input by calling the setNumReduceTasks() method of the Job object. The parallel algorithm for remote sensing image feature extraction based on custom partitions is shown in detail in the following Algorithm 2.

Algorithm 2: Parallel algorithm for remote sensing image feature extraction

Input: Preprocessed remote sensing image data
Output: Image features

```
1. Set the input format as SequenceFileInputFormat and the output format as IntWritable. Set MyPartition and specify NumReduceTasks.
2. Start the MapReduce task.
3. Mapper (Text key, BytesWritable value, Context context):
//Obtain the byte array for remote sensing image data
byte[] ImageByte=value.getBytes();
//Create a BufferedImage object based on the byte array
BufferedImage image=ImageIO.read(new ByteArrayInputStream(ImageByte));
StringBuffer s=new StringBuffer();
//Obtain the gray histogram of image
GrayHistogram(image,s);
//Store the gray histogram in the Context object
Context.write(key,new Text(s.toString()));
4. Reducer (Text key, Iterable<Text>values, Context context):
   //Obtain the iterator object that encapsulates the value
 Iterator<Text> it=values.iterator();
   //The image feature processed by Reducer is written into the Context object for output.
   while(it.hasNext()){
       Context.write(key,it.next());
                   }
```

After remote sensing image data are preprocessed, the SequenceFile binary format that can be processed by Hadoop is generated. Then, the Map operation is performed, and remote sensing image features are extracted. The subsequent custom partitioning enables the Reduce task on multiple nodes of the cluster, leading to the concurrent output of remote sensing image features. The process is shown in Figure 2. The custom partition increases the number of Reduce in Hadoop-based remote sensing image feature extraction, thereby improving the parallel processing efficiency of the program.

5. Hadoop-based Parallel Design for Remote Sensing Image Data Mining

In the field of data mining, cluster analysis is a popular research branch. As a classical algorithm in cluster analysis, the

K_Means algorithm features simplicity, rapid convergence, and easy implementation. However, the initial clustering centers and the number of clusters need to be determined in advance, and thus it can be easily influenced by subjective factors and lead to a local optimum. Therefore, this paper makes an improvement, in which the Canopy algorithm is used to "roughly" cluster data. Then, the K_Means algorithm is used to perform "fine" clustering.

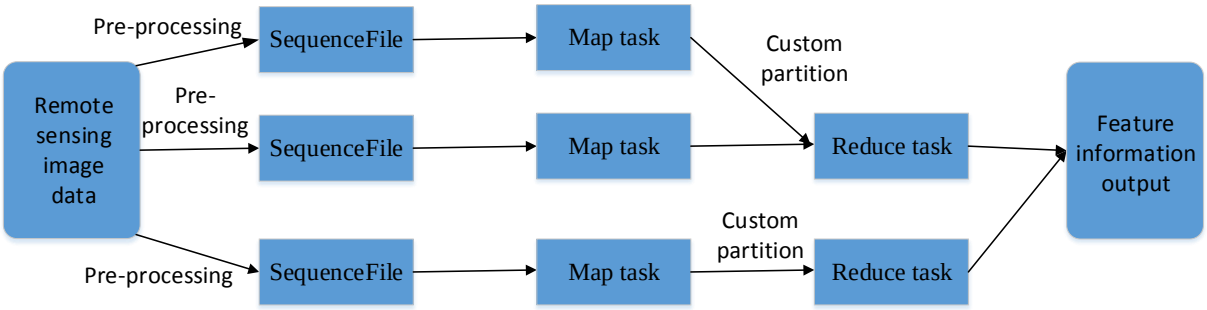

Figure 2. Flow chart for parallel extraction of remote sensing image features based on custom partition

5.1. K_Means Clustering Algorithm

The K_Means algorithm is a distance-based clustering algorithm that aims to partition the data in the dataset into k clusters, so that the data points in the cluster are closest to the cluster center. Its main idea is listed as follows:

(1) A random selection of *k* objects from the dataset is used as the initial cluster center vector.
(2) For the rest of the data objects, their distance to the cluster center vector is calculated in turn to divide them into nearest clusters.
(3) Each cluster center is recalculated to be compared with the previous cluster center.
(4) The above iteration is repeated until the clustering criterion function converges.

The standard deviation is generally used as the standard measure function by Equation (3).

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{3}$$

Where V is the sum of standard deviation of all objects in the dataset, x_j represents a point in the object space, and μ_i is the mean of points in S_i (x_j and μ_i are multidimensional).

The main problem of this algorithm is that it requires the user to set the number of clusters to be generated in advance, and it is sensitive to initial values. Different initial values may lead to different clustering results.

5.2. Classification Algorithm for Remote Sensing Images based on the Improved K_Means Algorithm

Regarding the problems of the K_Means clustering algorithm, this paper uses the Canopy algorithm in preprocessing the data to obtain the number of cluster centers and the initial cluster centers. Canopy is a fast pre-clustering algorithm that computes data by using two artificially determined thresholds, T_1 and T_2 ($T_1 > T_2$), to classify confusing data into *n* datasets. The *n* classes calculated by Canopy can be used to determine *k* in the K_Means algorithm. This method can reduce the number of iterations of K_Means, avoid the K_Means local optimum, and improve the data processing efficiency.

When using the improved K_Means algorithm to perform clustering analysis of remote sensing image data, it is necessary to obtain the initial clustering points through the Canopy algorithm and then use the Canopy clustering results as the initial partition of K_Means for subsequent iterations. Therefore, the entire process includes two stages: the first stage executes the Canopy algorithm, and the second stage implements the K_Means algorithm. The implementation of MapReduce in the Canopy algorithm is shown in Algorithm 3.

Algorithm 3: The implementation of MapReduce in the Canopy algorithm

1. Mapper:
//Create a List collection that encapsulates the ClusterCenter object

```
List<ClusterCenter> conters=new LinkedList<ClusterCenter>();
IntWritable dummy=new IntWritable(1);
Vector vector=new Vector(new StringTokenizer(value.toString()).nextToken);
//Add the vectors that can be used as cluster centers to the centers collection
boolean isInCanopy=false;
for(ClusterCenter center:centers):
      //Calculate the distance between the data object and the cluster center
      double dist=DistanceMeasure.distance(center,vector);
      if(dist<T1){
            //Add the data object to the corresponding cluster
            Context.write(dummy,center);
            idInCanopy=true;
                  }
     if(!isInCanopy){
              //Take the data object as a new cluster center
          centers.add(new ClusterCenter(vector));
          context.write(dummy,new ClusterCenter(vector));
                     }
2. Reducer:
  //Calculate the number of cluster centers in each Canopy and output the number via the Context object.
for(int i=0;i<numOfCentersPerCanopy;canopy;i++){
       Context.write(new ClusterCenter(CanopyId,randVector),new IntWritable(0));
                                }
```

In the above code, ClusterCenter is the defined class for cluster centers, and Vector is the class that encapsulates data collection. When calculating the distance between the data object and the cluster center, we use the Manhattan distance equation, which can be expressed by Equation (4).

$$dis([x_1, x_2, \cdots, x_n], [y_1, y_2, \cdots, y_n]) = \sum_{i=1}^{n} |x_i - y_i| \tag{4}$$

Where $[x_1, x_2, \cdots, x_n]$, $[y_1, y_2, \cdots, y_n]$ represent the coordinate values of *n*-dimensional vectors $\alpha(x_1, x_2, \cdots, x_n)$ and $\beta(y_1, y_2, \cdots, y_n)$ in *n*-dimensional Euclidean space.

The data processed by Canopy will then serve as the initial clustering point of the K_Means clustering algorithm to partition the data objects. The Map function partitions the data object on the node into its nearest cluster vector, and the Reduce function regenerates a new cluster center vector. The implementation of MapReduce in the K_Means algorithm is shown in Algorithm 4.

5.3. Hadoop-based Parallel Algorithm for Data Mining in Remote Sensing Images

This paper realizes image preprocessing by converting remote sensing image data into SequenceFile binary format and then uses the MapReduce program to extract image features on the basis of the custom partition strategy. Finally, it adopts the improved K_Means clustering algorithm to complete the classification of remote sensing images. The flow chart is shown in Figure 3.

Algorithm 4: The implementation of MapReduce in the K_Means algorithm

```
1. Mapper:
Vector vector=new Vector(new StringTokenizer(value.toString()).nextToken);
ClusterCenter nearest=null;
double nearestDistance=Double.MAX_VALUE;
  //Add the data object to its nearest cluster center
for(ClusterCenter center:centers):
   double dist=DistanceMeasure.distance(center,vector);
   if(nearestDistance>dist){
     nearest=center;
     nearstDistance=dist;
                          }
  //Output Map results
context.write(nearest,vector);
2. Reducer:
List<ClusterCenter> conters=new LinkedList<ClusterCenter>();
IntWritable dummy=new IntWritable(0);
  //Obtain new cluster center
context.getCounter(Counter.K_CENTER).increment(1);
```

```
int k=(int)contxt.getCounter(Counter.K_CENTER).getValue();
  //Output cluster identifier and new cluster center
for(Vector vector:values){
     Context.write(new ClusterCenter(vector),new IntWritable(k));
                          }
```

After being submitted to the Hadoop cluster, the image data is preprocessed and converted to SequenceFile, the built-in binary storage form in Hadoop. Then, the master node distributes the preprocessed image data to each slave node in the cluster to perform image feature extraction by executing the MapReduce program concurrently in multiple nodes of the cluster. The extracted image feature information is then sent back to the master node to be aggregated and redistributed to the slave node, which restarts the new MapReduce program to complete cluster analysis. Finally, the main node outputs the image classification results.

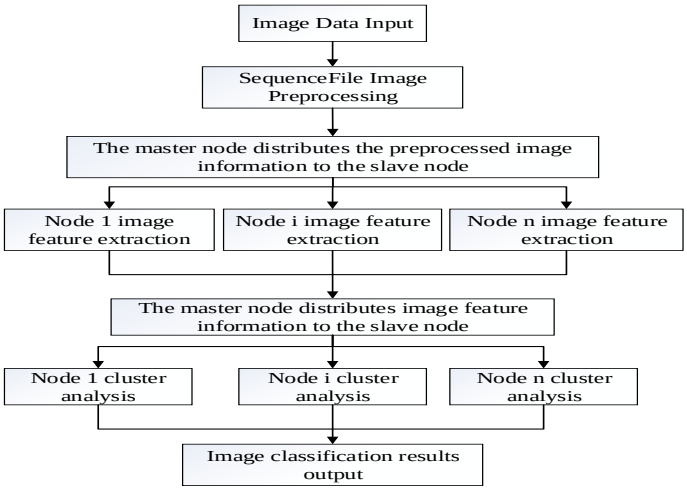

Figure 3. Hadoop-based parallel algorithm for data mining in remote sensing images

6. Experimental Results and Analysis

6.1. Experimental Environment

The Hadoop cluster consists of five servers: one is the Master, and the other four each have four virtual nodes (16 slave nodes total). The four servers are tower servers Dawning I450-G10 with an InterXeon E5-2620 six core 2.1GHZ processor, 8 GB memory, and 300 G hard disk. Red Hat 6.2 is installed on each node with Linux 2.6.32, jdk-7u71-linux-x86. gz and hadoop-2.5.2.tar.gz.

6.2. Analysis of the Processing Efficiency of Image Feature Extraction

6.2.1. Contrastive Experiment on Processing Time and Speedup Ratio

This experiment selects Landsat7 ETM+ remote sensing data with a resolution of 30 m and a volume of approximately 60 MB data in each image. It analyzes the differences in processing time and speedup ratio between the improved parallel algorithm for image feature extraction based on the custom partition strategy and the traditional parallel algorithm in the processing of 60 images as the number of DataNodes increases in the cluster. The speedup ratio is defined by Equation (5).

$$speedup = \frac{T_1}{T_p} \tag{5}$$

Where T_1 represents the run time for one node, and T_p is the time for parallel run of p nodes.

As the number of nodes in the cluster increases, the trends of processing time and speedup ratio in the improved parallel algorithm and the traditional parallel algorithm are respectively shown in Figures 4 and 5.

From Figures 4 and 5, it can be seen that the parallel algorithm for feature extraction based on the custom partition

strategy has a better speedup ratio than the traditional parallel algorithm. This is because the custom partition increases the number of Reduce tasks in the cluster so that the MapReduce program can be run in parallel on multiple nodes in the Hadoop cluster. This improves the utilization of nodes and the parallel processing efficiency of feature extraction in remote sensing images.

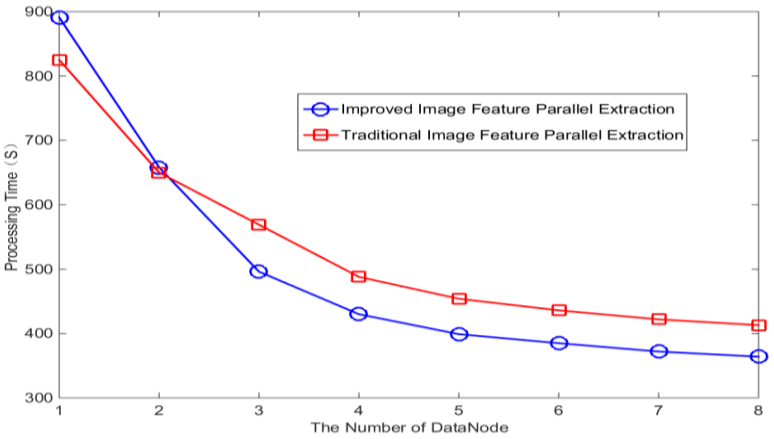

Figure 4. Contrast in processing time (as the number of nodes increases)

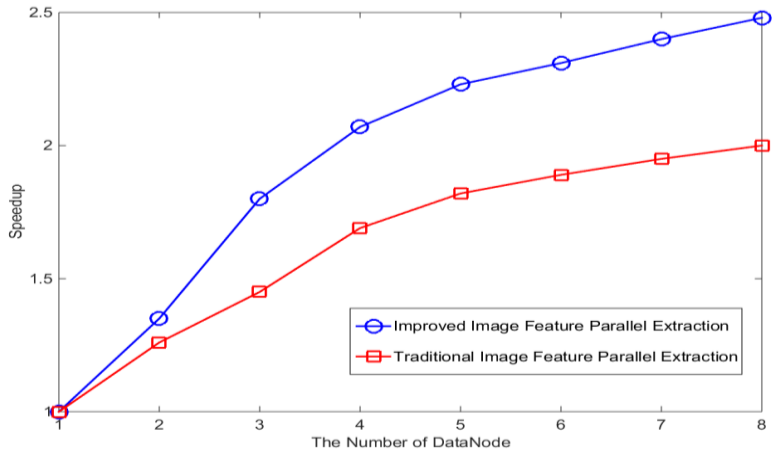

Figure 5. Contrast in speedup ratio (as the number of nodes increases)

6.2.2. Contrastive Experiment on Processing Time and Throughput Rate

This experiment selects Landsat7 ETM+ remote sensing data with a resolution of 30m as the amount of data increases from 242.8 MB to 4379.2 MB. It analyzes the differences in processing time and throughput between the improved parallel algorithm and the traditional parallel algorithm for image feature extraction as the amount of data increases. The throughput is defined by Equation (6).

$$F = \frac{M}{T} \tag{6}$$

Where F is the rate of throughput, M is the amount of data, and T is the processing time.

As the amount of data increases, the trends of processing time and throughput in the improved parallel algorithm and the traditional parallel algorithm are shown in Figure 6.

Figure 6 shows that when data amount is small, the parallel algorithm for image feature extraction based on custom partition has a lower throughput than the traditional parallel algorithm. With a small amount of data, there are fewer tasks that need parallel processing in the cluster, and thus one Reduce task can meet the processing needs. After being partitioned, however, the number of Reduce tasks increases, and more time will be consumed due to data movement among different nodes, causing the processing efficiency to decline. As data increases, the degree of parallelism in clusters will be improved,

and the throughput rate is increased compared with that of the traditional algorithm. However, due to the existence of communication overhead among nodes, the growth rate for data throughput will slow down or even decrease slightly and finally remain relatively balanced.

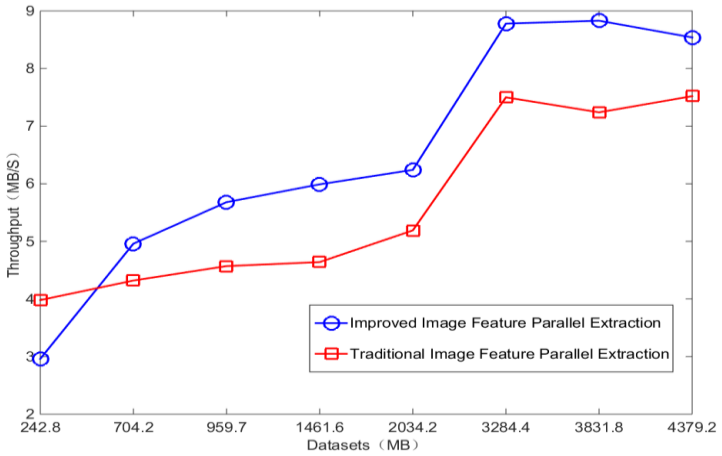

Figure 6. Contrast in throughput (as the data amount increases)

6.3. Analysis of the Processing Efficiency of Image Data Mining

This experiment selects Landsat7 ETM+ remote sensing data with a resolution of 30 m. Each remote sensing image is a 20 dimensional vector after feature extraction. It studies the differences in speedup ratio between the improved K_Means clustering algorithm adopted in this paper and the traditional K_Means clustering algorithm in the classification of 300 images as the number of nodes increases. The trend is shown in Figure 7 below.

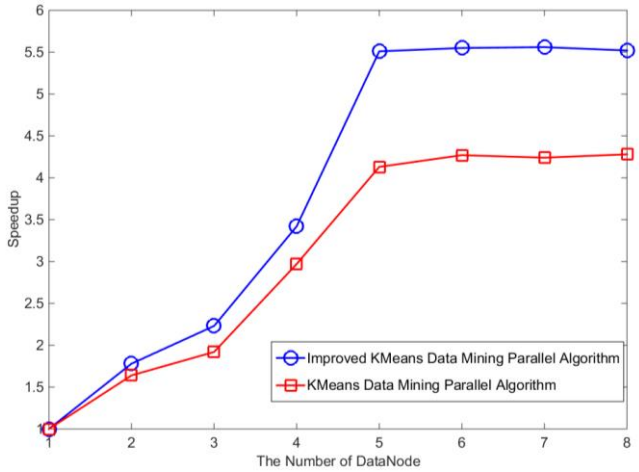

Figure 7. Contrast in speedup ratio (as the number of nodes increases)

It can be seen that the improved K_Means parallel algorithm for data mining has a better speedup than the traditional K_Means parallel algorithm. This is probably because the traditional K_Means clustering algorithm takes a long time to determine the number of optimal clustering class, while the improved K_Means clustering algorithm used in this paper simplifies the data distance measurement by the Canopy algorithm. Compared with the traditional K_Means clustering algorithm, this reduces the computation time in determining clustering centers and improves the processing efficiency. As the number of nodes increases, the task scheduling and data communication overhead on clusters will increase, and the growth rate for speedup ratio will slow down.

7. Conclusions

Remote sensing image data mining, as an important part of data mining, plays an important role in meteorological analysis and earthquake prediction. This paper conducts a Hadoop-based study on a parallel algorithm for remote sensing image data mining. First, the remote sensing image data is preprocessed and converted to a binary byte stream that can be stored in

SequenceFile in Hadoop. Then, the image features are extracted in parallel by the MapReduce program based on the custom partition strategy. Finally, the Canopy algorithm and the K_Means clustering algorithm are combined to finish the classification of remote sensing images in accordance with the extracted features.

Acknowledgements

The work described in this paper is supported by the National Natural Science Foundation of China (No. 31770768), the Natural Science Foundation of Heilongjiang Province of China (No. F2017001), the Fundamental Research Funds for the Central Universities (No. 2572017CB32), and China Forestry Nonprofit Industry Research Project (No. 201504307).

References

1. M. S. Chen, J. Han, and P. S. Yu, "Data Mining: An Overview from a Database Perspective," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 866-883, June 2002
2. M. H. Dunham, "Data Mining Introductory and Advanced Topics," Tsinghua University Press, Beijing, China, 2005
3. Y. T. Zhang and L. Gong, "Data Mining Principles and Technology," Electronic Industry Press, Beijing, China, 2004
4. Q. J. Wang and L. F. Xue, "Data Mining Technology and Its Application in Geoscience," *World Geology*, Vol. 19, No. 3, pp. 235-239, March 2000
5. Z. Y. Li, H. Z. Wang, W. Shao, J. Z. Li, and H. Gao, "Repairing Data through Regular Expressions," *PVLDB*, Vol. 9, No. 5, pp. 432-443, 2016
6. R. Agrawal, T. Imielinski, and A. Swami, "Database Mining: A Performance Perspective," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, pp. 914-925, June 2002
7. "The Apache Software Foundation. Apache Hadoop 2.9.2," (http://hadoop.apache.org/docs/stable/, accessed November 13, 2018)
8. Q. L. Han, S. Liang, and H. L. Zhang, "Mobile Cloud Sensing, Big Data, and 5G Networks Make an Intelligent and Smart World," *IEEE Network*, Vol. 29, No. 2, pp. 40-45, 2015
9. Y. H. Huang, "In-Depth Understanding of Big Data: Big Data Processing and Programming Practice," Machinery Industry Press, Beijing, China, 2014
10. G. Holmes, A. Donkin, and I. H. Witten, "WEKA: A Machine Learning Workbench," in *Proceedings of the 2nd Australian and New* Zealand *Conference on Intelligent Information Systems*, pp. 357-361, Brisbane, Australia, December 1994
11. D. Talia, P. Trunfio, and O. Verta, "Weka4WS: A WSRF-Enabled Weka Toolkit for Distributed Data Mining on Grids," in *Proceedings of European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 309-320, Berlin, Germany, September 2005
12. G. Liu, B. Hou, and Z. W. Zhai, "Hadoop Open Source Cloud Computing Platform," Beijing University of Posts and Telecommunications Press, Beijing, China, 2011
13. Q. He, F. Z. Zhuang, L. Zeng, W. Z. Zhao, and Q. Tan, "PDMiner: A Cloud Computing based Parallel and Distributed Data Mining Toolkit Platform," *Scientia Sinica*, Vol. 44, No. 7, pp. 871-885, July 2014
14. L. J. Yang, "Application of Data Mining Technology in Weather Data based on Hadoop Cloud Platform," Beijing University of Posts and Telecommunications, 2015
15. W. B. Pan, "Research and Application of Parallel K-Means Meteorological Data Mining based on Cloud Computing," Nanjing University of Information Science and Technology, 2013
16. L. M. Bao, "Research on Parallelization of Dynamic K-Means Algorithm in Remote Sensing Image Mining," Nanjing University of Posts and Telecommunications, 2017
17. Y. W. Li, "Research on File Copy Storage Improvement and Small File Merge and Access Optimization on Hadoop Platform," Wuhan University of Technology, 2015

Yanhua Wang is a Ph.D. student in the School of Information and Computer Engineering at Northeast Forestry University. Her research interests include cloud computing and software engineering.

Yaqiu Liu is a professor in the School of Information and Computer Engineering at Northeast Forestry University. His research interests include cloud computing and software engineering.

Weipeng Jing is an associate professor in the School of Information and Computer Engineering at Northeast Forestry University. His research interests include cloud computing and software engineering.