

Article

Efficient Retrieval of Massive Ocean Remote Sensing Images via a Cloud-Based Mean-Shift Algorithm

Mengzhao Yang, Wei Song and Haibin Mei *

College of Information Technology, Shanghai Ocean University, Shanghai 201306, China;
mzyang@shou.edu.cn (M.Y.); wsong@shou.edu.cn (W.S.)

* Correspondence: hbmei@shou.edu.cn; Tel.: +86-130-5231-2351

Received: 21 June 2017; Accepted: 19 July 2017; Published: 23 July 2017

Abstract: The rapid development of remote sensing (RS) technology has resulted in the proliferation of high-resolution images. There are challenges involved in not only storing large volumes of RS images but also in rapidly retrieving the images for ocean disaster analysis such as for storm surges and typhoon warnings. In this paper, we present an efficient retrieval of massive ocean RS images via a Cloud-based mean-shift algorithm. Distributed construction method via the pyramid model is proposed based on the maximum hierarchical layer algorithm and used to realize efficient storage structure of RS images on the Cloud platform. We achieve high-performance processing of massive RS images in the Hadoop system. Based on the pyramid Hadoop distributed file system (HDFS) storage method, an improved mean-shift algorithm for RS image retrieval is presented by fusion with the canopy algorithm via Hadoop MapReduce programming. The results show that the new method can achieve better performance for data storage than HDFS alone and WebGIS-based HDFS. Speedup and scaleup are very close to linear changes with an increase of RS images, which proves that image retrieval using our method is efficient.

Keywords: remote sensing (RS); fast retrieval; ocean disasters; mean-shift algorithm; Hadoop system; pyramid HDFS storage

1. Introduction

Ocean disasters have claimed thousands of lives and cause billions in economic damage each year. We must use the latest remote sensing (RS) technologies and computational methods to rapidly detect or forecast some devastating events and their effects (e.g., red and green tides, oil spills, storm surges, typhoons, and the sea ice). With the rapid development of high-resolution ocean satellites and data transmission technology, the number of RS images has grown considerably and the big data era of RS images has truly emerged. To process the massive images rapidly to meet fast requirement of ocean disaster warnings, Cloud computing platforms [1] such as MapReduce, the Hadoop system and the Spark system, need to be designed.

Efforts have been made in the past few years towards efficient storage and processing via high-performance Cloud computing technologies. Google, as one of the leading companies in the field of big data, firstly proposed the MapReduce programming model [2] which is designed to process large amounts of distributed data efficiently. Parallel processing based on MapReduce programming is used to solve some problems within the field of RS applications such as storage, detection, retrieval and classification [3–6], etc. Although MapReduce is an efficient solution to the big data problem, there are a lot of limitations. MapReduce is only a computing architecture, and has no system for sharing and storing files. Hence, a distributed file system has to be combined into the MapReduce model to construct an effective Cloud computing environment. Over recent years, the Hadoop distributed file system (HDFS) has been used as an open source framework for processing and analysis of massive RS

data [7], which has become a highly popular solution for storing and processing a large amount of image data for analysis purposes. In order to meet the characteristics of WebGIS accessing patterns, Liu et al. [8] proposed WebGIS-based HDFS storage to support popular web applications, but their method is sensitive to small file I/O performance, and the accessing time of their method is still relatively long. Being aware of the quality of service of the DataNodes, Cloud storage from HDFS [9,10] can increase the storage utilization ratio and improve storage performance. This makes the Hadoop distributed computing model more suitable to unstable wide area network environments. Rajak et al. [11] store the outputting results in the HBase of the Hadoop system and perform parallel RS data processing. The speedup and performance show that by utilizing Hadoop, they can distribute the workload across different clusters. In the geospatial data processing field [12], the framework is also designed to enable the storage and processing of spatial and RS data in a distributed environment.

Recently, high-resolution images have proliferated due to the advanced technology in the RS field. The size of one scene of satellite image data is 300–400 MB, and even a few GB (such as for True Marble image data, where each file is 1.5 GB). Mosaic RS images may be more than 10 GB or 50 GB [13]. If we divide large-scale RS images into too many small files such as in [11,12], HDFS has to be modified to provide better storage performance. Image pyramids are hierarchy models which are simple and effective for representing large-scale satellite images at different levels of detail. In this paper, we create a pyramid model for RS images in order to achieve image multi-resolution organization. For large-scale RS images, traditional sequential pyramid building is a time-consuming task for many fast warning requirements of ocean disasters. Hence, we build a pyramid model on the Hadoop system and realize efficient storage of large-scale ocean RS images. It has good storage and extensibility, and thus higher efficiency is achieved for massive RS images.

In the retrieval field of images, some methods via clustering algorithms have also been designed to handle images or data. Using K-means clustering techniques, approaches for retrieving semantically-relevant images [14,15] were presented. However the authors indicate there are many issues of interest to be optimized in the query stage of image retrieval. Similarly, an novel image retrieval method of an improved K-means clustering algorithm [16] was proposed and experimental results demonstrated that the proposed method can achieve a good retrieval accuracy. In order to improve retrieval speed, a fast K-means algorithm [17] was designed based on a level histogram for image retrieval and a good speedup was achieved. Later, parallel algorithms of K-means [18,19] were designed on a variety of heterogeneous computing devices to meet fast demand. On the other hand, using the modern Cloud computing platform such as the MapReduce paradigm, [20], distributed computing of fuzzy C-means (FCM) [21,22] and K-means [23,24] were investigated. Speedup of retrieval using these distribution methods is improved greatly. Despite their implementation work in correctly achieving competitive purity results compared to state-of-the art clustering algorithms, FCM and K-means have some significant limitations. They require that the number of clusters be pre-determined. In practise, it is often difficult to find the right cluster number, so we often just pick a sufficiently large cluster number. This will result in situations that some natural clusters will be represented by multiple clusters found by the K-means algorithm. Besides, the K-means algorithm is in general incapable of finding non-convex clusters, which makes the K-means algorithm inadequate for complex non-linear data such as hyperspectral ocean RS images [25]. However, the mean-shift algorithm can be done quickly when the input data is small and implement clustering without pre-determining the cluster number. To solve the problem of massive non-linear ocean images and improve the performance bottlenecks in clustering, we adopt the technology of Cloud computing and modify the Mean-shift algorithm for distributed processing in the Hadoop system.

In order to design pyramid HDFS storage and improve scaleup for handling massive volumes of ocean typhoon images quickly, a high processing speed has become an indispensable requirement when providing fast retrieval. In this paper, we present an efficient storage and retrieval of massive typhoon image via a Cloud-based mean-shift algorithm. We design the distributed construction and tiles coding method to realize the pyramid-based HDFS storage. Higher throughput rate and

construction performance are achieved. Based on the pyramid HDFS storage method, an improved mean-shift algorithm for efficient retrieval is proposed by modifying the canopy algorithm [26] to adapt to the Hadoop MapReduce framework. The specific steps of the improved mean-shift algorithm are described and significant improvements in the processing performance are obtained.

The remainder of this paper is organized as follows. The efficient storage of massive ocean image via the pyramid HDFS model is designed in Section 2. Improvement of the mean-shift algorithm via a combination with the canopy algorithm in the Hadoop system is described in Section 3. Then, experimental results are discussed in Section 4, and finally conclusions are given in Section 5.

2. Efficient Storage of Massive Ocean RS Images via the Pyramid HDFS Model

With a sharp increase in the size of ocean RS images (such as in typhoons), a large size (GBs or even TBs) brings a great challenge for image pyramid building. The traditional serial processing strategies of image pyramid building like the Geospatial Data Abstraction Library (GDAL) [27] take quite a long time to process large-scale ocean images, and therefore, how to quickly build pyramid for large-scale RS images becomes an urgent problem.

In this paper, we design a distributed construction method to build an efficient pyramid storage strategy of massive images on a Cloud computing platform. Firstly, we design the overall storage flow of RS images tiles including a mapping task and reducing task shown in Section 2.1, which can show the storage process of massive ocean RS images in the Hadoop system. Then, based on the overall flow chart, we can solve how to divide large-scale images into small tiles via distributed pyramid construction algorithm in Section 2.2. Later, the small tiles can be further coded and organized to two types of data structures for distributed storage in Section 2.3. Finally, original RS images are transformed to a computable data organization, and this is restored in the HDFS for processing and retrieval.

2.1. Storage Flow of RS Images

In this paper, we design the storage flow of RS images based on the MapReduce architecture. Reading and writing tasks are divided into many smaller new tasks. Therefore we enable smaller tasks to execute independently in the distribution system, and shorten the time of the entire storage procedure. Figure 1 shows the storage flow of tile data via the pyramid HDFS model.

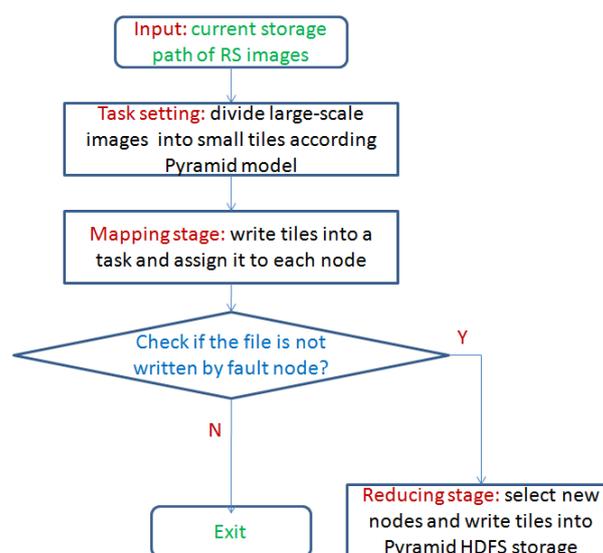


Figure 1. Storage flow of remote sensing (RS) images tiles.

From Figure 1, in our design, the input RS images are divided into small tiles according to the pyramid HDFS model in the task setting stage. Then we write tiles into smaller tasks and assign them to each node in the Hadoop system. Finally, we check whether the task is completed successfully.

2.2. Distributed Construction Method via the Pyramid Model

Before using ocean RS images, they must be divided into tiles according to the image pyramid model. After segmentation, the size of the original image is small enough to be processed and the number of level is generally not more than 20 layers when constructing the image pyramid model. In this paper, the zoom maximum level of the RS images is influenced by the resolution of the original image. Meanwhile, considering different width and height of image sizes, the maximum number of layers needs to be calculated by two aspects: width and height. The final number of layers is selected as the maximum value from two aspects. The maximum division layer algorithm (MDLA) is described in Algorithm 1.

Algorithm 1: Maximum division algorithm.

Input: *Extent* is the coordinate range of original RS images; *cellSize* is the pixel value of RS images; *tileSize* is the slice size of image tile.

Output: *level* is final number of division layers.

- 1: Input coordinate range of original RS images *Extent*, and evaluate the width and height of RS images: $W_0 \leftarrow Extent.width, H_0 \leftarrow Extent.height$;
 - 2: Evaluate resolution of *i*-th layer ($1 \leq i \leq 20$): $Res_w = W_0 / (2^i \times tileSize)$, then $i = i + 1$;
 - 3: If $cellSize.width + K \geq Res_w$ (K is a correction constant), stop evaluation and achieve the value *i* as maximum value of division layers. Or if $cellSize.width + K < Res_w$, repeat Step 2 to evaluate resolution of (*i* + 1)-th layer;
 - 4: Reassign $W_0 \leftarrow H_0$, and evaluate Res_h according to processing method in Step 2 and Step 3;
 - 5: Evaluate the maximum value $level \leftarrow \max(Res_w, Res_h)$;
 - 6: **Return** final number of division layers *level*.
-

Based on the maximum division layer algorithm Algorithm 1, we design the algorithm how to construct a pyramid HDFS model on the Hadoop system. The distributed construction algorithm of the image pyramid model is described in Algorithm 2.

Algorithm 2: Distributed construction algorithm of image pyramid model.

Input: *RS* is the source of RS images; *cols* and *rows* are number of rows and columns respectively; *CellSize* is a function to get pixel value; *tileSize* is the slice size of image tile.

Output: *AttributeData* is the attribute data; *NetData* is net data;

- 1: Read the partitioned image *RS* into HDFS and achieve the coordinate range of RS images *Extent*;
 - 2: Evaluate pixel value of RS images $cellSize = CellSize(Extent.width/cols, Extent.height/rows)$;
 - 3: Divide layers using $MDLA(Extent, tileSize, cellSize)$ in the Algorithm 1;
 - 4: Evaluate image resolution Res_i when dividing each layer *i*, and produce division data *divData*;
 - 5: If the terminal condition of MDLA is satisfied, return final number of division layers *level* and then implement Step 7;
 - 6: If the terminal condition of MDLA is not satisfied, then $i = i + 1$, and return Step 4;
 - 7: Ingest the data of current layer to HDFS with two parameters: *i*, *divData*;
 - 8: Based on two parameters in Step 7, call *SinkLevel* function to establish the pyramid HDFS model, and produce the attribute data *AttributeData* and net data *NetData*;
 - 9: If $level \geq i$, repeat Step 8. If $level < i$, the repeat stops, *AttributeData* and *NetData* are outputted into the HDFS finally;
-

The data sets in the calculation process can be stored in HDFS, which can reduce the number of accessing data, shorten the image computation time, and improve the construction speed of the pyramid image model. Therefore, it can meet the requirements of the efficient construction of the image pyramid model and fast processing.

2.3. Storage Strategy of Massive RS Images

Traditional database storage such as ArcSDE or Oracle are quite complicated for massive ocean RS images. We use HDFS to store and manage image tiles, and do not need to build additional index information, which can greatly improve accessing efficiency of massive images. Here we mainly study an efficient storage method that is based on the distributed file system in Hadoop. After partitioning the original image into many blocks as tiles, the layer organization of the pyramid model is generated by stitching them, and the pixels in one image layer are mapped to the other image layer according to the coding process by the Tile Map Service (TMS) algorithm [28], which is the specification established by the Open Source Geospatial Foundation (OSGeo). In this paper, the tile data of each partition image is not stored separately, but firstly will be assigned to one or more DataNodes in the Hadoop system. Then the range of image tile data is located by encoding value *SpatialKey* and level number *i*. Finally, data values in the metadata tiles are obtained. Figure 2 shows the coding process of the pyramid structure of image tiles.

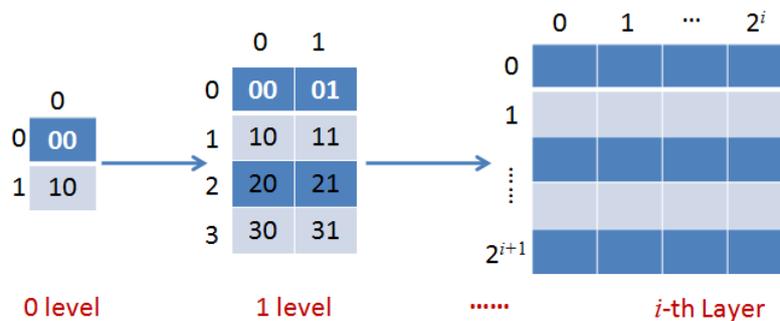


Figure 2. The image tiles coding process.

From Figure 2, the basic principle is to cut the map into tiles at the i -th layer, and store the tiles in the server according to the layer levels. When the map is magnified one level every time, every original 256×256 tile will be split into four 256×256 tiles. Based on TMS, we can find tiles through zoom level, column number and row number. Assuming the zoom level of images is n ($0 \leq n < 20$), when the image is segmented based on geographic coordinate system, the column number is $2^{n+1} + 1$, and row number is $2^n + 1$.

In the coding process, due to regularity of tiles in geographic coordinate systems, we can design an accurate index relationship which is established between the encoding value and coordinates. Through the latitude and longitude of coordinates, the tile encoding value can be obtained. On the contrary, longitude interval can be also found via the encoding value. This relationship is described by the computation functions as follows.

Assuming x and y coordinates of the image tile, x_{tile} and y_{tile} can be achieved by the accurate index relationship as follows.

$$x_{tile} = \left(\frac{lon_deg + 180}{360} \right) \times n \tag{1}$$

$$y_{tile} = \frac{(1 - \log(\tan(lat_deg) + \sec(lat_deg)))/\pi}{2} \times n \tag{2}$$

where lon_deg and lat_deg are the degrees of longitude and latitude, respectively.

Then the values of longitude and latitude can be derived from Equations (1) and (2) as follows.

$$lon_deg = \frac{x_{tile}}{n} \times 360 - 180 \tag{3}$$

$$lat_deg = \arctan \left(\sinh \left(\pi \times \left(1 - \frac{2 \times y_{tile}}{n} \right) \right) \right) \tag{4}$$

Therefore, in the process of constructing pyramid model on Hadoop system, we can determine the loaded image tiles according to the central coordinates and the level number of original RS images. Similarly, when clicking on base map, we can also fix the range of latitude and longitude according to the encoding value of image. Finally the coordinates of one pixel can be exactly located on the image tile.

The pyramid model in this paper is constructed based on MapReduce programming, and its purpose is to accelerate the process of image processing. We process the original RS images as a computable data organization and store it in HDFS. Two types of data structures are produced after processing. One class of data is attribute data of the JSON file format, and the other data is net data without head information as shown in Figure 3.

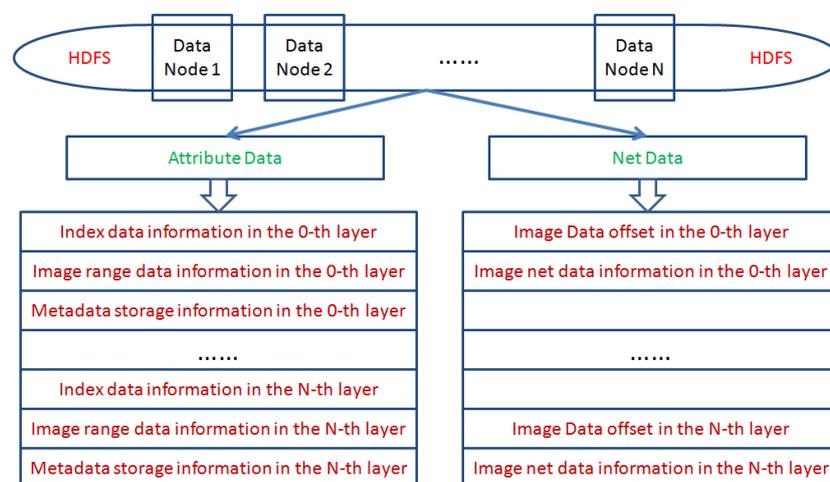


Figure 3. Two types of data for distributed storage. HDFS: Hadoop distributed file system.

From Figure 3, the attribute data includes index data, image range data, and metadata, which are respectively stored in the 0-th layer to the N -th layer. Net data includes data offset and image net data information, which is respectively stored in the 0-th layer to the N -th layer. When constructing the image pyramid model, these image data will be stored in each node of HDFS.

3. Improvement of Mean-Shift Algorithm via Canopy Algorithm in the Hadoop System

After storing the large-scale RS images into the distributed HDFS, it is necessary to study how to use two-class data effectively in ocean disasters such as for typhoon image retrieval. We have considered the mean-shift algorithm to conduct feature clustering and image retrieval. However the traditional mean-shift algorithm can not be directly used on the Hadoop platform, and it must be improved for the Hadoop platform to adapt to the MapReduce framework. In this paper, we improve the mean-shift algorithm for large-scale RS images via combing with canopy algorithm in the the Hadoop system.

3.1. Iterative Mean-Shift Algorithm

Mean-shift is a nonparametric iterative algorithm or a nonparametric density gradient estimation using a generalized kernel approach, which is one of the most powerful clustering techniques.

Given n data points $x_i (i = 1, 2, 3, \dots, n)$ in a d -dimensional space R_d , the multivariate kernel density estimate using a radially symmetric kernel (e.g., Epanechnikov and Gaussian kernels), $K(x)$, is given by,

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) \quad (5)$$

where h (termed the bandwidth parameter) defines the radius of kernel. The radially symmetric kernel is defined as,

$$K(x) = c_{k,d} k(\|x\|^2) \quad (6)$$

where $c_{k,d}$ represents a normalization constant which assures $K(x)$ integrates to 1. $k(x)$ is the profile of the kernel. The modes of the density function are located at the zeros of the gradient function $\nabla f(x) = 0$.

Then we take the gradient of the density estimator (5),

$$\nabla f(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^n (x_i - x) g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) \quad (7)$$

Based on the gradient, some further algebraic manipulation yields,

$$\nabla f(x) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^n g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) \right] \left[\frac{\sum_{i=1}^n x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x \right] \quad (8)$$

where $g(x) = -k'(x)$ denotes the derivative of the selected kernel profile. The first term is proportional to the density estimate at x (computed with the kernel $G = c_{g,d} g(\|x\|^2)$). The second term, called the mean shift vector, m , points toward the direction of maximum increase in density and is proportional to the density gradient estimate at point x obtained with kernel K .

$$\nabla m(x_i^t) = \frac{\sum_{i=1}^n x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x \quad (9)$$

The procedure of mean-shift for a given point x_i is shown in three steps as follows.

1. In order to quickly find points toward the direction of maximum increase in density, evaluate the vector of mean-shift $m(x_i^t)$ based on Equation (9).
2. Using $m(x_i^t)$ in Step 1, translate the window of density estimation from x_i to x_{i+1} : $x_i^{t+1} = x_i^t + m(x_i^t)$

- Repeat Step 1 and Step 2 until convergence, i.e., $\nabla f(x_i) = 0$ according to Equation (8).

The mean-shift clustering algorithm can be described as a practical application of the mode finding procedure, which is illustrated in Figure 4:

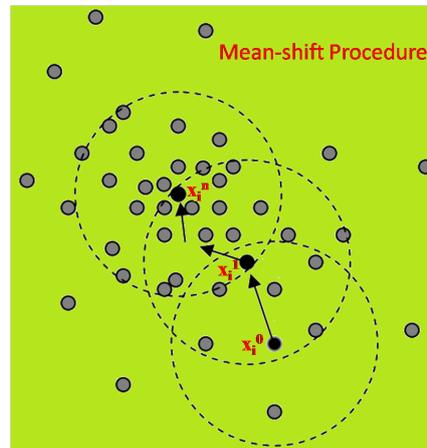


Figure 4. Finding procedure of mean-shift.

From Figure 4, starting at data point x_i , the mean-shift procedure finds the stationary points of the density function. Superscripts denote the mean shift iteration, the shaded and black dots denote the input data points and successive window centres, respectively. The dotted circles denote the density estimation windows. The set of all locations that converge to the same mode defines the basin of attraction of this mode.

3.2. Improved Mean-Shift Algorithm via Hadoop MapReduce Programming

In order to detect or forecast ocean devastating events and their effects efficiently, special features from massive data has to be extracted rapidly and precisely. In this paper, we improve the mean-shift algorithm by fusing the canopy algorithm [29] and redesign it via Hadoop MapReduce programming. The canopy generation algorithm, called canopy clustering, is a fast and specialized algorithm to deal with massive and high-dimensional data. It divides the input data points into some overlapping clusters (called canopy). Usually, points in the canopy are reclustered with higher precision algorithm, so the canopy algorithm is often used to preprocess data, which can bring much convenience to mean-shift algorithms.

In the context canopy refers to a group of similar points, or a cluster. Canopy clustering is based on two distance thresholds $T1$ and $T2$ that attempt to estimate the possible cluster centre (or canopy centre). The advantage of canopy clustering is that it gets the cluster at a very fast rate, just traversing the data to get the results. Based on the canopy clustering, we can quickly obtain the data set to estimate the number of clusters and the approximate central position. Figure 5 shows the process of creating canopy.

The canopy technique is more useful when the data set is large such as in massive ocean RS images. From Figure 5, the key idea to perform clustering is as follows.

- The RS images $RSList$ is firstly acquired from the storage system and sorted according to certain rules. $T1$ and $T2$ ($T1 < T2$) are set as initial distance thresholds.
- In order to have objectivity, select one RS image A from the $RSList$ randomly. Then calculate the distance between A and the other vectors in the $RSList$ using the distance calculation formula. Here, the distance calculation is obtained by Euclidean formula as follows:

$$d(X, Y) = \sqrt{\left(\sum_{i=1}^n (x_i - y_i)^2 \right)} \quad (10)$$

3. Create canopy for the vector which its distance d acquired in Step 2 is between $T1$ and $T2$ ($T1 < d < T2$), and remove the other vectors from the *RSList* table.
4. Repeat Step 2 and Step 3 until the *RSList* table is empty.

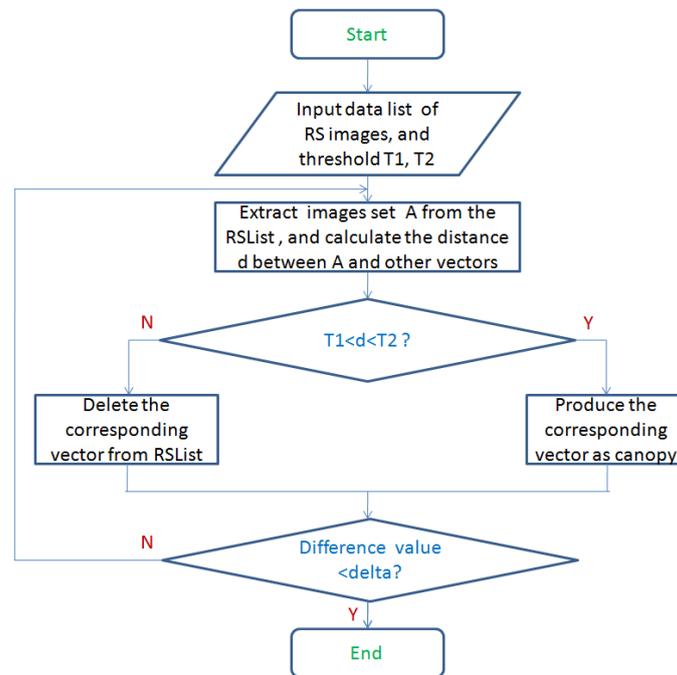


Figure 5. Flow of creating canopy.

Based on the key idea to perform canopy clustering above, we redesign the canopy algorithm based on the MapReduce parallel computing mode. Distributed canopy clustering in the Hadoop system is implemented by tasks of Map and Reduce as follows:

1. Map task: Based on RS images *RSList* from HDFS in Section 2, we randomly extract a sample vector as the center vector of the canopy, and then traverse all the sample data vector and calculate the distance from the center vector. According to the distance value, some RS images are selected to create canopy. Finally the center vectors of all canopies are the output of the Map task.
2. Reduce task: Integrating the center vectors from Map task stage in Step 1, we generate a new canopy center vector as the final output.

Before describing the improved mean-shift algorithm, we give the following parameter settings: the distance threshold $T1$ is set as the fixed radius of each window. The distance threshold $T2$ is used to determine when two canopies are merged into one. In addition, a threshold δ is set to determine whether the algorithm could stop. Figure 6 shows the flow diagram of mean-shift algorithm by modifying the canopy algorithm.

From Figure 6, the specific steps of the improved mean-shift algorithm is as follows.

1. Initially, via the canopy clustering described above, we create a canopy for each input sample data point.
2. For each canopy in Step 1, if the distance between the input value and the current canopy is less than the distance threshold $T1$, it would be classified as the current canopy. Each canopy computes its mean drift vector $m(x_i)$ in Section 3.1. A new canopy center vector (the mean drift vector) can be obtained by adding the full dimension of the points contained in each canopy divided by the number of points contained. The weight of the center vector is represented by the number of points it contains.

3. For the new canopy in Step 2, we calculate the distance between them. When the distance is less than the threshold $T2$ (as convergence condition in Section 3.1), they would be classified as a canopy, while the corresponding attributes are also superimposed.
4. When the difference between two values of canopy which are evaluated before and after iterations is less than a given threshold δ , the algorithm ends as shown in Figure 6.
5. After the algorithm is over, the number of remaining canopy is outputted as the number of clusters.

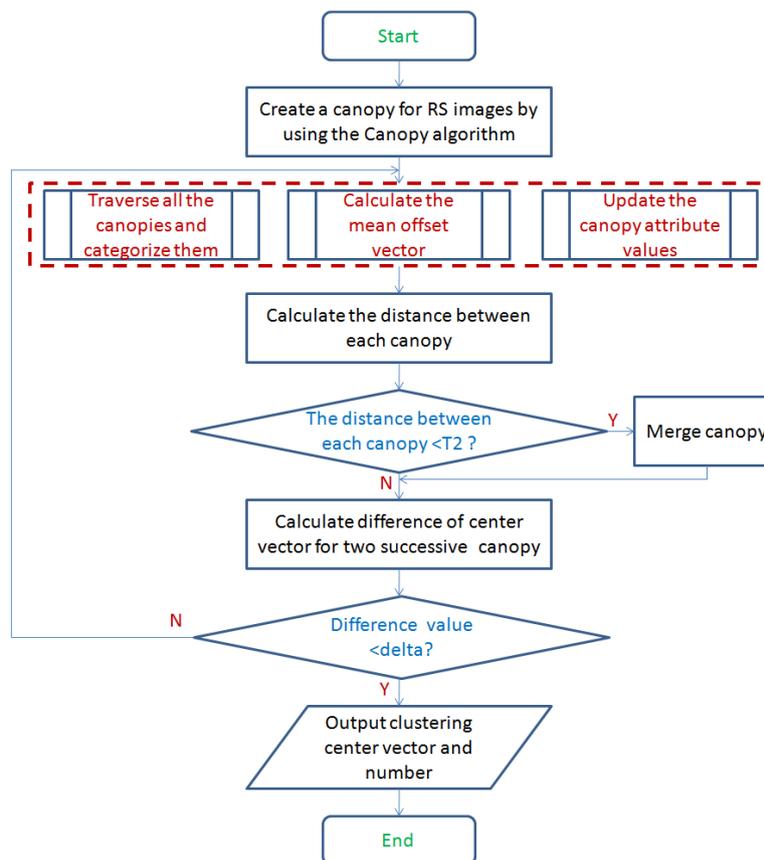


Figure 6. Flow diagram of mean-shift algorithm by modifying the canopy algorithm.

Based on the improved mean-shift algorithm above, the distributed mean-shift algorithm fused with the canopy algorithm can be designed and implemented to adapt to the Hadoop MapReduce framework as follows.

1. In the Map phase, based on values of $T1$ and $T2$ we use the mapper function to classify the canopy created for each RS image. Then we use the new center vector to update the canopy, and finally output the results to the reducer function.
2. In the Reduce phase, based on the output results from Step 1, we use the reducer function to integrate them and then return them to driver in Step 3.
3. The driver would decide whether to continue the loop or exit by comparing the result of reducer with threshold δ .

The main work of the algorithm's Mapper, Reducer, and Driver can be seen in detail in Figure 7.

After clustering the significant and differential features of ocean RS images (such as rotation of typhoons in Cloud systems), the feature matching for RS images retrieval would be performed. In this paper, we use the Euclidean formula mentioned in this section to evaluate the distance difference. We calculate the distance of the RS images feature vector to be retrieved from each cluster center vector, and output the closest category scene image as the retrieval result.

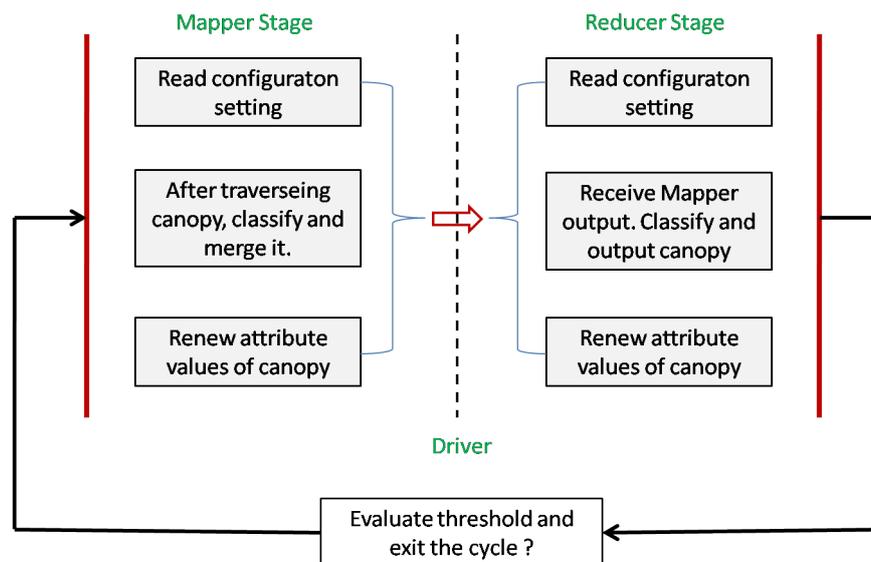


Figure 7. Storage structure of image data in MapReduce processing.

4. Results

We have implemented the efficient storage and retrieval of massive ocean image via the Cloud-based mean-shift algorithm. The servers are running Centos OS and each has 32 GB memory. The single processor speed is 3.2 GHz using AMD Opteron 6-Core 4180. Therefore, each server has six processors and each processor has six cores. The program used for the experiments is implemented in the Scala programming language. In addition, Apache Hadoop 2.6 is used as the framework for distributing the computations on the cluster. In the experiments, the programming framework is used to provide the needed connectivity. Its component packages allow access to the MapReduce and the HDFS. All the test data is from the online satellite images of typhoons in the ocean. The images are 22 layers and 3 to 9 layers with total of 87,311 tiles are selected. The tile is large up to 164 KByte and is the smallest, 3.66 KByte.

In the test, the easy URL way is used by TMS to request typhoon image tiles, and the request method with multiple parameters is not considered here. Given an $http://URLTyphoonLayerName/z/x/y.png$, URL is the server net address, $TyphoonLayerName$ is the layer name of typhoon, z represents the zoom level of image, and (x, y) shows the image tiles coordinates.

4.1. Efficient Storage on Hadoop System

Firstly, the construction algorithm is respectively applied into Hadoop stand-alone mode and Hadoop distributed mode (using MapReduce programming model). Typhoon data size in the experiment is 100 MB, 1 GB, 10 GB, 100 GB, and comparison results are shown in Figures 8 and 9.

From Figures 8 and 9, when the data size of typhoon RS images is small, the throughput rate of stand-alone mode is higher than that of the distributed mode, however construction performance of its image pyramid is lower than that of the distributed mode. When the typhoon data size is increasing, both the throughput rate and construction performance in the pyramid distributed model are higher. In particular, when the typhoon data size of RS images is massive, performance of our method is more obvious over the stand-alone mode, which shows the achievement of our storage method.

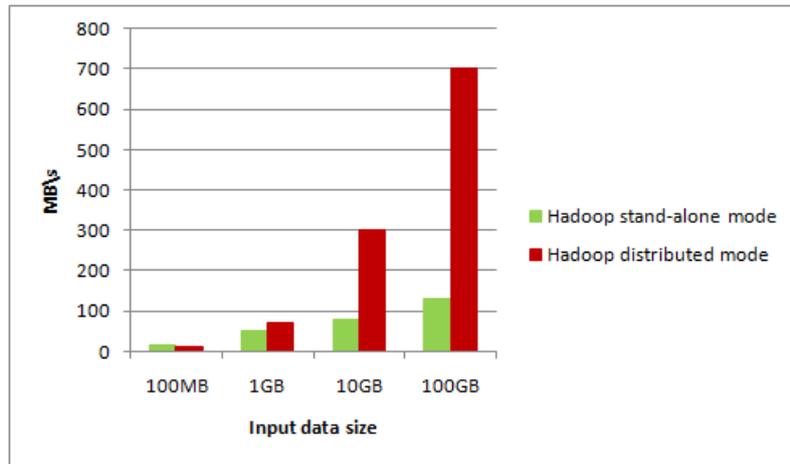


Figure 8. Throughput rate comparison between two modes.

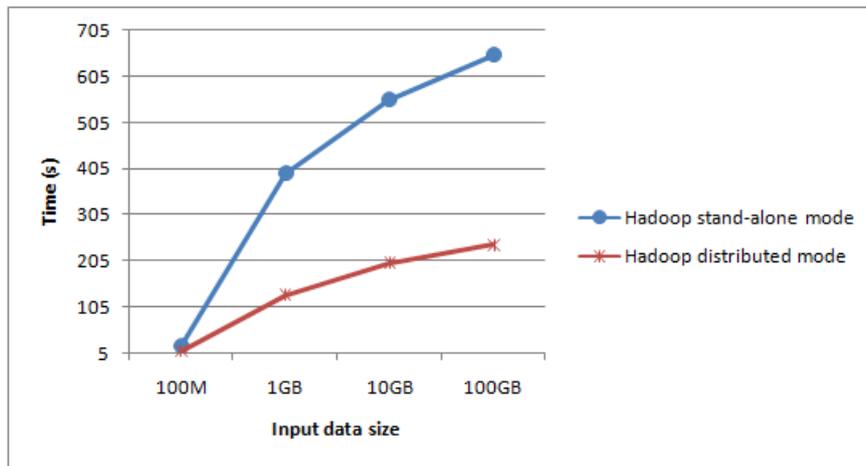


Figure 9. Construction performance comparison between two modes.

In addition, we select 1000, 3000 and 8000 tiles to test the reading performance of the typhoon data. Figure 10 shows the reading time comparison of different tile data file when using traditional Hadoop HDFS storage, WebGIS-based HDFS storage [8] and our method.

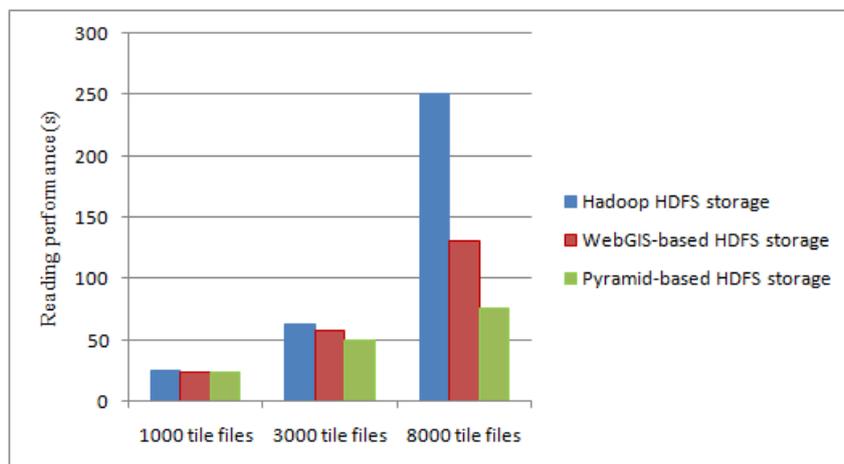


Figure 10. Reading time comparison of different tile data files.

From Figure 10, in the Hadoop HDFS storage, a large amount of block information is required to be read for small tile files. The reading data process is constantly jumping between many DataNodes, thus reducing the reading performance. Using the index file, WebGIS-based HDFS storage looks up the corresponding data file and reads it. It not only reduces the number of block information, but also reduces the jumping number between DataNodes when reading data. However, there is further space for improvement of the accessing time of their method. Based on the parallel framework of MapReduce, the reading data in our method is directly from each DataNode of Hadoop using a predefined pyramid storage structure. In this process, the Map task and the Reduce task take up only a little amount of time. When the amount of typhoon data is small, the performance improvement of the reading efficiency is not obvious, but with the increase in the number of files, the performance has been significantly improved.

4.2. Retrieval of Massive Ocean RS Images

Based on efficient storage via our method, we implement a high performance retrieval for large-scale ocean typhoon image. Test and analysis are carried out from three aspects: speedup, scaleup rate and time performance as follows.

The speedup is defined as the ratio of the retrieval time of Hadoop stand-alone mode to the retrieval time of Hadoop distributed mode under the same size images set. In this experiment, we firstly keep the size of the input image set unchanged, and gradually increase the number of DataNodes from one to ten to show the speedup values. Figure 11 shows the retrieval speedup ratio of five groups of different size image sets (1 G, 2 G, 4 G, 8 G, 16 G) on different DataNodes in the Hadoop distribution system.

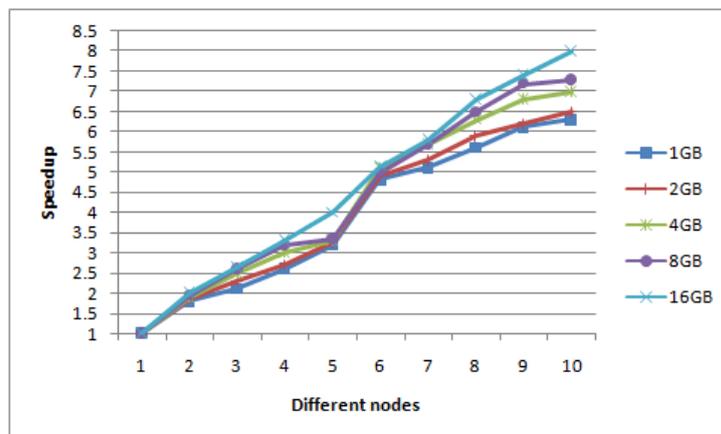


Figure 11. Speedup ratio in five groups of different size image sets.

Ideally, the speedup ratio of the system should increase linearly with the increase of the number of DataNodes, and the efficiency should always remain unchanged. But in reality, due to the I/O, communication cost and loading balance, the speedup does not increase linearly. However from Figure 11, the speedup ratio of our algorithm is close to linear. When the typhoon data set is larger, the curve is closer to linear and the speedup performance of the algorithm will get better and better.

Scalability means the ability of our method to handle a growing amount of image sets, which also refers to the capacity of distributed system to handle an increase in load. Scaleup is defined as the ratio of the computing time for processing data on one DataNode to the computing time for processing $m \times$ data on m DataNodes. We have performed scalability experiments with the increasing number of DataNodes, and reported the scaleup values. In our experiment, our method has good robustness and scalability. New nodes could be added into the Hadoop system at runtime to meet dynamic requirements, thus obtaining better performance in most cases and providing elastic computing as needed. Scaleup is demonstrated by using three group typhoon datasets to implement and test it.

The number of data points in the datasets used are as follows: 1 G, 2 G, 4 G, 8 G in the first group, 2 G, 4 G, 8 G, 16 G in the second group, and 4 G, 8 G, 16 G, 32 G in the third group. Data in each group is implemented on 1 node, 2 nodes, 4 nodes, 6 nodes, 8 nodes and 10 nodes, respectively. Figure 12 shows the test result of scaleup performance.

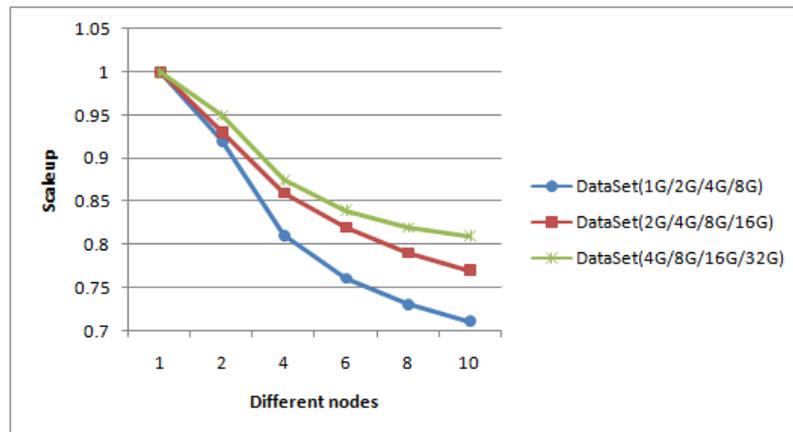


Figure 12. Scaleup in three group datasets.

From Figure 12, for the same dataset, when the number of nodes increases, the scaleup is reasonably reduced. This is because when the number of nodes increases, the communication cost between the nodes will also linearly increase. When the data size increases, the execution time of the algorithm almost linearly increase, which is in agreement with true computation and shows our method scales very well.

Also, we compare different running time for typhoon image retrieval on different sizes of the file and different DataNodes to highlight the effectiveness of our method. Firstly, the total typhoon dataset is divided in subsets with the respective sizes of the original file: 10%, 50%, 80% and 100%. Next, the data subset with 10% of original data is used for retrieval, that is, nearly 10 GB of data. Similarly, 50%, 80% and 100% of total dataset are used to implement the improved mean-shift algorithm. Table 1 shows average runtime using the improved mean-shift algorithm.

Table 1. Time and gain when implementing our algorithm on different sizes of the file.

Size of File	Time to Use One Node (m)	Time to Use Ten Nodes (m)	Gain (%)
10%	4.95	4.45	11.24%
50%	18.74	8.64	116.89%
80%	34.02	10.53	223.08%
100%	47.06	12.21	285.42%

From Table 1, using half of available data it is possible to observe an improvement about 117% in the retrieval time. Even with 100% of data, the time necessary to perform a task is improved by approximately 285%, while keeping same retrieval accuracy both in training and testing. The performance would be more greatly improved with the increase sizes in the typhoon file size.

5. Conclusions

We have implemented an efficient storage and retrieval of massive ocean image via the Cloud-based mean-shift algorithm. From experiment and results, throughput rate and construction performance of pyramid-based HDFS storage are much higher. Especially, when the data size of RS images is massive, the advantage of our method is more obvious than the stand-alone mode. Also, our method has a shorter reading time of different tile data file over traditional Hadoop HDFS storage and WebGIS-based HDFS storage [8]. Based on the pyramid HDFS storage, the improved mean-shift

algorithm for massive image retrieval is presented. Tests and analysis were carried out from three aspects: speedup, scaleup rate and time performance. The speedup ratio of our algorithm is close to linear, and our method also has good robustness and scalability. It is observed that a significant improvement in the retrieval time is achieved finally.

We have demonstrated that the proposed methods have achieved some good advantages in storage and retrieval, but many exciting directions remain to be explored. In future work, we plan on implementing some practical applications in ocean disaster warnings such as typhoon track prediction based on our method.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (No. 41671431), the Ability Construction Project at the Local University of Shanghai Science and Technology Commission (No. 15590501900), the Program for Professor of Special Appointment (Eastern Scholar) at the Shanghai Institutions of Higher Learning (No. TP2016038), the Youth Science and Technology Project of Shanghai Ocean University (No. A2-0203-00-100216), and the Doctoral Start-up Fund for Scientific Research of Shanghai Ocean University (No. A2-0203-00-100346). We thank Yuhao Yang for analyzing and testing the image storage efficiency.

Author Contributions: The work was conducted under the cooperation of authors. Mengzhao Yang conceived the algorithm, and wrote the paper; Haibin Mei designed the experiments and verified the results; Wei Song guided the study and reviewed the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, C.A.; Gasster, S.D.; Plaza, A.; Chang, C.I. Recent developments in high performance computing for remote sensing: A review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2011**, *4*, 508–527.
2. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. In Proceedings of the Conference on Symposium on Operating Systems Design & Implementation, San Francisco, CA, USA, 6–8 December 2004; pp. 107–113.
3. Tesfamariam, E.B. *Distributed Processing of Large Remote Sensing Images Using MapReduce—A Case of Edge Detection*; LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2011; pp. 1–10.
4. Lv, Z.; Hu, Y.; Zhong, H.; Wu, J.; Li, B.; Zhao, H. Parallel K-means clustering of remote sensing images based on MapReduce. In Proceedings of the International Conference on Web Information Systems and Mining, Sanya, China, 23–24 October 2010; pp. 162–170.
5. Venkatraman, S.; Kulkarni, S. MapReduce neural network framework for efficient content based image retrieval from large datasets in the cloud. In Proceedings of the International Conference on Hybrid Intelligent Systems, Gammarrh, Tunisia, 4–6 December 2013; pp. 63–68.
6. Almeer, M.H. Cloud Hadoop map reduce for remote sensing image analysis. *J. Emerg. Trends Comput. Inf. Sci.* **2012**, *3*, 637–644.
7. Pan, X.; Zhang, S. A remote sensing image cloud processing system based on Hadoop. In Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems, Hangzhou, China, 30 October–1 November 2012; pp. 492–494.
8. Liu, X.; Han, J.; Zhong, Y.; Han, C.; He, X. Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS. In Proceedings of the IEEE International Conference on Cluster Computing and Workshops, New Orleans, LA, USA, 31 August–4 September 2009; pp. 1–8.
9. Song, G.H.; Chuai, J.N.; Yang, B.W.; Zheng, Y. QDFS: A quality-aware distributed file storage service based on HDFS. In Proceedings of the IEEE International Conference on Computer Science and Automation Engineering, Shanghai, China, 10–12 June 2011; pp. 203–207.
10. Patel, A.; Mehta, M.A. A novel approach for efficient handling of small files in HDFS. In Proceedings of the IEEE Advance Computing Conference, Bangalore, India, 12–13 June 2015; pp. 1258–1262.
11. Rajak, R.; Raveendran, D.; Bh, M.C.; Medasani, S.S. High resolution satellite image processing using Hadoop framework. In Proceedings of the IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore, India, 25–27 November 2015; pp. 16–21.
12. Giachetta, R. A framework for processing large scale geospatial and remote sensing data in MapReduce environment. *Comput. Graph.* **2015**, *49*, 37–46.

13. Lin, F.C.; Chung, L.K.; Wang, C.J.; Ku, W.Y.; Chou, T.Y. Storage and processing of massive remote sensing images using a novel cloud computing platform. *Gisci. Remote Sens.* **2013**, *50*, 322–336.
14. Murthy, V.S.V.S.; Vamsidhar, E.; Kumar, J.N.V.R.S.; Rao, P. Content based image retrieval using hierarchical and K-means clustering techniques. *Int. J. Eng. Sci. Technol.* **2010**, *2*, 209–212.
15. Ho, J.M.; Lin, S.Y.; Fann, C.W.; Wang, Y.C.; Chang, R.I. A novel content based image retrieval system using K-means with feature extraction. *Comput. Sci. Inf. Syst.* **2012**, *9*, 785–790.
16. Lv, M.L. Novel image retrieval method of improved K-means clustering algorithm. *Comput. Sci.* **2013**, *40*, 285–288.
17. Lin, C.H.; Chen, C.C.; Lee, H.L.; Liao, J.R. Fast K-means algorithm based on a level histogram for image retrieval. *Expert Syst. Appl. Int. J.* **2014**, *41*, 3276–3283.
18. Zhu, H.; Zhang, Q.; Ren, X.; Jiao, L. Parallel fast global K-means algorithm for synthetic aperture radar image change detection using OpenCL. In Proceedings of the Geoscience and Remote Sensing Symposium, Milan, Italy, 26–31 July 2015; pp. 322–325.
19. Yin, C.; Zhang, S. Parallel implementing improved K-means applied for image retrieval and anomaly detection. *Multimed. Tools Appl.* **2016**, 1–17, doi:10.1007/s11042-016-3638-1.
20. Mmel, R. Google's MapReduce programming model-revisited. *Sci. Comput. Program.* **2008**, *70*, 1–30.
21. Ludwig, S.A. MapReduce-based fuzzy C-means clustering algorithm: Implementation and scalability. *Int. J. Mach. Learn. Cybern.* **2015**, *6*, 923–934.
22. Phu, V.N.; Dat, N.D.; Tran, V.T.N.; Chau, V.T.N.; Nguyen, T.A. Fuzzy C-means for english sentiment classification in a distributed system. *Appl. Intell.* **2016**, 1–22, doi:10.1007/s10489-016-0858-z.
23. Issa, J. Performance characterization and analysis for Hadoop K-means iteration. *J. Cloud Comput.* **2016**, *5*, 3–17.
24. Yang, M.Z.; Mei, H.B.; Huang, D.M. An effective detection of satellite image via K-means clustering on Hadoop system. *Int. J. Innov. Comput. Inf. Control* **2017**, *13*, 1037–1046.
25. Ye, K.; Jiang, X.; He, Y.; Li, X. vHadoop: A scalable Hadoop virtual cluster platform for MapReduce-based parallel machine learning with performance consideration. In Proceedings of the IEEE International Conference on Cluster Computing Workshops, Beijing, China, 24–28 September 2012; pp. 152–160.
26. Ghuli, P.; Shukla, A.; Kiran, R.; Jason, S.; Shettar, R. Multidimensional canopy clustering on iterative MapReduce framework using Elefig tool. *IETE J. Res.* **2015**, *61*, 14–21.
27. Meng, C.; Zhang, Z.; Zhang, L.; Li, J. Method of building image Pyramid model Based on GDAL and NetCD. *Hydrogr. Surv. Charting* **2012**, *32*, 49–51.
28. Xu, D.; Yuan, Z.; Yu, T. The research of remote sensing image segmentation and release which are based on Tile Map Service. In Proceedings of the International Symposium on Geomatics for Integrated Water Resources Management, Lanzhou, China, 19–21 October 2012; pp. 1–4.
29. McCallum, A.; Nigam, K.; Ungar, L.H. Efficient clustering of high-dimensional data sets with application to reference matching. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; pp. 169–178.



© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).