Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 129 (2018) 238–247

Procedia Computer Science

www.elsevier.com/locate/procedia

2017 International Conference on Identification, Information and Knowledge in the Internet of Things

An improved distributed storage and query for remote sensing data

Weipeng Jing[a], Dongxue Tian[a]

[a]*College of information and computer engineering, Northeast Forestry University, Harbin 150040, China*

Abstract

With the rapid development of information technology, the amount of remote sensing data is increasing at an unprecedented scale. In the presence of massive remote sensing data, the traditional processing methods have the problems of low efficiency and lack of scalability, so this paper uses open source big data technology to improve it. Firstly, the storage model of remote sensing image data is designed by using the distributed storage database HBase. Then, the grid index and the Hibert curve are combined to establish the index for the image data. Finally, the method of MapReduce parallel processing is used to write and query remote sensing images. The experimental results show that the method can effectively improve the data writing and query speed, and has good scalability.

Selection and peer-review under responsibility of the scientific committee of the 2017 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI2017).

Keywords: remote data; distribute storage; data query; HBase; mapreduce ;

1. Introduction

In recent years, with the rapid development of remote sensing technology, the number of image data (such as aerial photographs, satellite remote sensing images, ground photographs, etc.) that people can obtain is increasing in geometric progression[1]. The rapid storage and query of remote sensing images play an important role in the field of ground observation, military and exploration. How to efficiently store, organize, manage and publish, improve processing ,and distribution efficiency in front of massive image data has become an urgent problem to be solved.

The traditional management system based on the file system[2] takes advantage of the read and write advantages of the file system, but doesn't guarantee the integrity of the data and the low efficiency of processing when multiple users concurrent access. In [3], a method of using Oracle database to manage image data is proposed, which solves the problem of integrality and consistency of data effectively. However, in the presence of large amounts of image data, this method is limited by hardware devices of single node so there are many problems such as single node fault, scalability and low query efficiency. With the development of open source big data technology, some methods of using large data technology to process remote sensing images have been proposed [4]. In [4], the Hadoop platform is used to

E-mail address: nefujwp@163.com

Selection and peer-review under responsibility of the scientific committee of the 2017 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI2017).
10.1016/j.procs.2018.03.071

store the vector space data. This method effectively solves the problems of single node failure and lack of extensibility. However, the problem of generating a lot of small files in the process of storing data is not considered, which results in a large amount of memory being used to increase the burden on the primary node. The [5] based on [4] proposed a distributed Key-Value storage model to manage massive image data and store small files into a large data file, which effectively solves a large number of small file problems. But, it doesnt describe of each layer of image metadata, thus increasing the data retrieval time. In [6], metadata is designed for each layer of images and MapFile is used to merge the tiles which generated during the storage process, thus improving the access capability of the image data. But the above methods are the image data stored directly on the HDFS. Due to the HDFS does not support any position of the document changes, not suitable for small throughput and low latency operation, so the above method cant update the data of any location directly and the query time is too long.

In order to solve the above problems, the use of non-relational database HBase remote sensing image data storage has been favored by domestic and foreign scholars. In [7], the processed image data is stored in HBase, which can update the data at any position to solve the multi-temporal problem stored in the remote sensing image data, but does not take into account the insertion efficiency of the data. In [8], the paper designs a hierarchical index structure based on Key-Value, which maintains high insertion capacity while maintaining high fault tolerance and high availability. But the hierarchical index structure increased the storage space of the index, when the amount of data is not conducive to management. In order to reduce the index space, the literature [9] will be quad-tree and grid index as the main and secondary index, the overall index space needed is smaller and has high performance in the range query, but the query time is longer in this way. In order to reduce the query time, the literature [10] designed a two-level index mechanism to reduce the query time while supporting high-level access to the geographical location. In [11], the quadratic index is realized by using the coprocessor to realize the distributed real-time query of weather data based on HBase, which effectively reduces the query time. The above methods are using two level index to reduce spatial data query time, but the study of remote sensing image data are few and rarely consider the remote sensing image itself to optimize the characteristics of HBase storage model.

In this paper, the storage model of remote sensing image is constructed by using the storage advantages of HBase for unstructured data, and then a distributed storage and query method based on HBase is proposed.

2. Storage Model Design of Image Data

In order to realize the distributed storage of image data, the HBase model of image data storage is designed on the basis of traditional tile Pyramid image data storage. The processing efficiency of the image data is improved by indexing and metadata describing the image data. In addition, through the design of the filter family to achieve the purpose of screening data to improve the accuracy of the query.

2.1. Division of image data

The most common method of image data block organization is to create a pyramid model of grid image with multi-scale hierarchical structure features, so that users can access image blocks of specific resolution levels and specific spatial regions as needed [12]. The traditional magnification method [13] divides the original image with the size of $2^n * 2^n$ resolution. By improving the above method, this paper can make the image of any resolution at different levels.

Assuming that the original image has a resolution of $n * m$ and a tile size of $2^p * 2^p (p = 0, 1, 2, ...)$, then the number of layers needed to build in Pyramid is $L = min(L_1, L_2)$.

Where L_1 needs to be meet:

$$\frac{m}{2^{L_1}} \leqslant 2^p (L_1 = 0, 1, 2, 3, ...) \tag{1}$$

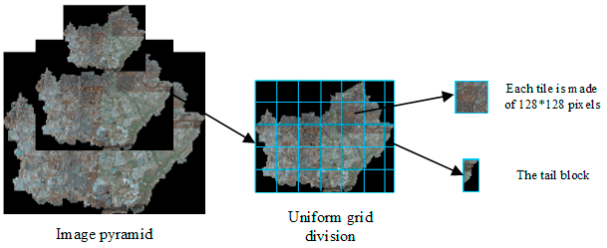

Fig. 1. Two classification model.

Similarly, L_2 needs to be meet:

$$\frac{m}{2^{L_2}} \leqslant 2^p (L_2 = 0, 1, 2, 3, ...) \tag{2}$$

After obtaining different levels of image layers then the uniform grid method is used to divide each image. In the process of tile division if there is exist less than the pixel of the "tail block", it can be make up first then split it again. Where the size of "tail block" $TialX$ and $TialY$ can be obtained by the following formula.

$$TialX = 2^{-k} * n - (2^{-k} * n * mod(2^p)) \tag{3}$$

$$TialY = 2^{-k} * n - (2^{-k} * m * mod(2^p)) \tag{4}$$

The k represents the image layer number where the tile data is located, n and m represent the resolution of the original image, mod is the residual function and 2^p is the size of the tile. In this article $2^p = 128$, which is the default block size when ArcSDE creating an image pyramid.

It can be seen from Fig.1 that the image data is hierarchized and any layer is selected for meshing, When the "tail block" appears, we can get the size of the "tail block" by formulas (3) and (4) then fill it up.

2.2. Remote sensing data storage model

Based on the storage characteristics of distributed database HBase, the grid ID and Hilbert curve are combined as the index and the description information of each tile is taken as value. The composition $< key_m, value_m >$ pairs are stored in different columns.

The image data storage model is shown in the above table. The metadata area in the table stores the metadata of each layer of the image for quickly obtaining the description of the layer image at the time of retrieval to reduce the query time. The main data area is used to store all the tile information of the image data. Among£the GeoInfo column family is used to store description information of the tile data and the Filter column family is used to store the data filtering information.

2.2.1. Metadata design

Image data stored on HDFS, which usually stores information of metedata in the memory of the master node [14]. In order to improve the storage and query efficiency of data, this paper stores the metadata information of each layer in the HBase table so that the description information of the layer image can be obtained quickly.

Table 1. Image data storage model.

RowKey			ColumnFamily:GeoInfo	ColumnFamily:Filter	
			metadata area		
-/-	level	x	y	null	null
rowkey1	v1	v2	v3	null	null
			master data area		
-/-	value	GridID	longitude	MidTileX	MidTileY
rowkey2	v4	v5	v6	v7	v8
rowkey3	v9	v10	v11	v12	v13

The metadata of each layer includes four pieces of information, the layer number k of the image data, the resolution $X = n * 2^{-k}$ and $X = m * 2^{-k}$ of the tile data, the latitude and longitude range of the layer image data, the latitudinal span $TailS\ panX$ and the longitudinal span $TailS\ panY$ of the true coordinates of the layer tile. Where the $TailS\ panX$ and $TailS\ panY$ of the tile can be obtained by the following formula.

$$TailS\ panX = \frac{X_{max} - X_{min}}{2^{L-k}} \tag{5}$$

$$TailS\ panY = \frac{Y_{max} - Y_{min}}{2^{L-k}} \tag{6}$$

Among them, (X_min, Y_min) and (X_max, Y_max) represent the initial latitude and longitude coordinates of the image data, K represents the image layer where the tile is located and L represents the total number of layers of the image data Pyramid.

2.2.2. Index structure

This paper aims to improve the efficiency of data query by constructing an index of each layer of image data. In HBase, the RowKey for each row is indexed. In order to support multi-conditional queries, the most common method is to splice the query conditions into the RowKey, but the index space required for this method increases as the query condition increases and this sorting method results in a lot of image data space position jump, which increasing the query time.

In this paper, we use the Hilbert curve to fill the grid of each image £so that the two-dimensional spatial data into one-dimensional space, which ensures the spatial proximity of the data. Because RowKey is in dictionary sorting stored in HBase, we need to convert the Hilbert value. The final RowKey value is:

$$RowKey = k + L(L(\varphi)) + L(\varphi) + \varphi \tag{7}$$

Where k represents the current layer of the remote sensing image, $L(x)$ is the length of the current x,φ is the Hibert code for the current grid, and "+" is used to segment each string. This method reduces the space required for the index while maintaining the proximity of the image data space.

The improved index structure is shown in Figure 2. Figure 2 (a) shows the position of the tile data in the grid; Figure 2 (b) shows the value of the improved HID obtained by Hilbert filling; Figure 2 (c) shows the storage location of the 0th tile in the HBase table.

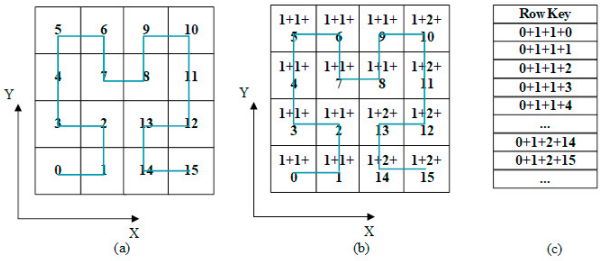

Fig. 2. Index and HBase table mapping.

Table 2. Design of FilterFamily.

RowKey	Time Stamp	FilterFamily: Filter	
		column	value
rowkey1	t1	MidTileX	value1
rowkey1	t1	MidTileY	value2

As can be seen from Figure 2 (c) , the data stored in the HBase table are arranged in descending order according to the value encoded by Hilbert. This ensures that when the region is divided, the image data adjacent to the spatial location can be stored together. It can be seen that the improved index has a high spatial proximity in the HBase table.

2.2.3. Design of filter column family

As the index design is not add the necessary conditions, for which we designed a filter column used in the image data retrieval filter out irrelevant information that to achieve the data in the server-side filtering. The median latitude and longitude coordinates of the current tile are stored in the filter column, which values are obtained from the following equations (8) and (9).

$$MidTialX = X_{min} + \frac{x+1}{2} TileS\,panX \tag{8}$$

$$MidTialY = Y_{min} + \frac{y+1}{2} TileS\,panY \tag{9}$$

Among them, (X_{min}, Y_{min}) represent the starting latitude and longitude coordinates of the image data, (x, y) represents the grid coordinates where the tile data is located and $(TileS\,panX, TileS\,panY)$ is the vertical and horizontal coordinate span of the image tile. The logical view stored in HBase is shown in Table 2.

3. Parallel processing of image data

In this paper, a method of image data writing and query based on MapReduce is designed, which improves the efficiency of image data processing by parallel processing of image data after hierarchical block processing.

3.1. Parallel writing of image data

MapReduce is Hadoop's open source computing framework, which follows the principle of partitioning and through the migration calculation can let each machine as soon as possible access and processing data.One of the

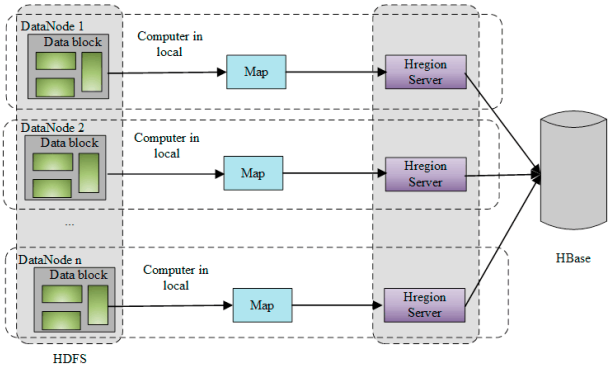

Fig. 3. Parallel frame of image data storage.

biggest features of HBase is that it can be combined with MapReduce. In the process of storing data, we only need to implement the setup method and the map method. Where the setup method is responsible for initializing the data information and sharing the data information and the map method is used to write the data to HBase. The specific process shown in Fig.3.

As can be seen from the above figure that the most of operations in this process are performed on the local node where the tile data is stored and only a small amount of information is exchanged between the data node and the master node. The specific algorithm pseudo-code is as follows:

Algorithm 1 parallel region query of image data

Input: All the information of the image data
Output: HBase data table

```
setup (Mapper.context)  init GridID[][] and Hilbert[][] ← compute(by the level)
init args[] ← context.getConfiguration() // Initialize the table information by passing the information through context.
map (key, value, context) {
    if (level < count) {
        RowKey = getRowKey(key) // Get the RowKey of the tiles
        ImageValue = value.toBASE64() // Convert the picture to BASE64 encoding
        Put put = getPut(args[], ImageValue) // Store the tile information into the put
        context.write ( RowKey, put) // Write to HBase
    }
}
```

3.2. Parallel query of image data

Due to HBase stores data in different regions and each region can be loaded by a region server, so we can use MapReduce parallel processing method to query the image data. In the query process, each node only need get the data that meets the query condition under the current region and finally writes the data directly to the HDFS through the map function.

In the area query, we first get the layer of the image to be searched according to the input information. Secondly, the specific description of the tile is obtained according to the metadata information of the image layer to be read. Then, according to the query range of the input spatial data and map it into the array of the layer grid ID, which we can get the corresponding RowKey values in the grid range for HBase retrieval identification. Finally, by redesigning the Map function in the MapReduce method to query the data in the range, the overall algorithm flow is as follows.

Algorithm 2 Parallel query of image data

Input: (x1, y1), (x2, y2) of the area query;
Output: All tile data in this layer image query range;

Step 1: get the iamge level // user input

Step 2: GridID[x][y] ← compute(the level ; the metadata information) // Calculate the latitude and longitude coordinates of the grid storage by calculating the grid information of the layer image according to the layer number.

Step 3: BinarySearch() ← Calculate x and y according to the input query range // According to the input query range, using the binary search to get the corresponding grid coordinates.

Step 4: get start RowKey and stop RowKey ← Hilbert[][] // According to the array of Hilbert values corresponding to the layer grid that the maximum and minimum values are obtained and converted into the starting and ending row keys.

Step 5: map (key , value)

if(fit(getFilter(value), search(x1,y1,x2,y2))) ← SingleColumnFamily( ) // According to set the single-column filter to determine whether this line of data is filtered.

key = getrowkey(value)

value = getvalue (image_value)

write (key, value)

Step 6: end

4. Experimental results

In the experiment, three high-resolution image data with a rate of 15 meters in Landsat8 8 band were selected. After processing, the size of image data was 700MB, 1.72GB and 2.64GB respectively. The parallel storage and query tests of the three images are carried out to verify the efficiency of the proposed method.

4.1. Experimental environment

In this experiment, the virtual software XenServer6.2 will be three dawn I450-G10 tower server (InterXeon E5-2407 quad-core 2.2 GHz processor, 8GB memory) virtual into 9 hosts, an HP Compaq dx2308 (Intel Pentium E2160 1.8 GHz Processor, 1GB memory) as master. Each virtual machine is equipped with ZooKeeper 3.4.8 version, HBase version 0.98.23, Hadoop version 2.7.2 and in Cenos6.4_final (kernel 2.6.32) system to build Hadoop cloud cluster that the HDFS total size 450GB.

4.2. Image data storage experiment

In the test of remote sensing image data storage, we divide the three images into blocks and then store the images in 9 nodes respectively. It can be seen from Table 3 that the write rate is increased every second when the amount of data increases, because the write of the unit data amount is reduced, but the overall stability is maintained. When the amount of data increases, HBase writes data to occur region segmentation operations, which is the main reason for the time-consuming.

In order to verify the effect of the number of nodes on the effect of data writing, we use different nodes to store the three images respectively. As can be seen from Fig. 4, the storage time of the data decreases as the number of nodes increases. When the amount of data is larger, the storage time is reduced obviously. When the number of nodes is less, the data import time is longer, because a single node may store several region, which results in heavy load. It can be seen that the time variation of image1 from 1 node to 4 nodes is most obvious and after 6 nodes is almost flat.This is

Table 3. the three images take times.

Image name	Image size / pixel	All time consuming /s	Write rate /(MB $\cdot s^{-1}$)
image1.tiff	18595*13158	83	8.434
image2.tiff	24576*25088	209	8.452
image3.tiff	27648*34304	307	8.805

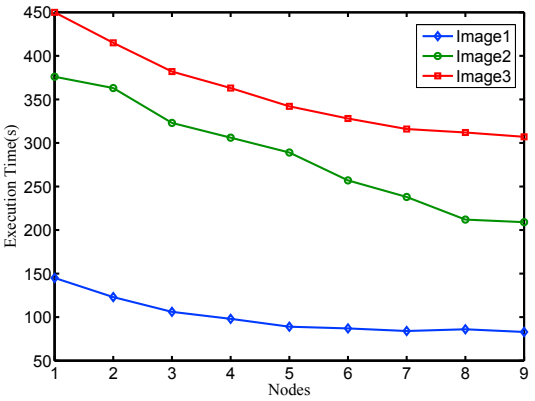

Fig. 4. Data import time of different nodes.

because the image1 data volume is small and the HBase table needs to be divided less. When the table is not cutting, the region has only a few fixed nodes, resulting in an increase in the number of nodes, but the time required to write the data is not noticeable.

4.3. Image data query experiment

We use MapFile, MySQL Cluster and this paper designed based on MapReduce HBase (MRHBase) for comparative analysis. We choose the largest resolution layer in image3 as the query object, which takes the 10%-80% of the whole image as the input query range and the query time of the test data is as follows.

Figure 5 shows that the query time of MRHBase in different regions increases most slowly and MapFile grows fastest. When the query area is less than 20%, the MySQL cluster query time and HBase cluster parallel query time is similar, but when the query scope increases that the query time of MRHBase increases much slower than that of MySQL. Among them, the query performance of MapFile is the worst because MapFile accesses the file randomly by calling the get () method to access the data in the file. In the query can only one by one to read the dat and when the data increases the query time will be longer and longer. In the query can only one by one to read the data and when the data increases the query time will be longer and longer. It can be found, when the greater the amount of data that the MRHBase effect more obvious.

In order to verify the extensibility of MRHBase, we use different nodes to query the largest layer of image3 in resolution. Which query area contains the number of tiles is about 32768, the size of about 786.8MB.

It can be seen from Figure 6 that the query time is gradually reduced as the node increases. When the number of nodes is from 3 to 6, the query time decreases fastest£By looking at the partition of the HBase table, we can see that the number of region is 6 and evenly distributed over different nodes, so each node can query the data under the current region at the same time.When the node continues to increase, it is necessary to read data from the nodes that already exist in region, resulting in an overall decrease in time.

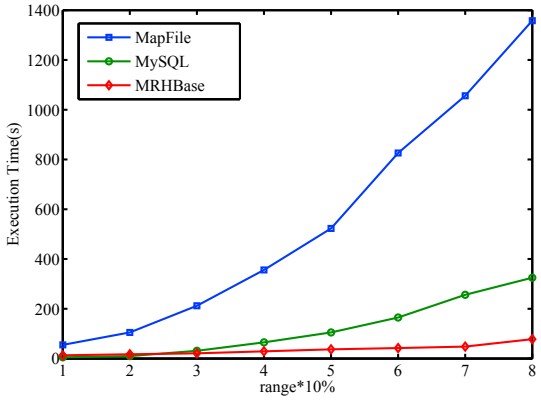

Fig. 5. Query time in different areas.

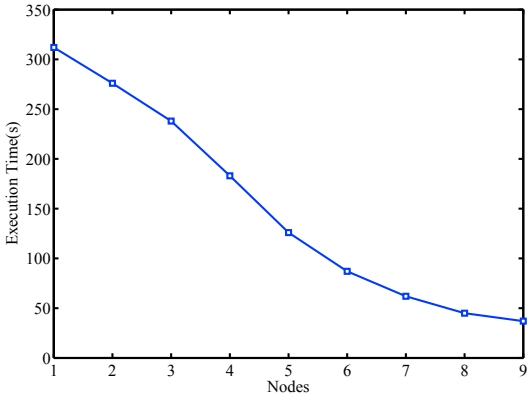

Fig. 6. Query time for different nodes amounts.

5. Conclusion

In this paper, the Hilbert curve is combined with the grid index and uses the characteristics of HBase distributed database to study the storage of image data. The spatial proximity of image data is guaranteed by redesigning the indexing and table storage patterns of HBase. Finally, the experimental results show that compared with MapFile and MySQL, it has better query efficiency and higher scalability.

References

[1] YangR,RamapriyanH,MeyerC.DataAccessandDataSystems[M]//AdvancedGeoinformationScience.2010.
[2] Gting R H. An introduction to spatial database systems[J]. Vldb Journal, 1994, 3(4):357-399.
[3] Kothuri R K V, Ravada S, Abugov D. Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2002:546-557.
[4] Zheng K, Fu Y. Research on Vector Spatial Data Storage Schema Based on Hadoop Platform[J]. International Journal of Database Theory & Application, 2013, 6(5):85-94.
[5] Zhong Y, Sun S, Liao H, et al. A novel method to manage very large raster data on distributed key-value storage system[C]// International Conference on Geoinformatics. IEEE, 2011:1-6.
[6] Chi Z, Zhang F, Du Z, et al. Cloud storage of massive remote sensing data based on distributed file system[C]// IEEE International Conference on Signal Processing, Communication and Computing. IEEE, 2013:1-4.
[7] Rajak R, Raveendran D, Bh M C, et al. High Resolution Satellite Image Processing Using Hadoop Framework[C]// IEEE International Conference on Cloud Computing in Emerging Markets. IEEE, 2015:16-21.

[8] Nishimura S, Das S, Agrawal D, et al. MD-HBase: A Scalable Multi-dimensional Data Infrastructure for Location Aware Services[C]// IEEE International Conference on Mobile Data Management. IEEE, 2011:7-16.
[9] Han D, Stroulia E. HGrid: A Data Model for Large Geospatial Data Sets in HBase[C]// IEEE Sixth International Conference on Cloud Computing. IEEE, 2014:910-917.
[10] Zhang N, Zheng G, Chen H, et al. HBaseSpatial: A Scalable Spatial Data Storage Based on HBase[C]// IEEE, International Conference on Trust, Security and Privacy in Computing and Communications. IEEE Computer Society, 2014:644-651.
[11] Ma T, Xu X, Tang M, et al. MHBase: A Distributed Real-Time Query Scheme for Meteorological Data Based on HBase[J]. Future Internet, 2016, 8(1):6
[12] Xia Y, Yang X. Remote Sensing Image Data Storage and Search Method Based on Pyramid Model in Cloud[C]// International Conference on Rough Sets and Knowledge Technology. Springer-Verlag, 2012:267-275.
[13] Wei X, Lu X, Sun H. Fast View of Mass Remote Sensing Images Based-on Image Pyramid[C]// International Conference on Intelligent Networks and Intelligent Systems. IEEE, 2008:461-464.
[14] Zhang G, Xie C, Shi L, et al. A tile-based scalable raster data management system based on HDFS[C]// International Conference on Geoinformatics. IEEE, 2012:1-4.