On Massive Spatial Data Cloud Storage and Quad-tree Index Based on the HBase

Xiaolan Xie, Zhuang Xiong, Guoqing Zhou, Guoyong Cai
Institute of information science and technology, Guilin University of Technology, Guangxi, 541004, China
Guangxi Key Laboratory of Spatial Information and Geometric, Guilin, 541004, China
School of Computer Science and Engineering ,Guilin University of Electronic Technology, Guilin 541004, China

Abstract
This paper achieves a linear quad-tree retrieval structure in the HBase non-relational database, and the MapReduce is applied to operate data insertion and index creation, so that the insertion and query spatial data can basically achieve parallelism. In order to test the spatial data query capabilities and different parameters on retrieval efficiency, we use MapReduce to retrieval quad-tree index structure, then compare with the single one's efficiency thus verifying the validity of cloud cluster retrieval calculated under the case of massive spatial data.

Keywords: spatial data; cloud computing; quad tree; storage; index

1. Introduction

The rapid development of high resolution sensor technology puts the geographic information system (GIS) into a serious situation, which are the rapid increase of data and whether they are put into effective use. The requirements of spatial database storage have been from the current GB level to TB level even to PB level. Applied cloud computing has already become an important way to solve mass and regional distribution of spatial data[1].

MapReduce calculation model has almost become the standard of mass data batch. however, the research about the combination of MapReduce and spatial data storage and retrieval is still sparse. This paper actualize a space data storage scheme and a linear quad-tree index structure under the HBase, and tests the influence of different parameters on the MapReduce retrieval algorithm efficiency.

2. Quad-tree

Quad-tree is a data structures that is widely used in the analysis and classification of spatial data. A rectangular spatial data is divided into four

parts(the quad-tree nodes). Each part can be divided into four small section, while a quad-tree whose depth is n has 2n *2n leaf node.

Morton code is a common method to quad-tree, which uses a one-dimensional data to represent the two-dimensional spatial information. Form the algorithm, the Morton code is staggered by the row of two-dimensional and column of the binary, and from the arrangement, the Morton code is more like a Z increasing order of sequence, a 4 hexadecimal Morton code diagram shown in figure 1 [2].

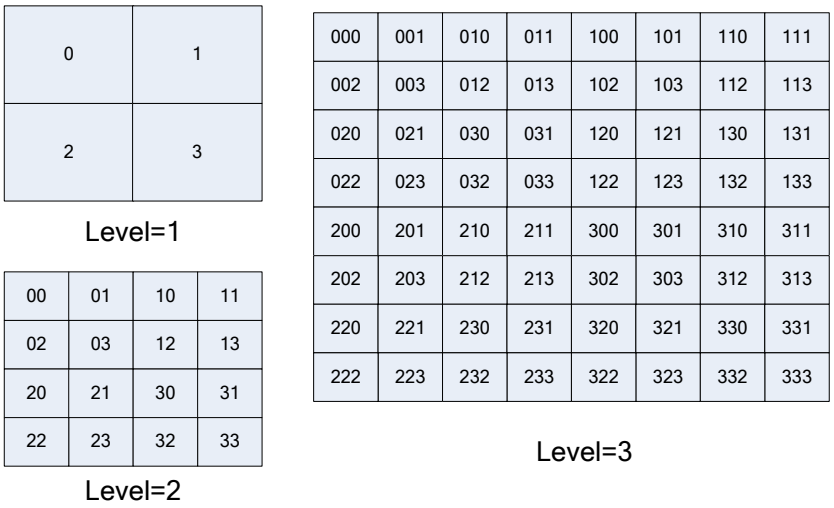

Fig.1: A 4 hexadecimal Morton code diagram

3. Quad-tree on HBase

3.1 The Column Properties of HBase

HBase is a suitable database for unstructured data storage. The data in HBase is stored in the table. A table is composed by rows and columns, and a columns is divided into a number of column families (row family) permission, disk and memory access, whose statistics are carried out at the column family level, and that is the properties of Hbase "stored in the column" [3].

If the spatial data possess text, images, voice and movie four descriptions attributes, then 4 description field must be added in the traditional relational database. However, a Column Family in Hbase could contain these four descriptions attributes. HBase will not waste storage space for non-exist attributes, and will easily seek data according to attributes (such as: query image description elements).

3.2 Storage Spatial Data into HBase

The key in HBase table is used of spatial data's autoid, while Column is used to store the different attributes description, in table 1 (Timestamp omited, the same below)

Table 1: HBase spatial data table

Key	Column Family				
355	Point=1F4E30E6 3DA62	Word= hankou	Photo=as .png		
356	Point=1F4E30E6 4E302	Word= xiantao		Sound=bo. mp3	
357	Point=1F34E306 3DA62		Photo=lt 2.jpg	Sound=ks. wmv	Movie=e3.a vi

The key in HBase index table is to save the area's Morton code, and Column to store all the space elements and coordinate information (point spatial data to save the coordinates, line or polygon spatial data save MBR information), parts of the point of spatial index table in table 2:

Table 2: HBase spatial index table

Key	Column Family		
0ED3	3692=146.9,40.6	3693=195.14,279.6	3701=135.8,100.8
0ED4	3697=221.8,288.3	3710=46.51,246.2	
0ED5	3734=184.1,96.4	3740=48.64,205.8	

Data in HBase arranged according to the Key, and most of operations in space retrieval are "regional". If the spatial data index in accordance with the regional Morton codes are arranged linearly, the adjacent region index stored in HBase database will be closer. Therefore, it is likely to retrieval the spatial data contained in a region and its adjacent region.

If the spatial data is not point but lines or geometric figure, we can create another index table to describe the spatial data across regional space for convenient query. The key is to save the autoid of spatial data, and Column to save the regional Morton code and MBR information, as shown in table 3:

Table 3: HBase spatial index-plus table

Key	Column Family	
3692	0ED3=146.7,40.6,195.1,279.5	0EE3=135.8,100.9,240.4,119.0
3697	3ED4=46.5, 141.2,76.5,246. 8	0F83=118.1,167.4, 287.3,229.8
3734	04D5=48.6,205.2, 265.9,267.4	1DA3=202.8,83.2, 206.4,20.3

4. The Actualize of Mass Point Insertion and Query Algorithm with MapReduce

4.1 Insertion Algorithm

Use MapReduce to complete the insert operation steps of massive spatial data [6, 7]:

Step1: Map

1). Read the inputs of the X, Y coordinates, and calculate the Morton code of that space.

2). Convert the autoid and other attribute of the point to Key-value, and output it (Key is to save the autoid of point, value to save other attribute).

3). Convert the Morton code, x/ y coordinate and autoid of point to Key-value, and output it (Key = "Morton" , and value is to save the coordinate, Morton and autoid).

Step2: Combiner/Reducer

If the value of key is Morton code, convert the value to the coordinate, Morton code and autoid. Then set the Morton code to the Key, set the autoid and coordinate of point to the name and content of Column, and upload the data into HBase's index table.

If the value of the key isn't Morton code, set the autoid to the Key, the other attribute to Column's name and value, and upload the data into HBase's data table.

Line and polygon are more complex than point, which is less used in plotting. The correspondent attributes are inserted in the data tables, while the autoid and the object of MBR are inserted into the index table. If the line and polygon span several regions, the autoid and MBR data of geometry should be inserted in every index table of that area.

Because of the uncertainty of Combiner run, the repetition of the same data to the data base is inevitably a waste of performance (For the covering properties of Hbase, the final result will not be affected by the repetition). Therefore, the single Reduce will be adopted in this paper. If there are enough computing power and higher requirements for efficiency, Combiner or multiple Reducer are considered as options..

4.2 Query Strategy

A typical selection to find points in rectangular area, the steps as follows:

Step1:Map

1). Read a row(a Morton code of an area) in Hbase, and calculate whether this area belongs to the scope of inquiry.

2). If this area isn't included in the scope of inquiry, end the task in this row. Run the node of the Map task and apply for the next Map task.

3). If the area is included the scope of inquiry, all points in the shape is suitable for the selection and output autoid of all of the point in this shape.

4). If the area intersects to the scope of inquiry, continue the step 5 and decide whether the points in the shape are suitable.

5). Read the X, y coordinates of all the points. If the point is suitable the select, output the autoid. If not, next point will be continued to examine.

Step2: Combiner/Reducer

Theoretically speaking, the same autoid in spatial point data is not exist. The value in Key-value which the Combiner/Reducer received should not be a key corresponding to the multiple values. Combiner/Reducer here plays a role of summary, and will output the Key-value as original.

Whereas, The output of lines and polygon in Map may have duplicate values (the output of the point retrieval in Map is not repeated, so the final results could be directly made after being merged.). Therefore, the Combiner and Reducer of the lines and geometric shapes must be equipped with the function of removal

repeat, which means that only one result should be output from the multiple repeats.

After Reduce operation, we get the autoid of all the points that are suitable for the select. At the meantime, we can retrieval a detailed description in data index according the autoid.

5. MapReduce and Hbase Performance Tuning

Common method to connect Hbase by MapReduce is extending the parent class of Mapper which is provided by org.apache.Hadoop.HBase. By extending the class and overwriting the Map, we can achieve the MapReduce to connect Hbase. However, the Htable will be splatted according to region ‘s start/end key, which remains to be processed by the MapReduce provided by Hbase. Each Mapper processes one region data. According to default configuration, The amount of region is less than that of cluster nodes, but each region is very huge. Hence, the tasks of Map will be much fewer after being splatted. While the workload for single Map is much heavier, which makes the workload for MapReduce difficult to balance.

To solve this issue, we can set the size of region to a small value. The default value for Hbase.hregion.Max.filesize is 256 m [8], we may set it to 1M or even smaller. This method works more effectively, but it will have a big effect on the response time of MapReduce group when frequently invoking spilt and compaction. What’s more, it may bring a lot of management and other aspects of troubles.

6. Experiments

6.1 Platform

Hardware: Master: 3.40GHz i5-3570 CPU 4G Memory;
Four Slave: 3.30GHz i3-3220 CPU 4G Memory;
System: Fedora18 x86_64 Hadoop-1.0.3 HBase-0.94.2 jdk-7u40;

Test Data: using 1 million random spatial points generated by a Java program, among which 90 percent has the attribute of text, 50 percent has the attribute image, 30 percent with sound attribute, and 10 percent with movie attribute.

6.2 Performance between the MapReduce and a standalone computer

Record the time of select operation in table 4(data time unit: thousand, time unit: ms, OM stands for memory error in Java):

Table 4: the time of select operation

Data size	10	50	100	200	400	500	800	1000
Hadoop	14260	14453	13109	14347	17424	20279	27647	34559
signal	247	1789	3569	7578	OM	OM	OM	OM

The data amount is abscissa, and the retrieval time is vertical axis. Figure 2 shows the retrieval time changing with the data amount.

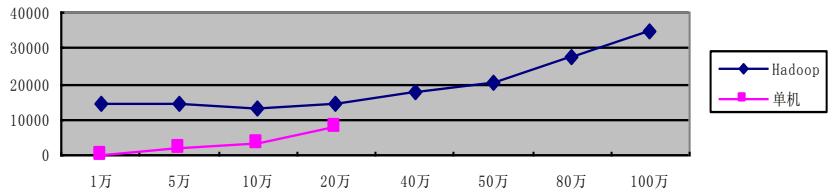

Fig.2

It is shown that it is not suitable for the cluster to possess the less data. When the amount of data is not too much, the retrieval cost is much slower than standalone computer since the calculation model mainly depends on the IO. The cost time passing the data which is calculated by Map to Reduce process is much higher than the Map and Reduce processing time. Therefore, if the size of the data does not reach a certain degree, it will affect the performance of retrieval to apply for the cluster. When the data volume is less than 200 thousand, the increasing of data has no influence on retrieval time because most time of cluster is costed by IO.

When the data's size reaches a certain degree, the calculation required time is much higher than the transmission one, then the advantages of the cluster begins to show. The increasing rate of the standalone possessing time increases faster than the in cluster environment in the later stage. When the data amounts to 200 thousand, it can not proceed normally in standalone environment, while it can be easily handled by cluster. Therefore, it is better to apply cluster than standalone for the mass data .

6.3 Different quad-tree Level on retrieval efficiency

Create spatial index for different spatial data in Hadoop cluster with setting level=6, 7, 8, then run space retrieval operation, and record the retrieval time as table 5(data time unit: thousand, time unit: ms, OM stands for memory error in Java).

Table 5: different quad-tree level on retrieval

Data size	10	50	100	200	400	500	800	1000
Level=6	14260	14453	13109	14347	21332	23563	OM	OM
Level=7	14380	14235	13250	14544	14424	17279	33435	OM
Level=8	13116	14979	13421	14379	14676	17135	27647	34559

According to the table, when the data amount is less, the quad-tree level has almost no influence on the retrieval performance. Because the quad-tree of Hbase comes from the linear array of the quad-tree of Morton code. The retrieval for key of Hbase is a binary, so there is not too much influence on retrieval performance with the increasing of quad-tree level. It is sure that more index table will be brought with the increasing of quad-tree level. At the meantime, the data in index table decreases relatively. Therefore, all of all, the increasing quad-tree level makes no influence on retrieval performance for the less data.

For larger amount of data, the data is relatively huge in the index table. The retrieval time for index table is much less than the judgement time for intersect space, even the level is too less to process. Comprehensively speaking, We can set a high value for quad-tree level in the early stage of the creation for the index so that we can avoid recreating spatial index in later stage when the data amount increases.

6.4 Different Region Size on Retrieval Efficiency

Set the HBase Region size to 32 m, 64 m, and 4m. Create spatial index for different sizes of spatial data, then run space retrieval operation, and record retrieval time in table 6 (data time unit: thousand, time unit: ms):

Table 6:different Region Size on retrieval

Datasize	400	500	800	1000
4M	14325	17864	26985	34437
32M	14424	17279	27647	34559
64M	15038	16932	28050	34332

The influence of region size on performance seems not that big as expected. The most important reason is that the number of computer cluster is limited, and the biggest role of Region partition is to make the graphs to parallel processing, but if it splits too much, it will make HBase split operation too frequently as well as the query Region, thus leading to the performance issue of Hbase. Therefore, it will get better result when the amount of region is twice than the amount of calculation.nods.

7. Conclusions

In order to solve the problem of mass spatial data storage and retrieval, this paper implements a massive spatial data storage and implementation method of quad-tree index on Hadoop and HBase cloud platform. The experiment shows that the cloud can indeed deal better with the huge amounts of spatial data, which is difficult for standalone machine to process. At the same time, this paper tested the influence of quad-tree layer, number and size of the Region on the efficiency of retrieval performance, and presents a parameter selection method to improve the query performance of the cloud data.

Acknowledgements

This research work was supported by National High Technology Research and Development Program 863 under Grant No. 2013AA12A402, Natural Science Foundation of Guangxi Provincial under Grant No. 2013GXNSFAA019349 and GuangXi Key Laboratory of Spatial Information and Geomatics under Grant No.GuiKeNeng 1103108-25, 1207115-13 and 1207115-23.

References

[1] ZhuQing,ZhouYan. *Distributed Spatial Data Storage Object*. Geometric and Information Science of Wuhan University,31(5) ,pp.391-394, 2006.

[2] HeRongYuan, LiQuanJie,FuWenJie.*Guide to Developing and Application With Oracle Spatial Database*. Surveying and Mapping Press,pp.1-48, 2008.

[3] iammonster.HBase profile. http://jiajun.iteye.com/blog/899632, Jan.30,2011.

[4] wikipedia.shapefile. https://zh.wikipedia.org/wiki/Shapefile,Jul.22,2013.

[5] LuoXingYan. *Reretrieval and Implementation on HBase based Column-oriented Compression Algorithms* .South China University of Technology, pp.11-19, 2011.

[6] LiuTong. *Design and implementation of the data analysis system besed on Hadoop*. Beijing University of Post and Telecommunication,pp. 9-31,2012.

[7] TianXiuXxia,ZhouYaoJun,etc. *The Technology and Application of Distributed Computing and Storage Based on Hadoop Architecture*. Journal of Shanghai University of Electric Power,27(1) ,pp.70-74,2011.

[8] Apache.HBase Handbook. http://hbase.apache.org/book/config.files.html，Sep20,2013.