Research and Implementation of Geography Information Query System Based on Hbase

Guangxin Zhu\textsuperscript{a}, Mingjie Xu\textsuperscript{b}, Haiyang Chen\textsuperscript{c}, Jun Yu\textsuperscript{d}, Zhu Mei\textsuperscript{e}, Yang Ling\textsuperscript{f} and Lin Wang\textsuperscript{g}

State Grid Electric Power Research Institute (SGEPRI) Nanjing, China

\textsuperscript{a}zhuguangxin@sgepri.sgcc.com.cn, \textsuperscript{b}xumingjie@sgepri.sgcc.com.cn, \\
\textsuperscript{c}chenhaiyang@sgepri.sgcc.com.cn, \textsuperscript{d}yujun@sgepri.sgcc.com.cn, \\
\textsuperscript{e}meizhu2016@aliyun.com, \textsuperscript{f}842587420@qq.com, \textsuperscript{g}wanglin18@sgepri.sgcc.com.cn

Abstract. With the rise of cloud computing research, the advantages of cloud computing and storage resources are discovered continuously: distributed computing, massive, dynamic and so on. Consequently, more and more application systems begin to migrate to the cloud platform. However, the difference between cloud platform and traditional single or multiple sever model bring certain challenges to the system development. In this thesis, a Geography information query system based on HBase was studied and implemented relied on National Geography Public Welfare Project. The query system can make users to retrieve the information of sea wind and satellite images by the graphical interface. Through the effective application of query technology, the system can efficiently extract the concerned Geography data information for the fishery production and Geography disaster prevention. This thesis firstly displayed the Geography information query system with the running examples, and secondly tested and analyzed the HBase query optimization technology to verify its great expandability and usability. Finally, the performance test of Geography information query system indicated that the system had high scalability and high reliability in the cloud environment.

1. Introduction

In recent years, China's National Geography Bureau has been committed to the construction of "digital geography" [1], China is a large geography country with 3 million square kilometers of territory [2]. In order to give full play to the convenience of information resources sharing brought by digital geography, The National Geography Bureau (NGB) undertakes the National Geography Public Interest Project (NGIP), "Geo-environmental Information Cloud Computing [3] and Cloud Service System Framework". The Northeastern University is responsible for the technical research and construction of cloud computing platform. The main research contents of this sub-task are: the division and placement strategy of geographical data under cloud environment, the data loading and updating technology under cloud environment, the data query processing and optimization technology under cloud environment, and the monitoring technology of cloud computing platform. In addition, according to the status of system task execution, the project also studies the storage and computing resource strategy of dynamic distribution system, the energy saving and efficient data distribution balance strategy, the task description model...
based on the quality of service (QoS) and the scheduling algorithm based on the quality of service, and provide data operation interface, parallel data processing interface, and monitoring interface for upper layer applications. Geography information data processing is undoubtedly a large amount of data processing, using cloud computing platform for the processing of large data, can efficiently solve the problem of data processing [4-9]. This paper is based on the sub-task of "Technology Research and Construction of Cloud Computing Platform", with cloud computing as the platform, HDFS is used to store image data, data storage and query for research content, and the query efficiency is improved by improving the query algorithm.

2. The Design of The Overall Framework of Geography Information Query Processing System

2.1. Design Thinking

As part of the cloud-based platform application, the query system is guaranteed to meet the user's needs while minimizing the system resources occupied by its runtime. This requires a trade-off between computing speed and system overhead when designing the system. Therefore, pay attention to the following principles in the design:

1. Reduce some unnecessary functions, in order to reduce system overhead, and make the user interface of the query system as simple and concise as possible to ensure user satisfaction.

2. The query system should have high availability. As a cloud-based query system, it can quickly restore normal use when the cloud platform increases or decreases nodes without affecting the normal operation of the entire system.

3. The query system should be portable and adapt to different operating environments. The query system can run on cloud platforms built by different organizations and different enterprises.

4. The query system should have good scalability. When a new node joins the cloud platform, the function can be dynamically expanded without modifying the code of the system itself; or when the query item increases, the original can be re-originated. A new module can be implemented by adding a module to the system.

5. The query system should have real-time performance, and the system can return the data required by the user within a certain time according to the conditions proposed by the user.

6. The system should be as simple as possible to simplify the deployment and configuration steps, so that users can use it more easily.

2.2. System Framework

The system uses a more traditional C/S architecture. The data transmission between each module and the data transmission between the client and the server are shown in Figure 1. When the client starts, it can present an interface to the user. The MODIS [10] data operation section and the weather data operation section are included in this interface. Among them, the MODIS data operation section includes the query of the data according to the time and space restriction conditions, the display function for query results, user operations on query results, and uploading local data to the server. The operation part of meteorological data mainly includes the functions of querying data according to the name of the base station, geographical location, and uploading local data to the server and so on. The server is responsible for the storage of data, and the data of MODIS is divided into two parts for storage. The description information of the image is stored in HBase [11], and the MODIS image is stored in HDFS.
2.3. Design and Implementation of Data Storage Layer

At this level, the data storage and management is mainly completed. Here, Hadoop's HDFS and HBase manage the data and provide an interface for query access. Here we will introduce the MODIS data and the storage implementation of geometric meteorological data. The overall implementation process is shown in Figure 2.

3. Design and Application of Algorithm

3.1. Spatial Query Optimization

In this paper, in order to maximize the efficiency of spatial keyword queries (here mainly for the spatial information of NearGoos meteorological data), we divide the space into several regions, and use a value to represent all the data in a region. The basic idea is shown in Figure 4, which is named B-order naming convention.

In the application of this paper, the spatial information is two-dimensional and bounded. Therefore, in this paper, the whole two-dimensional space is divided into N data blocks, and the data in each data block is the data adjacent to each other. Data is data within the same range. The spatial information of the entire data block is represented by the same one-dimensional string. B-order assigns the same B-
order value to objects with similar spatial coordinate values, in order to bring together spatially adjacent
or identical data objects.

For the two-dimensional space coordinates of the NearGoos meteorological data object, this paper
uses this situation to reduce the dimensionality of the object, and the B-order value of the encoding is
used as the spatial information part of the rowkey. In practical applications, the value of latitude and
longitude is limited, so you can use a hundred and ten digits of longitude and ten digits of latitude to
form a string representing all the data falling within this 10*10 square range. When the comparison is
large, the granularity of the data division can be adjusted according to the actual situation to achieve
the best effect. For example, if the spatial information of a piece of data is 110.333 degrees longitude and
28.455 degrees of latitude, then according to the rules used in this article, The data falls within the data
block with a B-order value of 112 (if the longitude latitude lacks a hundred or ten digits, and 0 is used
to take place, such as 001).

The key generation function is "rowkey = B-order value + keyword". In this way, HBase can store
data objects with similar spatial space as much as possible; when querying, it can effectively use spatial
information to filter data, reduce the scope of query, and improve the efficiency of query. As shown in
Figure 3, the gray box sunglasses are the range that needs to be queried. In the actual query, only the
nine areas 35, 36, 37, 43, 44, 45, 51, 52, 53 are scanned and judged. That is, the area where the area 44
is completely in compliance with the requirements is not judged by the space condition, and the
condition determination is required for the other eight areas.

Algorithm 1: Query Algorithm B-order

Enter: TableName: the data table where the data is located; Space: space qualification
Output: none
BEGIN
1. IF ( The data part to which the block belongs meets the requirements )
2. SomeRowkey.add (B-order value) (SomeRowkey is a list here);
3. IF ( the data part of the block belongs meets the requirements )
4. AllRowkey.add (B-order value) (All Rowkey is a list here);
5. For ( removing the B-order values in SomeRowkey one by one )
6. { call HBase's scan method to set the query range using the B-order value;
7. Filter the results of the query using the space qualifications entered by the user; }
8. For ( take out the B-order value in All Rowkey one by one)
9. { call HBase's scan method to set the query range using the B-order value;
10. Store the result in ResultR; }
11. return(ResultR)
END

Figure 3. Space division
The spatial keyword query in this paper is: For a spatial keyword query \( Q(\text{TableName}, \text{Space}) \), \text{TableName} is the data table where the data object to be queried is located, \text{Space} is the space qualification condition input by the user, and the returned result is this space. All data objects contained within the qualification.

For such a query, the proposed spatial keyword search algorithm based on B-order is divided into 3 steps:

Step 1: The system determines the range of keywords to be queried according to the conditions entered by the user and the B-order value generation rule;

Step 2: Filter all query keywords into two types, one of which is that the part of the data is desirable, and the second is that the data contained in the data block is qualified.

Step 3: Use the obtained keyword set and determine the data range scanned by HBase's Scan method to get the final result.

In this section we will discuss how to improve the time and space query efficiency of the system. In order to effectively improve the efficiency of query in HBase, we must consider how to reduce the amount of scanned data. To reduce the scope of the scan data table, then this article should make full use of rowkey, or create an additional index table. In this paper, when the NearGoos meteorological data time and space search are implemented, the rowkey code can be used to achieve the rowkey range according to the input conditions, so as to avoid scanning a lot of useless data. The rowkey encoding rule used in the implementation. Rowkey = B-order value + time information + random number, so that multiple consecutive rowkeys can be obtained according to the time and space conditions input by the user, thereby achieving relatively accurate scanning of data.

4. Experiment and Performance Analysis

4.1. Testing and Performance Analysis Based on B-order Value Space Query Technology

In this section, in order to test the impact of query efficiency after using B-order value space query technology, the B-order value space query technique is used to test the experiment. Test experiments were also performed without using the B-order spatial query technique.

The relationship between the time spent recording the query and the total amount of data is shown in Figure 5. Through the observation of the image data, it can be clearly found that when the rowkey is optimized by the B-order value space filling technique, the query efficiency is significantly improved. Through the observation of the pre-optimization curve, it is found that the time spent on the query is basically linear with the total amount of data, with slight fluctuations. In Fig. 4, since the query time is small after adopting the B-order value space filling technique, the curve is closely attached to the coordinate axis, and the relationship between the query time and the quantity cannot be found through the curve trend. Figure 5 is taken after taking log10 of the query result. Through the observation of the optimized curve before optimization, it is found that the query efficiency has two orders of magnitude improvement.

The reason for this result is that the entire data table needs to be scanned and judged before the B-order space-filling technique is used, and when the B-order space-filling technique is used for the query, we just need to scan the data in the data area labeled 127. Therefore, there is a significant reduction in query time.
The relationship between the recorded query results and the total amount of data is shown in Figure 6. Through observation of the image, it is found that there is a linear relationship between the time and the amount of data in the case of the query condition 1. In the case of query condition 2, the time taken by the query does not change significantly with the increase of the amount of data, and it is not difficult to find that it basically maintains a straight line.

The reason for the above experimental results is as follows: Here, in the case of the query condition 1, the data contained in the 127 data block does not all satisfy the condition, and the data in the data block needs to be judged one by one. In the case of the query condition 2, the data in the label 127 is the data satisfying the query condition. Here, as long as all the row keys in the data block are obtained, the data in the data block is not filtered one by one, so the query is performed. Time did not increase significantly as the amount of data increased.
When the system is in use, it is sometimes not only to deal with the user's spatial constraints, but also to meet the user's time and other qualifications. In this section, this article will further optimize the rowkey in the HBase table to add time information based on the spatial information. The overall structure of its rowkey is rowkey = B-order value + time information + random number.

The weather data of NearGoos is used to test the query time and data volume before and after adding time information to rowkey and record the result. The spatial query conditions are longitude 110 degrees to 120 degrees, latitude 30 degrees to 40 degrees, time limit conditions are 20030101 to 20031212 (the start time is January 1, 2003, and the termination time is December 12, 2003). The relationship between time spent and total data is shown in Figure 7.

Through the observation of the data results, it can be found that the query time is basically kept on a straight line after the time attribute is added to the rowkey, and does not change as the amount of data increases. When the time information is not added to the row key, the data block is known according to the condition of the space, but the data within the data block does not all satisfy the condition. At this time, the data in the data block needs to be filtered one by one according to the time limit condition. At this time, as the amount of data increases, the number of times the judgment needs to be made increases, and the time taken for the query also increases.

The reason is that in the experiment, the query starting condition 1 can accurately set the HBase scan method to query the start and end rowkey, and the data in this range is satisfactory, as long as the keyword (rowkey) is obtained. You don't have to scan the data. Therefore, the query time does not change significantly with the increase of the amount of data. In the case of the query condition 2, the range space is determined, but the data information is still judged. As the amount of data continues to increase, the number of judgments continues to increase, and the time spent on the query naturally increases.
5. Conclusion
Firstly, the HBase-based geography information query system is designed and implemented according to the requirements. Then, the B-order value space-filling algorithm is proposed to optimize the query processing. The B-order value method is used to divide the data in the two-dimensional space, and the B-order value is added to the rowkey, which improves the query efficiency. Finally, the geography information query system is deployed on the real cluster, and the running process and results of the system are presented in combination with the specific data. The B-order space-filling algorithm used in the geography information query system is tested and analyzed. The results show that the algorithm can effectively improve query efficiency and reduce the consumption time of spatial queries by two orders of magnitude over the original time.

Acknowledgments
This work was financially supported by the State Grid Corporation of Science and Technology (WBS number: 521104170019).

References
[1] Hou Wenfeng. Tentative Ideas on the Development of Digital Geography in China [J], geography science bulletin, 1999, 18(6): 1-10.
[2] Wang Shicheng. The 21st Century Geography Strategy (2) [J], ShandongFisheries, 1997, 14 (6): 1-4.
[3] Chen k, Zheng WM. Cloud computing: System instances and current research [J], Journal of Software, 2009: 1337-1348.
[4] Liu Peng. Cloud Computing (Second Edition) [M], Beijing: Publishing House of Electronics Industry, 2010, 1-25.
[5] LI Qiao, ZHENG Xiao. Research Survey of Cloud Computing [J], COMPUTER SCIENCE, 2011, 38 (4): 32-37.
[6] Zhang Xingwang, Li Chenhui, Qin Xiaozhu. Research and Initial Implementation of Large-scale Data Processing Based on Cloud Computing [J], NEW TECHNOLOGY OF LIBRARY AND INFORMATION SERVICE, 2011, 4 (17): 3.
[7] Niemenmäe M, Kallio A, Schumacher A, et al. Hadoop-BAM: directly manipulating next generation sequencing data in the cloud [J]. Bioinformatics, 2012, 28 (6): 876-877.
[8] Hadoop [EB/OL], http://developer.teradata.com/extensibility/articles/Hadoop-dfs-to-teradata, Teradata Developer Exchange, 2010. (Visit date December 2012)
[9] Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: wait-free coordination for internet-scale systems [A], Proceedings of the 2010 USENIX conference on USENIX annual technical conference [C], 2010, 8: 11-12.
[10] MODIS [EB/OL], http://zh.wikipedia.org/wiki/, 2013. (Visit date January 2013)
[11] HBase[EB/OL], http://baike.baidu.com/view/1993870.htm, 2013. (Visit date December 2012)