Regular Grid DEM Data Processing Based on Hadoop

Xiaosheng Liu*, Qiufeng Huang and Liang Zhong
Jiangxi University of Science and Technology, Ganzhou, Jiangxi, China

*Corresponding author e-mail: 371015234@qq.com

Abstract: Digital Elevation Model (DEM) has been widely used in various sectors. But its data structure is special, and the amount of data is large. When it comes to big data or large-scale data simulation, the data processing of stand-alone mode often fails to meet the needs of users. This paper focuses on the Hadoop cloud computing platform for large data parallel processing and its built-in data types is limited, study the organization method of the regular grid DEM in the cloud computing platform, and designs the data types that applicable to the regular grid DEM processing in MapReduce framework. Finally, realize the parallelization processing of DEM and use non-source flood analysis as an example to verify the feasibility of data types.

1. Introduction

The Digital Elevation Model (DEM) is a digital representation of the physical ground that express the elevation value of the ground with a set of ordered numerical matrices. The features of uniform distribution are relatively easy to calculate in computer[1], which makes DEM data has been paid more and more attention by the researchers. However, the digital elevation model, which is recorded by conventional grid methods, has large data and is limited by computer performance. The data is separated from each other in a single machine environment and the cost of data migration is too high[2]. Therefore, it is necessary to find a more available distributed computing platform to solve the problem of massive DEM data processing.

Hadoop[3] cloud computing platform is a distributed computing framework that can make applications run on a large number of cheap and hardware devices. It has high reliability and good scalability, and can provide effective support for data intensive applications[4]. But its built-in data type is limited, which cannot meet the needs of DEM data processing in cloud computing platform. Therefore, it is necessary for Hadoop to design a data type that is applicable to DEM processing and solve the basic problems of massive DEM data processing in cloud platform.

2. Overview of Hadoop Image Processing

Hadoop is the technical core of the distributed file system (HDFS) and data analysis processing framework (MapReduce) for data storage, which is used for efficient storage, management and analysis of massive data (greater than 1 TB)[5]. Among them, HDFS, as the storage system of Hadoop, runs in a hardware cluster and stores very large files in streaming data access mode. MapReduce is made up of Map function and Reduce function, which completes the task decomposition and the summary of the results. It is suitable for batch processing of data rather than real-time processing.

Currently, the image processing methods based on the Hadoop cloud computing platform mainly include the following three types: (1) Converting image files into binary data streams, directly processing the data with Hadoop's built-in data types, Such as ZHENG Xinjie[6] et al using...
MapReduce model to study the tracking of light in the distributed environment, and the processing of massive video files in cloud computing environment, such as ZHOU Pengfei[7] and LIU Zongquan[8] et al. However, in image processing, when a pixel needs to use other related pixels to calculate, the serialized image data cannot provide support. (2) Using HIPI (Hadoop Image Processing Interface), its operating object is HibiImageBundle (HIB), the image set of a single file on HDFS, which can effectively solve the problem of too many files in HDFS, but it only supports conventional image types such as JPG, PNG, and GIF. (3) The realization of user-defined image file interfaces, such as ZHANG Liang[9] et al will explore the implementation of parallel image processing in cloud computing environment, LIU Xiaoli[10] et al to realize remote sensing image query under distributed conditions. This paper uses the third way to design data types suitable for Hadoop that cloud computing platforms for DEM special data structures.

3. DEM Data Processing Implementation

This article uses the MapReduce[11-13] framework that adopt a divide-and-conquer approach. Before processing, the DEM data is uploaded to each node of the cluster. During processing, each node reads the data nearby and uses each DEM data block as a job slice, which is used as a job record to deal with(map), and then the processed data is appropriate to merge (combine) and sort (shuffle and sort), and then redistribute it (to the reduce node). The main idea is to divide the problems that need to be implemented into Map (Map) and Reduce(simplified) process, as shown in Figure 1:

![Figure 1. MapReduce processing flow chart.](image)

When carrying out non-source flooding analysis, the slices are independent of each other and their correlation does not need to be considered. Therefore, the Reduce operation is not need to perform in non-source flooding analysis. Considering the need of data flow and inundation analysis of MapReduce, this paper first designs the corresponding key-value pair type according to the content of DEM data, obtains the DEM data content, compares the DEM size, and realizes the serialization and deserialization of DEM data, and then designs the input and output format of DEM data based on the type of key-value pair, including the block, read, write and create of DEM data. Finally, the Map process is realized, which is the realization process of non-source flooding analysis algorithm. The main contents of DEM data processing are as follows:

3.1. Key-value Pair Type

MapReduce handles data in the form of <key, value>, and cannot deal directly with a file stream, only the serialized classes can act as a key or value in this framework. The serialization format that come with Hadoop is Writable, and only the class that implements the Writable interface can be used as a value type, while the class that implements the WritableComparable< T > interface can be used as a key type or as a value type. In this paper, the Text type is used as the input and output key type of the Map function that records the DEM file name. The type of BytesWritable is used as input values type of the Map function, and the DEM data stored in the HDFS system will be read into the memory in the
streaming data access mode. As the output value type of the Map function, the custom class DEM contains the coordination information, the projection information and metadata, as well as the height, width and pixel value of the general image, and its basic structure is shown in Figure 2.

3.2. DEM Data Input and Output

The DEM data is stored in the HDFS file system in the format of GeoTIFF. Because the HDFS system uses the streaming data access mode, the DEM data on HDFS cannot be directly accessed by the access mode of the local file system. In this paper, the DEM data stored in the HDFS file system is read into the memory by the BytesWritable method to form a <key, value> key-value pair, namely <Text, BytesWritable>. Its input format DEMFileInputFormat inherits from the FileInputFormat class, which is used to describe the data input specification of MapReduce jobs, including file directory checking and file splitting methods. And each DEM file is used as an input block in this paper. By using the custom record reader DEMRecordReader to convert the separated data (each DEM file) to the input key-value pairs of the Map process one by one, the nextKeyValue() determines whether the current key-value pair exist or not. When the key-value pair exist, a custom DEM class is called to complete the reading of the DEM data, and the basic structure is shown in Figure 3.
The custom output format DEMFileOutputFormat inherits from the FileOutputFormat class, with <Text, DEM> as a key-value pair. The DEM data processed by the file name is passed to the record writer DEMRecordWriter, and the DEM data is stored in the specified format by the record writer and uploaded to the HDFS file system.

4. Experiments and Analysis

This paper takes the non-source flooding analysis in flood inundation analysis as an example to demonstrate that the DEM data type designed is reasonable and feasible, including the input and output key-value pair types of MapReduce framework, and the DEM file input or output formats, which can satisfy the requirements for the DEM data processing in the cloud computing platform. Its operating environment is as follows:

In the local area network of 20Mb/s, and Hadoop cloud platforms are built with 3 virtual machines. The operating system is Centos7 64bit, JDK1.8 and cdh5.11 versions of Hadoop2.6. Each virtual machine is equipped with 2 Intel® Core™ i7-6700HQ CPU @2.60GHz, 4GB of memory, and a disk with a maximum capacity of 80GB. The main node is named hqf0. It is used to start the NameNode, JobHistoryServer, and ResourceManager services. The first child is hqf1, which is used to enable the SecondNameNode, DataNode and NodeManager services. Another child node is hqf2, which is used to start the DataNode and NodeManager services. The data source uses 29 DEM images covering the range of Jiangxi Province on the geospatial data cloud with a resolution of 30m and an image size of 3601x3601.

Before the experiment, the client is required to upload the required DEM image to the HDFS, and the flooding analysis is performed in ascending order. Firstly, one image will be added to the experiment, and then one image is added at a time until 29 images are processed. The experiment assumes that the given submerged water level is 120m, and the processing time for 29 flooding analyses is shown in Figure 4.

According to the results of data processing, it takes about 5s to prepare for the execution from Job to start MapReduce. It takes approximately 7 seconds for each node to process a DEM data block individually. When the number of started Map tasks is less or equal to the number of child nodes, different data blocks are processed in different child nodes. In addition to the time required for mobile computing, in the same operating environment, the processing time is approximately equal, that is the data amount has a negative correlation with the number of child nodes. When the startup Map task fails or the processing speed is slow and cannot be completed on time, the MapReduce framework will start another task on other idle nodes. At this time, there will be time fluctuations. As shown in Figure

![Figure 4. DEM data processing time](image-url)
4, the time of 10 blocks of DEM data processing is longer than the time of 11 block DEM data processing. In addition to the time fluctuations that occur in the task failure, when the number of child nodes is constant, the data volume and the time used are approximately linearly distributed. Using 1 and 10 block DEM data for experiments to illustrate the situation, and the detailed operation record is shown in Table 1.

| Number of DEM | Killed | Successful | Submit time | Start time | Map start time | Map finish time | Node finish time | Job time | Total time |
|---------------|--------|-------------|-------------|------------|----------------|----------------|------------------|----------|------------|
| 1             | 0      | 1           | 18:36:14    | 18:36:19   | 18:36:22       | 18:36:26       | hqf2             | 18:36:26   | 7          | 12         |
|               |        |             | 18:44:31    | 18:44:43   | hqf1           |                |                  |           |            |
|               |        |             | 18:44:32    | 18:44:44   | hqf1           |                |                  |           |            |
|               |        |             | 18:44:31    | 18:44:40   | hqf2           |                |                  |           |            |
|               |        |             | 18:44:33    | 18:44:53   | hqf1           |                |                  |           |            |
|               |        |             | 18:44:34    | 18:44:56   | hqf1           |                |                  |           |            |

| 10            | 1      | 10          | 18:44:24    | 18:44:29   | 18:44:32       | 18:44:45       | hqf2             | 18:44:57   | 28         | 33         |
|               |        |             | 18:44:32    | 18:44:48   | hqf2           |                |                  |           |            |
|               |        |             | 18:44:35    | 18:44:56   | hqf1           |                |                  |           |            |
|               |        |             | 18:44:35    | 18:44:57   | hqf1           |                |                  |           |            |

It is known from the above table that when the DEM data amount is 1, its effective task is 1, and the task processing time is 7 second, its total time is 12 second. When the DEM data amount is 10, the number of tasks started is 11, and its effective number of tasks is 10. Among them, six tasks are processed in the child node hqf1, and other’s tasks are processed in the child node hqf2. The Job processing time is 28 second and the total time is 33 second. From the total usage time, the processing time of second Job running in 2 child nodes is approximately 3 times than first Job, which greatly reduces the processing time. From the point of view of task allocation, when processing the 10 pieces of DEM data of second Job, it is not evenly allocated 5 tasks in each child node, but allocated to the idle child nodes, which greatly reducing the waiting time and improving the processing efficiency. The experiment also shows that increasing the number of child nodes appropriately can effectively reduce the running time and obtain the desired result more quickly when the amount of data is enough and the number of started tasks is far higher than the number of child nodes.

## 5. Conclusion

This paper studies the DEM data processing based on Hadoop cloud computing platform, and designs Hadoop data types that suitable for DEM data processing. In non-source flooding analysis experiments, the time including remove calculation time and the data processing time. When the DEM data amount is 1, the processing is performed in one child node, which is equivalent to the stand-alone mode. When the number of DEM (set to n) is less than or equal to the number of child nodes, it is equivalent to n child nodes processing DEM data at the same time, and the time consuming approximately equal to the time when DEM data amount is 1. When the amount of DEM data is much larger than the number of child nodes, the time cost is much less than the time spent processing DEM data in stand-alone mode. The experimental results show that the data type designed in this paper is reasonable and feasible, and that can achieve the purpose ss of DEM data parallel processing on the cloud computing platform.

## Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (Grant:41561091) and the Jiangxi Provincial Science and Technology Department Key Project (Grant:20142BBE50024).
References

[1] WANG Jiang, DU Daosheng. An Improved Algorithm for Simplification of Regular Grid DEM[J]. Journal of Geomatics.2017(2):34-36.
[2] KANG Junfeng. Technologies of Storage and Efficient Management on Cloud Computing for High Resolution Remote Sensing Image[D]. Hangzhou: Zhejiang University.2011.
[3] CHEN Ji-rong, LE Jia-jin. Reviewing the big data solution based on Hadoop ecosystem[J]. Computer Engineering & Science.2013(10):25-35.
[4] WAN Chuanmei, XIE Zhenglan. Detailed Hadoop application development practices[M]. Beijing: China Railway Publishing House.2013.6.
[5] AN Junxiu, WANG Peng, YAN Yujian. The basis and practice of Hadoop big data processing technology[M]. Beijing: People’s Posts and Telecommunications Press. 2016.11.
[6] ZHENG Xin-jie, ZHU Cheng-rong, XIONG Qi-bang. Design and Implementation of Distributed Ray Tracing Using MapReduce[J]. Computer Engineering. 2007, 33(22): 83-85.
[7] ZHOU Pengfei, GUO Qiaojin, HU Jie. Video Processing System Based on Hadoop Platform[J]. Informatization Research.2016,42(6):64-67.
[8] LIU Zhongquan, CUI Tong, LIU Chang. Hadoop-Based Fast Massive Video Processing Method[J]. Command Information System and Technology.2015(2):57-60.
[9] ZHANG Liang-jian, HUAN Fei, WANG Yang-de. Parallel Image Processing Implementation under Hadoop Cloud Platform[J]. Information Security and Communications Privacy. 2012(10):59-62.
[10] LIU Xiao-li, XU Pan-deng, ZHU Guo-bin. Parallel and Distributed Retrieval of Remote Sensing Image Using HBase and MapReduce[J]. Geography and Geo-Information Science.2014(5):26-28.
[11] LIU Yi, CHEN Luo, JING Ning, et al. Parallel Batch-Building Remote Sensing Images Tile Pyramid with MapReduce[J]. Geomatics and Information Science of Wuhan University.2013(3):278-282.s
[12] YU Jinjiang, ZHU Zhixiang, LI Congying. Comparative Study of Old and New Architecture of Hadoop MapReduce[J]. Computer & Digital Engineering.2017(1):83-87.
[13] HUANG Shan, WANG Botao, WANG Guoren, et al. A Survey on MapReduce Optimization Technologies.2013(10):885-905.