PROCEEDINGS OF SCIENCE

Implementation Based on Hadoop Ophthalmic Imaging Serialization File Store

Xuguang Zhao
Institute of Information Engineering , Capital Normal University Beijing ,100048,China
E-mail: zxg0406@126.com

Shudong Zhang, Zhongshan Ren
Institute of Information Engineering , Capital Normal University Beijing ,100048,China

With the rapid development of ophthalmic treatment sector, the realtime access to vast amounts of data of patient treatment and image information plays a particularly important role in the aspect of medical imaging storage and management. Nowadays, the cloud platform of distributed storage based on Hadoop is becoming more and more popular in the field of industry; but the Hadoop is for the storage of large files while the ophthalmic image and other medical images are generally of 500KB size, or even smaller. They are stored in the Hadoop, which results in its bottleneck. As a result, various small images merge to file into a sequence of operations as far as possible and stores the merged file into the DataNode, reducing the number of files stored in HDFS. And it is not immediate into the HDFS data nodes when the small file has to be transferred after combination. It is put into the cache server and used to wait for Serialization File, and finally uploaded to the HDFS so that a large amount of ophthalmic images can be effectively stored in the Hadoop platform.

ISCC 2015
18-19, December, 2015
Guangzhou, China

© *Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).*

PoS(ISCC2015)030

1. Introduction

Modern medical care is gradually entering the information age and the medical imaging technology has achieved rapid development in the last decade. A variety of imaging techniques make diagnostic tools of doctors to become more than ever before. Medical image diagnosis started to become especially important in modern medical activities and more and more modern medical diagnostic methods are related to the medical imaging information. Statistics show that a 500-bed medium-sized hospital produces variety of inspection images more than two million for a year, of which a CT generating 200000 images once a year. If each piece occupies 500Kbytes storage space, CT images will generate more than 100Gbytes storage space annually; if all medical images are digitized for the hospital, it will take up more than 2000 GB of storage space [1] .With the deepening digital hospital, the number is increasing rapidly. Cloud storage is a good solution for a massive data management. As an advanced service model, it is undeniable and has become one of the most clear directions for the current storage industry in the future. In various fields of massive data management system solutions, you can learn from the idea of cloud storage to build their own massive data management system. Cloud storage has many advantages in massive data storage, including excellent manageability, scalability, high reliability, low cost, and capacity close to unlimited expansion, a very big advantage of building a variety of massive data storage and management system. Hadoop is the most widely distributed storage and distributed computing platform, which has been widely used in many industries. Its performance and reliability have been widely used. Its distributed file system HDFS characteristics are suitable for streaming read and writer writing files almost never changes, which is consistent with the characteristics of medical image data; in this sense, it is very valuable to study the management of small file storage management[2].

2.HDFS File Storage Mechanism in HADOOP

HDFS is a typical master slave storage mode. It is a distributed file system designed to be able to run on common hardware. It is the foundation of data storage and data management in the whole cluster. For external clients, HDFS is just like a common hierarchical file system, which can be set up, deleted, renamed, move the folder or file and so on. The whole Hadoop cluster is composed by a group of nodes having only one control center NameNode node, which largely simplifies the system architecture. HDFS architecture contains a name node server NameNode, which is mainly used to provide HDFS internal metadata services, including multiple data nodes DataNode for providing block storage services and being responsible for processing data read and write requests. DataNode regularly reports the heartbeat packet and the operation command to the NameNode, NameNode controls DataNode by responding to the command of the return heartbeat. Fig. 1 is a schematic structure diagram of HDFS, showing three important roles in HDFS cluster: NameNode, DataNode and Client. NameNode and DataNode can be used in the ordinary cheap machines, but these machines are usually of GNU/Linux system. As the development language of Hadoop is Java, so as long as the machine is installed on the Java operating environment, you can deploy NameNode and DataNode.

PoS(ISCC2015)030

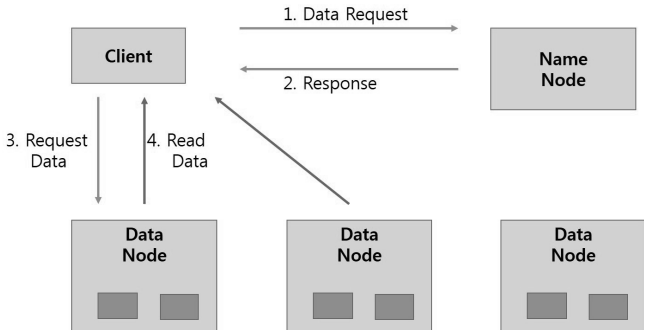

Figure 1: Schematic Diagram of HDFS

HDFS Hadoop distributed file system was originally designed for storage of large files, the default value of the data block is 64MB(this value can be modified in hdfs-site.xml {$HADOOP_HOME}/conf/, the property is dfs.block.size), small file generally means much less than 64MB of data block size of the file. The medical imaging data will be no more than 1MB in most cases, far less than the default size of the storage block 64MB. We study the processing and storage model of small files while putting forward a set of storage business process and making a lot of optimization[3].

2.1 Description of data storage In HADOOP

Although the HDFS is designed for large files, the files stored on the HDFS are similar to the traditional file system, which is also divided into blocks and then stored; however, the HDFS file block (data block) is stored like common files of the Linux file system, which is different from the traditional file system.When the Client operates the file content, the NameNode informs client of each data blocks residing at each DataNode; then the client directly communicates with the guardian process of the data node , that processes the local files corresponding to data node. At the same time, the data nodes communicate to other data nodes and replicate data quickly so as to ensure the redundancy of data, as shown in Fig. 2:

Data node ,as the slave node, will continue to report to the NameNode node. At the time of initialization, each node reports the current stored data blocks to the NameNode node. In the following data node, the data nodes are updated continuously to NameNode to provide the relevant information of the local modification, accept the instruction from the NameNode, create, move or delete the data blocks on the local disk[4].

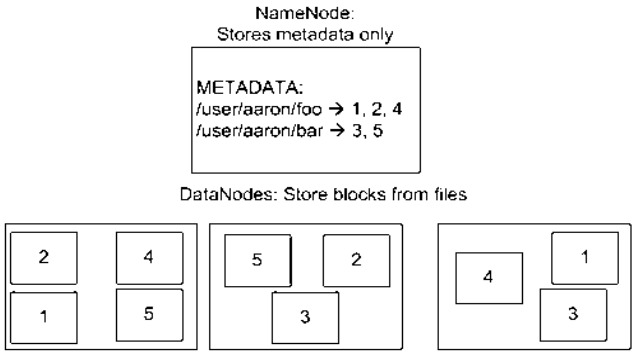

Figure 2: Stored Procedure Diagram

PoS(ISCC2015)030

2.2 Insufficient Storage of Small Files In HADOOP

NameNode is responsible for storing the metadata of the HDFS, a file corresponding to a metadata generally, a metadata of 150 bytes in size. The Namenode put metadata into memory rather than disk for the sake of fast read. HDFS was originally designed as a distributed file system for large file storage, so it has a high ability to store and handle large files and HDFS itself is designed to be a single point mode, NameNode's memory capacity is limited, which indicates that the number of metadata stored in the NameNode is also limited. When you need to write a file (regardless of size),the NameNode needs to generate 150 bytes of metadata information for each file, and then save it in memory. Each file needs to be divided into equal to the configured number of copies of data blocks and then writes to different DataNode. In this sense, if it is a large file, it is fully acceptable; but if it is a small file, most of the time is spent on metadata generation and distribution of copies, while the real file is not so long time to write, so it can be found that the efficiency is relatively low upon comparison.

3. Sequence File Composition

Small file increases the NameNode resource consumption and low efficiency of writing. It will be used to serialize large files for overcoming these problems. The SequenceFile is a data type that is designed by Key/Value to store the binary form of the type in Hadoop, which is composed of a sequence of characters. Hadoop provides append (key, value) method to carry out the adding operation of SequenceFile, each Key and Value is the corresponding data Record[5].

Block compression will compress multiple Records together and the density will be more compact. When it is compressed, it is the number of Key Record and Key values for compression, the corresponding Value value and Value value for compression, compared to Record compression, the compression ratio will be greater . Although in the archiving process, the time consumed by Block compression is more than that of Record, but for storage. Block compression can reduce a lot of spatial redundancy. The first Sync synchronization information is compressed in the Block compression, then press Record to record the number (default is 10000), all the Key length, all Key values, all Value length, all Value values are compressed.

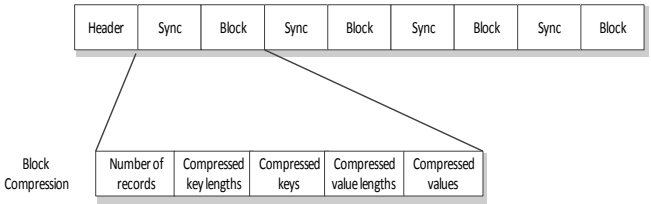

Figure 3: Internal Structure of the SequenceFile File for Block Compression

3.1 Sequence Ophthalmic Image

The research of this thesis is based on the National Fund project "the key technology research and application demonstration of the third party specialist imaging services". According to the previous data model of medical image, the design of this paper is as follows:

PatientID: different patients will have different IDs, it's the only identification of patients. According to previous data model, the structure of ID is "{name} _ {sex}} _ {date of birth}." Among them, the date of the name, gender, birth are UTF-8 encoding format.

PoS(ISCC2015)030

HsptID: hospital ID. Different hospitals have different IDs, which is the only one that distinguishes the hospital. This data is UTF-8 encoding format.

ChkTypeID: check type ID. Different imaging devices generate machine check that have different IDs, the only identification of the type of inspection. This data is UTF-8 encoding format.

ChkDate: check the time. Check in time is the UTF-8 encoding format. The data consists of year (4 bit) + Month (2 bit) + day (2 bit) + (2 bit) + points (2 digits) + (2 bit).

ChkID: Check ID is generated after the formation of the image every time. The composition of check is Chk_{ChkDate}_{HsptID}_{ChkTypeID}_{PatientID}. It is the only identification of a medical image data that is generated at a time.

The data model of the SequenceFile is designed as follows:

SeqDate: serialization time. Check the time is the UTF-8 encoding format. The data consists of year (4 bit) + Month (2 bit) + day (2 bit) + (2 bit) + points (2 digits) + (2 bit).

SequenceFile Name: SequenceFile naming format is "Seq_ {HsptID} _ {SeqDate} .seq".

Key SequenceFile named: key format is of ChkID format.

SequenceFile path in HDFS: the format of SequenceFile in HDFS is separated into layers with image format, hospital, time, the specific format of the physical path "hdfs: // {MasterIP}: 9000/{Image}/ {HsptID} / { Date} / {SeqName} ".

3.2 Serialization Process

The serialization process is:

·Establish SequenceFile of SequenceFile.Writer class, defining the target HDFS address[6].

·Obtain the addresses of all files to be uploaded, call the append method of the Writer for merging files,Key value is ChkID, Value value is FileStream of object files. The combined structure shown in Fig. 4:

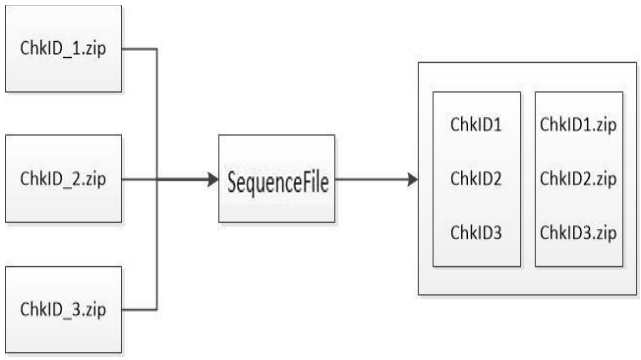

Figure 4: File Merge Structure Diagram

·It needs to record the corresponding file information after each file finished append operation in the process of the combination, and finally form the mapping index of the SequenceFile that "SequenceFile File →Serialization File →File Patient Information". It's shown in Fig. 5:

PoS(ISCC2015)030

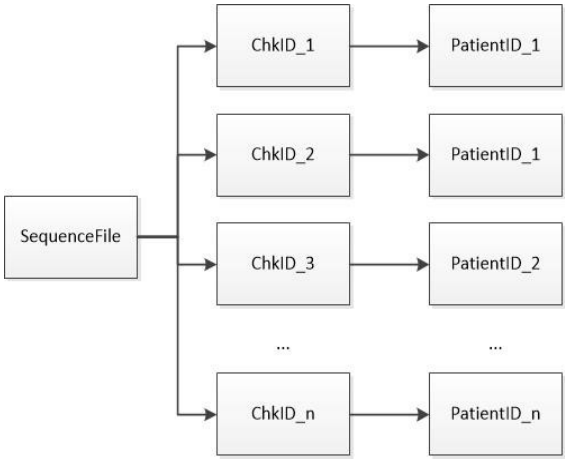

Figure 5: Map Index Diagram

4. Experimental Design and Test

In order to better handle small files like medical image problem in the medical cloud platform based on Hadoop, the small files will be optimized in the whole system. The idea of the overall architecture optimization has two points:

· It tries to merge all kinds of small files into a sequence of operations. The merged file is only stored in the data nodes of DataNode, which reduce the number of files stored in HDFS and improve the performance of HDFS storage. Then the merged serialization of documents is needed to make small file to file mapping index, and index records are stored to the cache to increase the efficiency of the query file.

· In the overall transmission structure, each record of medical image as generated does not reach the value of the set block size. When the small file needs to be transmitted after the merger, it's not stored immediately into the HDFS data nodes but first put into the cache server, waiting for the synthesis of SequenceFile file. Cache server can be set to multi-point mode, can not only enhance the cluster's availability and reliability, but also prevent data loss while the cluster is not available caused by downtime.

4.1 Optimization of Small File Processing based on Medical Cloud Storage Platform

As the speed of medical image data generation is very fast, the doctor will carry out picture review, academic research and clinical diagnosis, etc. for the medical data generated, thus it will be frequently read when the file has entered the storage device. Based on these characteristics, it is not enough to only deal with the small file, but has to make the appropriate optimization process in the medical cloud storage platform architecture[7].

In order to effectively reduce the access to HDFS and improve the speed of writing and reading the file, we set up the CacheNode file cache server between the client and HDFS. The function of the server is mainly for uploading the file and providing the file cache for the client . Pretreatment consists of extracting the uploading file, thumbnail generation and processing, recording the combination of serialization images and etc. In addition, we will establish index mapping for the merged serialization file in the cache server and set the cache. It will effectively reduce the number of reading and writing to HDFS, and allow the client to upload or download

PoS(ISCC2015)030

data quickly. After erecting CacheNode, the previous traditional PACS storage mode, namely "online - offline" mode, turned into a new "online - near-line - sharing" mode. The new storage model shown in Fig. 6:

Figure 6: Storage Mode Diagram

4.2 Experimental Environment and Data

We build a Hadoop cluster in this paper, which is composed of 5 hosts, one of which is NameNode node and the monitoring node server, and the other two are DataNode nodes, and the last two machines are CachNode nodes .

Experimental environment:

System	Proces Configuration	Other Information
NameNode	Intel core2、2.10GHz	Memory:8GB RAM Storage: 250GB HDD Operation System:Ubuntu 12.04, Hadoop 1.2.1 IP:192.168.1.90
DataNode1	Intel core2、2.10GHz	Memory:8GB RAM Storage: 250GB HDD Operation System:Ubuntu 12.04, Hadoop 1.2.1 IP:192.168.1.91
DataNode2	Intel core2、2.10GHz	Memory:8GB RAM Storage: 250GB HDD Operation System:Ubuntu 12.04, Hadoop 1.2.1 IP:192.168.1.92
CacheNode1	Intel core2、2.10GHz	Memory:8GB RAM Storage: 250GB HDD Operation System:Ubuntu 12.04, Hadoop 1.2.1 IP:192.168.1.93
CacheNode2	Intel core2、2.10GHz	Memory:8GB RAM Storage: 250GB HDD Operation System:Ubuntu 12.04, Hadoop 1.2.1 IP:192.168.1.94

Table 1: Hardware environment construction

Experimental data: the main study of this paper is the small file medical image, generally 500KB or so, and even smaller. We use the 1000 DICOM files as test data, the total size is about 470MB.

4.3 Construction of HDFS Cluster

The operation of all HDFS servers is as follows: extract the hadoop-1.2.1.tar file to the user folder, modify the related information in conf, add all IP addresses of DataNode nodes to the slaves file, and add the IP address of SecondaryNameNode to the master file. After all the machines are well configured, we run $HADOOP_HOME/bin/start-all.sh in NameNode for launching the NameNode and DataNode, then access to http://Master:50070/dfshealth.jsp. The web-page shown in Fig. 7:

PoS(ISCC2015)030

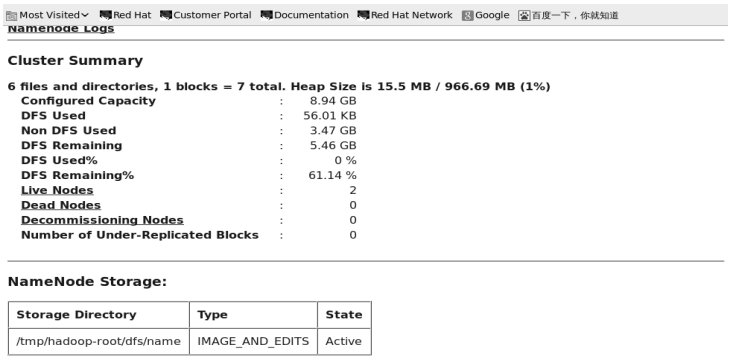

Figure 7: Master Login Interface Diagram

4.4 NameNode Memory Consumption Test

Firstly, we respectively put 1000,2000,3000 small files without serialization into HDFS, then record the memory utilization of NameNode process. Afte we user the same procedure to upload these files with serialization into HDFS, and then record memory utilization of NameNode process.

After experiment, NameNode memory consumption is shown in Fig. 8:

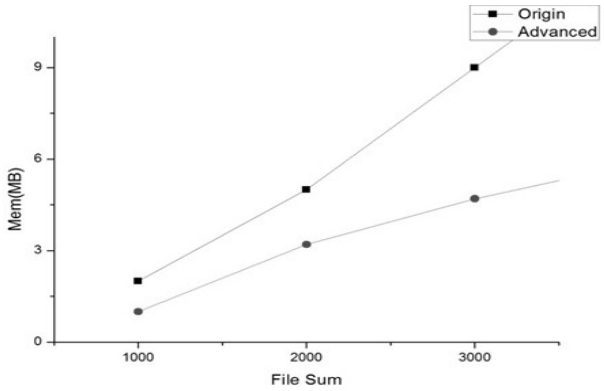

Figure 8: NameNode Memory Consumption Contrast

According to the experimental sample,NameNode memory consumption (unit MB) comparison after small files stored and processed :

The number of Small file	1000	2000	3000
Serialization memory consumption	1.44	3.02	4.56
Traditional memory consumption	1.79	4.68	8.86

Table 2: Memory consumption contrast

Fig. 8 shows that, with the increase of the number of small files, the memory consumption is nonlinear increased and the traditional storage consumption increases exponentially with file increasing. The rate of consumption is very fast, which will make the NameNode is easy to grind and the entire system will be unable to operate; with the number of small files increased for serialization store, the memory consumption increased slowly; thus the NameNode node memory usage has been significantly reduced when compared to the original HDFS system[8]. It is obvious that the structure redesigned and small file system optimization can effectively solve the problem of the memory of NameNode metadata.

PoS(ISCC2015)030

5. Conclusion

This paper mainly studies the technology of ophthalmic image data storage to solve the problem of ophthalmic image data storage. Based on the analysis of small file problems, it has been known that the existence of a large number of small files will make the writing efficiency low; however, the large amount of metadata consumed NameNode memory, causing the entire HDFS cluster performance burden, therefore, according to the research of small file system optimization, we use SequenceFile to sequence the file, which can be obtained as a large file or more close to the block size of the file so as to reduce the burden on the whole cluster caused by a large number of small files. In addition, we propose a new "online - near line - sharing" model, which improves the efficiency of file access, reduces the number of reading and writing to HDFS, and effectively solves the problem of low efficiency of reading and writing small files.

Finally, we set up the cloud storage platform of the ophthalmic image data and verify that the optimization scheme is feasibility by experiments. We detect the NameNode memory consumption by uploading a large number of files. The design scheme of this paper has greatly improved on the original performance by comparison of the serialization and non serialization.

References

[1] Pengjun Li . *Research and practice of the research and practice of the cloud service platform of medical image* [D].Guangzhou:South University of Science and Technology of China (2011) (In Chinese)

[2] YanNan Wang. *Based on massive Hadoop medical image data processing in the process of optimization method* [D].Beijing:Capital Normal University(2014) (In Chinese)

[3] Fei Guo,Binghong Zhan,Gang Liu. *Application Research of computers based on Hadoop clothing image storage and retrieval. Key technology* [J]. Technology of Computer Application.16(5):75-79(2014) (In Chinese)

[4] Qian Yan. *Massive medical data mining platform research and design* [D]. Wuhan :Wuhan University of Technology(2014) (In Chinese)

[5] Jing Zhang ,*Mass data management system based on Hadoop computer information*[J].Computer application and research.23(6):96-101(2010) (In Chinese)

[6] Weidong Zhang. *research and design Based on Hadoop images of cloud storage system* [D].Qingdao: Ocean University of China(2014) (In Chinese)

[7] Junzhou Luo,Jiahui Jin, Aibo Song, *system architecture and key technology of communication*[J].East. Cloud Computing. 17(9):56-62(2011) (In Chinese)

[8] Feng Zhang. *Information technology and information service mode of cloud computing applications*[J] Technology of Computer Application.11(5):25-29(2014) (In Chinese)

PoS(ISCC2015)030