A HADOOP-BASED DISTRIBUTED FRAMEWORK FOR EFFICIENT MANAGING AND PROCESSING BIG REMOTE SENSING IMAGES

C. Wang[a,b], F. Hu[b,*], X. Hu[a], S. Zhao[c], W. Wen[a], C. Yang[b]

[a] Hainan Geomatic Center, National Administration of Surveying, Mapping and Geoinformation of China, HaiKou, HaiNan, 570203, China – (cx8989, huxingshu)@163.com, 23327947@qq.com
[b] Department of Geography and GeoInformation Science and Center for Intelligent Spatial Computing, George Mason University, Fairfax, VA, 22030-4444, USA – (fhu, cyang3)@gmu.edu
[c] The 4th Institute of Photogrammetry and Remote Sensing, National Administration of Surveying, Mapping and Geoinformation of China, HaiKou, HaiNan, 570203, China - hnchj@sbsm.gov.cn

KEY WORDS: Remote Sensing, Image Processing, HDFS, MapReduce, GIS, Parallel Computing

ABSTRACT:

Various sensors from airborne and satellite platforms are producing large volumes of remote sensing images for mapping, environmental monitoring, disaster management, military intelligence, and others. However, it is challenging to efficiently storage, query and process such big data due to the data- and computing- intensive issues. In this paper, a Hadoop-based framework is proposed to manage and process the big remote sensing data in a distributed and parallel manner. Especially, remote sensing data can be directly fetched from other data platforms into the Hadoop Distributed File System (HDFS). The Orfeo toolbox, a ready-to-use tool for large image processing, is integrated into MapReduce to provide affluent image processing operations. With the integration of HDFS, Orfeo toolbox and MapReduce, these remote sensing images can be directly processed in parallel in a scalable computing environment. The experiment results show that the proposed framework can efficiently manage and process such big remote sensing data.

1. INTRODUCTION

Big Data, referring to the enormous volume, velocity, and variety of data (NIST Cloud/BigData Workshop, 2014), has become one of the biggest technology shifts in in the 21st century (Mayer-Schönberger and Cukier, 2013). Through remote sensing, various sensors from airborne and satellite platforms are producing huge volumes of remote sensing images for mapping, environmental monitoring, disaster management, military intelligence, and other applications. There are many mature software developed to process RS images in personal computers, such as Envi and Erdas. However, it is infeasible to process huge volumes of RS images in a personal computer due to the limitation of hardware resources and the tolerance of time consuming.

To handle the data- and computing- intensive issues in processing RS images, the techniques of parallel computing are applied. High performance computing is a new technology to do the parallel computing which make full use of the CPU's computing resource, but it is not suitable for the jobs with large I/O consuming. The RS image processing reads these data into memory first for further analysis, so the data I/O has become the bottleneck for HPC to process RS images.

Hadoop is an open-source software framework for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. It is composed of Hadoop Common, Hadoop Distributed File System, Hadoop YARN and Hadoop MapReduce. HDFS is an open source implementation of the Google file system (GFS). Although it appears as an ordinary file system, its storage is actually distributed among different data nodes in different clusters. MapReduce, a parallel data processing framework pioneered by Google, has been proven to be effective when it comes to handling big data challenges. As an open source implementation of MapReduce, Hadoop (White 2009) has gained increasing popularity in handling big data issues over the past several years.

There are many RS image processing libraries available on the Internet for further development by users. Orfeo ToolBox (OTB) is an open-source C++ library for remote sensing image processing, which provides affluent image processing functions, but it is targeted for a single PC, not for the parallel computing on a cluster.

To address the challenges posed by processing big RS data, this paper proposes a Hadoop-based distributed framework to efficiently manage and process big RS image data. This framework distributes RS images among the nodes in a cluster. By integrating the functions in OTB libraries into MapReduce, these RS images can be directly processed in parallel.

2. RELATED WORKS

A variety of research have been recently conducted on incorporating high-performance computing (HPC) techniques and practices into remote sensing missions (Lee, Gasster et al. 2011). There is much more information hidden in the remote sensing data than that can be seen, and extracting that information turns out to be a major computational challenge. For this purpose, HPC infrastructure such as clusters, distributed networks or specialized hardware devices, for example, field programmable gate arrays (FPGAs) and commodity graphic processing units (GPUs)(Mather and Koch 2011; Plaza, Du et al. 2011), provide important architectural developments to accelerate the computations related to information extraction in remote sensing (Lee, Gasster et al. 2011).

Golpayegani and Halem proposed a parallel computing framework that is well suited for a variety of service oriented science applications, in particular for satellite data processing (Golpayegani and Halem 2009). Lv, Z., et al. (Lv, Z., Y. Hu, et al. 2010) used map/reduce architecture to implement parallel

* Corresponding author

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
doi:10.5194/isprsannals-II-4-W2-63-2015

K-means clustering algorithm for remote sensing images (Lv, Hu et al. 2010). Li Bo., et al (Li, B., H. Zhao, et al. 2010) proposed a parallel ISODATA clustering algorithm on Map Reduce that is easy to use (Li, Zhao et al. 2010). Almeer tested 7 functions implemented in Java in Hadoop MapReduce environment (Almeer 2012), but the images have to be resized before processing to fit the Java heap size limitation. Kocakulak and Temizel implemented a MapReduce solution using Hadoop for ballistic image comparison (Kocakulak and Temizel 2011).

Other Researchers developed the parallel RS image processing algorithms with MPI. Generally, writing programs in MPI requires sophisticated skills for the users. With the increasing of image data, the parallel algorithms conducted by MapReduce exhibit superiority over a single machine implementation. Moreover, by using higher performance hardware the superiority of the MapReduce algorithm was better reflected. In the research conducted on image processing in the Hadoop environment, which is a relatively new field started for working on satellite images, the number of successful approaches has been few (Almeer 2012). Therefore, we decide to integrate OTB image processing tools with MapReduce to achieve efficient distributed storage and processing big RS image data.

3. SYSTEM DESIGN

This proposed framework is composed of two parts: data management and data processing as shown in Figure 1. In the data management part, the data can be directly fetched from other data platforms, and then stored in HDFS. In the data processing part, the input RS images will be assigned to reasonable number of Map tasks considering data locality and workload balancing. The OTB functions are embedded into MapReduce to directly process the data in each Map task. In the Reduce time period, the status of each Map task is collected to generate a log file. With the log file, we can monitor the status of MapReduce jobs.

3.1 Data Management

In order to help users to transfer such big data into HDFS, a data fetching module is developed. The data published in other data platforms can be directly downloaded into HDFS with customized configuration parameters, such as destination path, block size, and replication factor.

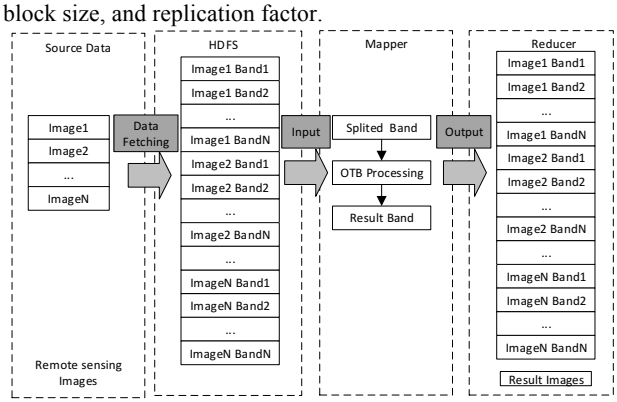

Figure 1 The architecture of the proposed framework

The traditional remote sensing data processing algorithms focus on the image file level, seldom on pixel level. However, in Hadoop computing architecture, structured image files, such as geotiff files, will be split into multiple blocks and stored in different data nodes by block size. It would lead to two problems:1) part of the original files cannot be recognized without the splitting metadata; 2) regrouping of the data requires excessive disk and network load which will affect the efficiency. Algorithms for reading and regrouping image binary data are also needed, which will add the complexity of the system development and finally affect the efficiency. To solve these problems, we set the block size parameter in Hadoop as big as the images when fetching data, which will keep each file from being divided.

3.2 Data Processing

3.2.1 Data Partition Period: To achieve parallel computation on input data, MapReduce partitions the whole dataset into many logical splits, and then assigns these splits to corresponding nodes to read and process the data in parallel. How these splits are partitioned and assigned directly impacts data locality, which makes a dramatic difference on the system performance. Considering the data locality and workload balancing, we customize FileInputFormat class. In FileInputFormat class, each band image will create a logic split, which will then be assigned to the computing node where the file is stored on. When fetching data, these image files have been evenly distributed among the cluster, so each computing node will be assigned a similar number of splits, which keeps the workload balanced among the cluster.

3.2.2 Map Period: After each computing node receives the assigned splits, it will launch a Map task for each split. In the Map task, we will first get the information delivered by the split, such as input file path, the image processing operation required by users and the file path for results. Then call the corresponding functions provided by OTB library to process the referred images. After the image processing, the result image will be directly stored in HDFS according to the file path referred by users. In addition, a status report for each image processing task will be recorded, and delivered to the next Reduce period.

3.2.3 Reduce Period: The Reduce task will collect all the status reports from the Map period, then analysis which image processing tasks succeed, and which fail. For the failed tasks, it will launch a new MapReduce job.

4. EVALUATION

4.1 Cluster Environment

A cluster with five high performance PCs has been setup for the experiments. The cluster is equipped with Hadoop 2.6.0 and consisted of a NameNode and four DataNodes. OTB package is installed on each DataNode. Both NameNode and DataNode use 8 CPU-cores (3.60GHz), 16 GB of RAM and 256 GB of SSD storage. The Name Node and all DataNodes are connected by 1 gigabit switch. Ubuntu 14.04, Hadoop 2.6 and Sun Java 8u45 are installed on both NameNode and DataNodes. Table 1 and Table 2 show the PC and cluster hardware configuration, while Tables 3 shows the cluster software configurations, respectively.

PC	Number	Details
Personal Computer	1	8 CPU-cores (3.60GHz), 16 GB RAM, 1 Gigabit Ethernet

Table 1. Single machine configuration

Name	Number	Details
Name Node	1	

Data Node	4	
Network	-	1 Gigabytes Switch
CPU	1	8 CPU-cores (3.60GHz)
RAM	16	GB
Storage	256	SSD disk

Table 2. Cluster hardware configuration

Name	Version	Details
Hadoop	2.6	Installed on each node
Ubuntu Server	14.04	
Java SDK	8U45	
OTB	5.0	Provides RS image processing tools

Table 3. Cluster Software Configuration

4.2 Data Source

The experiment data are TM satellite images generated by Landsat satellites, which are fetched from USGS website. They are consisted of 260 image files, each of which has 8071 × 7021 pixels resolution in TIF format. The spatial resolution is 30.0 meters per pixel, and we divide the dataset into 7 groups, and each group has 4, 8,16,32,64,128,256 image files (Table 4). Then we choose the BandMath tool in the OTB to apply a mathematical operation to the input image files to test the rum time.

Table 4: Data Groups for Test

Group	File Number	File Size (MB)
G1	4	220
G2	8	440
G3	16	880
G4	32	1760
G5	64	3520
G6	128	7040
G7	256	14080

4.3 Experiment Results

To compare the performance of the parallel mode and the sequence mode, two scenarios are designed. The first scenario is to run the referred image processing algorithm on a single node. The second scenario is to execute the referred algorithm on the cluster. In each scenario, 7 different data sizes are used for testing. To reduce the variability and measurement error, we conducted the operation ten times and took the average values. The average run-time for two scenarios are shown in Figure 2.

The figure shows that when the size of dataset is less than 1760 MB, the run-time for the PC is less than that for the cluster. That is because Hadoop has its own overhead to run a MapReduce job, such as launching Hadoop client, scheduling map tasks and so on. However, when the dataset's size is larger than 1760 MB, the run-time for the cluster is obviously less than that for the PC. Therefore, the proposed framework is better suited for large data size than for small data size when a computing intensive operation is required.

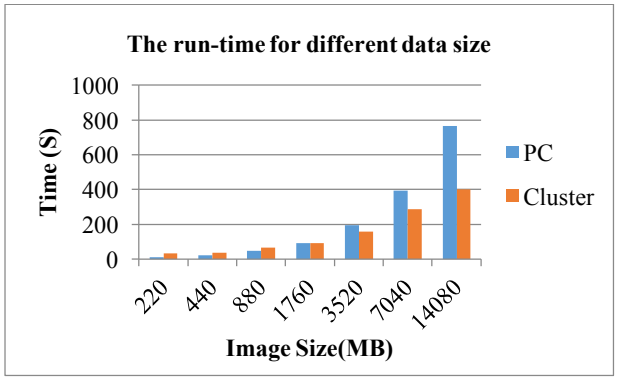

Figure 2. Time consumption for PC and cluster with different image size

5. CONCLUSION & DISCUSSION

In this paper, a Hadoop-based distributed framework is proposed to efficiently manage and process big RS image data. By integrating OTB RS image processing tools into MapReduce, this framework provides various parallel image processing operations. The experiment result shows that the proposed framework can reduce the run time when dealing with big data volume. In the near future, an algorithm for reading and regrouping image binary block data will be developed to support file split for addressing the special requirement for the block size, and achieve a better parallelism.

ACKNOWLEDGEMENTS

This work is supported by NASA HEC and NCCS, and NSF Spatiotemporal Innovation Center (IIP-1338925). The authors are grateful to their colleagues for their constructive comments and suggestions in writing this article.

REFERENCES

Almeer, M. H., 2012. Cloud hadoop map reduce for remote sensing image analysis. Journal of Emerging Trends in Computing and Information Sciences 3(4): 637-644.

Golpayegani, N. and M. Halem, 2009. Cloud computing for satellite data processing on high end compute clusters. Cloud Computing, 2009. CLOUD'09. IEEE International Conference on, IEEE.

Kocakulak, H. and T. T. Temizel, 2011. A Hadoop solution for ballistic image analysis and recognition. High Performance Computing and Simulation (HPCS), 2011 International Conference on, IEEE.

Lee, C., S. D. Gasster, et al., 2011. Recent developments in high performance computing for remote sensing: A review. Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of 4(3): 508-527.

Li, B., H. Zhao, et al., 2010. Parallel ISODATA clustering of remote sensing images based on MapReduce. Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010 International Conference on, IEEE.

Lv, Z., Y. Hu, et al., 2010. Parallel K-means clustering of remote sensing images based on mapreduce. Web Information Systems and Mining, Springer: 162-170.

Mather, P. and M. Koch, 2011. Computer processing of remotely-sensed images: an introduction, John Wiley & Sons.

Mayer-Schönberger, V., & Cukier, K., 2013. Big data: A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt

Plaza, A., Q. Du, et al., 2011. High performance computing for hyperspectral remote sensing. Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of 4(3): 528-544.

Apache Hadoop, 2013, Wikipedia, The Free Encyclopedia.

https://en.wikipedia.org/wiki/Apache_Hadoop (15 June. 2015)

White, T., 2009. *Hadoop: the definitive guide: the definitive guide.* O'Reilly Media, Inc., pp. 19-20