A Spark-Based Big Data Platform for Massive Remote Sensing Data Processing

Zhongyi Sun[1], Fengke Chen[1], Mingmin Chi[1,2(✉)], and Yangyong Zhu[1]

[1] School of Computer Science, Shanghai Key Laboratory of Data Science, Key Laboratory for Information Science of Electromagnetic Waves (MoE), Fudan University, Shanghai, China
mmchi@fudan.edu.cn

[2] State Key Laboratory of Satellite Ocean Environment Dynamics, Second Institute of Oceanography (SOA), Hangzhou, China

Abstract. With the fast development of remote sensing techniques, the volume of acquired data grows exponentially. This brings a big challenge to process massive remote sensing data. In the paper, an in-memory computing framework is proposed to address this problem. Here, Spark is an open-source distributed computing platform with Hadoop YARN as resource scheduler and HDFS as cloud storage system. On the Spark-based platform, data loaded into memory in the first iteration can be reused in the subsequent iterations. This mechanism makes Spark much suitable for running multi-iteration algorithms compared to MapReduce which has to load data in each iteration. The experiments are carried out on massive remote sensing data using multi-iteration singular value decomposition (SVD) algorithm. The results show that Spark-based SVD can obtain significantly faster computation timethan that by MapReduce, usually by one order of magnitude.

Keywords: Big data · Remote sensing · Spark · Hadoop

1 Introduction

With the fast development of remote sensing techniques, massive amounts of high spacial and spectral resolution images can be acquired for various applications, such as hazard monitoring and urban planning. This brings big opportunities for various applications based on the massive remote sensing data but also big challenges for big data storage and computation.

Usually, parallel and distributed computations are utilized to deal with the computational challenges of big remote sensing data. The parallel computing platform is often based on Compute Unified Device Architecture (CUDA) created by NVIDIA and implemented by the graphics processing units (GPUs) [4]. Thanks to efficiency and programmability of the CUDA GPUs, the technique has been successfully used in remote sensing applications, e.g., Cloud tracking and Reconstruction [9], remote sensing image fusion [17], and color balancing [15]. Similarly, the heterogeneous system OpenCL by combining multi-core

C. Zhang et al. (Eds.): ICDS 2015, LNCS 9208, pp. 120–126, 2015.
DOI: 10.1007/978-3-319-24474-7_17

GPU and CPU with operation system has been used to remote sensing data processing [1,3].

Although GPUs are capable of speeding up remote sensing data processing, it is still hard for a single computer, even a extremely expensive minicomputer or a PC server to analyze the big remote sensing data. Hadoop [2,6] based distributed storage and computing cluster provides a new resolution to the computation problem of big data. The work in [8,14] demonstrated that the performance in speed of MapReduce [6] based algorithm has an advantage over the single-thread process algorithm. However, in the MapReduce-based platform, the entire data set has to be loaded from hard disks to memory at each iteration, which takes a lot of time when processing massive data by multi-iteration data analysis algorithms.

To attack this problem, usually an open-source in-memory distributed computing framework, namely, Spark [16] is exploited for distributed computation. In the Spark-based platform, the data loaded into memory in the first iteration can be reused in the subsequent iterations. This mechanism makes Spark much suitable for running multi-iteration algorithms compared to MapReduce which has to load data in each iteration. In [16], multi-iterations programs using Spark based algorithm is up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. The Spark-based platform has been used to graph data applications [10], large scale security monitoring [13], log analysis [12] and so on.

As we know, Spark has not been used for processing remote sensing big data. In the paper, Spark-based platform is proposed to fulfil a multi-iteration algorithm for remote sensing application. In particular, the Hadoop Distributed File System (shorted as HDFS) is adopted and Hadoop YARN as resource scheduler. Here, the feature extraction algorithm, i.e., singular value decomposition(shorted as SVD) is implemented to evaluate the effectiveness of the proposed distributed platform. The experiments are carried out on two real-world massive remote sensing data. The results show that Spark-based SVD can obtain significantly faster computation time than that by MapReduce, usually by one order of magnitude.

The rest of the paper is organized as follows. The next section describes the preliminary knowledge of the proposed big data Architecture. The applied remote sensing datasets and machine learning algorithms are briefly described in Sect. 3. Section 3.2 reports and discusses the results provided by the SVD algorithm and different data sets. Finally, Sect. 4 draws the conclusions of this paper.

2 The Big Remote Sensing Data Architecture

In the paper, massive remote sensing data are dealt with in the big data processing architecture. In the following, the proposed architecture is first described. Then, the distributed computing models, i.e., MapReduce and Spark, and the related storage system are briefly introduced as follows.

2.1 Big Remote Sensing Data Architecture

The big remote sensing data are acquired from different sources, such as remote sensing data and data from the Internet and then input to the system. In the data processing stage, HDFS is chosen as the distributed file system. Then, big data in remote sensing are loaded to the HDFS at the beginning. In-memory based Spark, is chosen as the main principal distributed computing framework. Meanwhile, Hadoop MapReduce is also been integrated into the platform. Apache YARN is chosen as the scheduler responsible for allocating resources to various running applications between the HDFS and the distributed computing programming models, i.e., MapReduce and Spark. This means that the proposed platform supports the algorithm implemented by both Spark and MapReduce to process remote sensing data. Meanwhile, the set of data analysis algorithms, i.e., a machine learning library (MLlib) can be implemented in the Spark programming model. Similarly, MapReduce builds Apache Mahout, a scalable machine learning and data mining library. On top of the big data processing stage, different remote sensing processing tasks can be fulfilled, e.g., feature extraction and image classification.

2.2 Hadoop

Hadoop is a series of technology for distributed storage and processing of big data. It is an open source framework developed by the Apache Software Foundation. Hadoop is fault tolerant, scalable, and extremely simple to expand. Hadoop is capable of processing massive amounts of data sets which are unable to be dealt with or originally need expensive super-computers. Nowadays, Hadoop can manage thousands of computers, storage and process massive data in a PeraByte level. The core of Hadoop consists of two parts, a storage part Hadoop Distributed File System (shorted as HDFS) and a processing part MapReduce, which are introduced as follows.

HDFS. HDFS [2] is a Java-based file system designed for large data storage which was inspired from Google File System (shorted as GFS) [7]. The purpose of HDFS is to enhance the I/O performance of data storage and to ensure that the distributed storage system is scalable, fault tolerant. HDFS break data down into smaller pieces which called blocks and distribute them throughout the cluster. Each block of data is independently replicated at multiple servers. When a duplicate of a block is lost due to a hardware failure, HDFS can automatically provide the nearest duplicate instead and creates another duplicate of the block. HDFS supports redundant data storage which not only offers the tolerance to hardware failure, but also allows that Hadoop can divide a large task into small pieces and runs them on separate servers.

MapReduce. MapReduce [6] is the core of Hadoop. It is a programming model and an associated implementation for processing and generating large data sets.

Before the invention of the MapReduce, open multi-processing (shorted as OpenMP [5]) and many other models were popular in the field of distributed computing. Most of these models require a very long learning curve to master. However, MapReduce is a simplified data processing model. Users only need to modify the Map and Reduce functions according to the requirements of the task. A Map function processes a key/value pair to generate a set of intermediate key/value pair, and a Reduce function merges all intermediate values associated with the same intermediate key. This model can realize many real-world applications.

2.3 Spark

The Apache Spark [16] is a fast and general engine for large-scale data processing implemented in Scala [11]. Even though the MapReduce model has achieved an unprecedented success in implementing many real-world distributed tasks, it is not suitable for the applications built around a cyclic data flow model. The Spark is proposed to handle these applications while retaining the similar excellent properties of MapReduce, i.e., scalability and fault tolerance. Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing. In some specific applications, Spark may work 100 times faster than Hadoop [16].

The resilient distributed dataset (shorted as RDD) is used for the fundamental programming abstraction in Spark. The RDD is a logical collection of data partitioned across machines and can be rebuilt if a partition is lost. An RDD can be explicitly cached in memory across machines and reused for later MapReduce-like parallel operations. For the algorithms whose main body is a loop calculation, the intermediate RDD data sets do not need to read and to write from the hard disk at each iterative manner. This is one of the major reasons why Spark works faster. RDDs can be created in two ways, i.e., parallelizing an existing RDD and referencing a dataset in an external storage system. RDDs support many useful parallel operations in the latest release.

3 Experimental Results

To evaluate the effectiveness of the proposed big data platform for processing massive remote sensing data, a multi-iteration singular value decomposition (SVD) is implemented in both Spark and MapReduce platforms in terms of hyperspectral remote sensing data in different magnitudes of spatial resolutions.

In the followings, the datasets utilized are firstly introduced and then the corresponding experimental results are reported.

3.1 Datasets

The experiment is evaluated on two public remote sensing images briefly described as follows:

Paiva: It is a hyperspectral and high-resolution (1.3 m) image taken over the urban area of Pavia by the airborne ROSIS-03 optical sensor. The image consists of 1096*715 pixels with 103 bands ranging from 0.43 to 0.86 μm in the center of Pavia (denoted as (denoted as PaviaCenter), Pavia, Italy.

IndianPine: It is a hyperspectral image over the Indian_Pine test site on June 1992 by the AVIRIS instrument. The image size is 145*145 pixels with 220 bands from 0.37-2.5 μm.

Table 1. The details for the datasets used in the experiments.

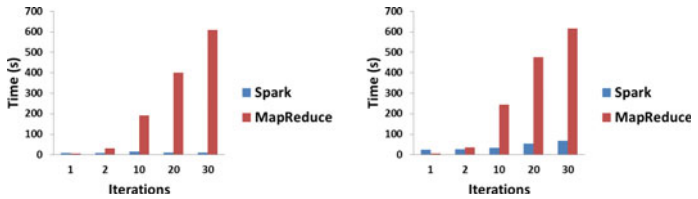
Dataset	Indian_Pines	Pavia
Number of pixels	34947	468666
Number of bands	200	102

The details of the datasets are summarized in Table 1.

3.2 Experiment Results

In order to evaluate the effectiveness of the proposed platform, the computation times are compared in terms of the SVD algorithm based on both the Spark and MapReduce platform. SVD is an effective algorithm for feature extraction (reduction) but time-consuming. The implementation of SVD in the distributed platform can greatly accelerate the algorithm. In the real environment of the experiments, the computation time cost is significantly influenced by the status of the experimental cluster. Accordingly, the results in a single trial could be unreliable. To ensure the robustness of the experiment results, all time costs are referred to the average ones over ten trials.

The time costs of implementing the SVD algorithms are shown in Fig. 1(a) and (b) using the Mahout (MapReduce) and MLlib (Spark) for the datasets Indian_Pine and Paiva, Respectively. From the experiments, one can see that the time cost of the SVD implemented by MapReduce is slightly faster than that by Spark on the same datasets when the singular value is taken as 1 due to



(a) Indian_Pines (b) Pavia

Fig. 1. Time costs for the SVD algorithm with Spark and MapReduce, respectively.

the higher hardware resource requirements of Spark. In the following iterations, the time costs of SVD implemented by MapReduce significantly increases due to the I/O operations. However, the time costs for Spark only slightly increase to run the decomposition computation in the distributed environment.

4 Conclusion

In the paper, a big data platform has been proposed to process massive remote sensing data processing, where Spark is utilized for distributed computation to avoid the expensive I/O operations compared to the MapReduce model as the principle distributed computing framework. In particular, the proposed framework is comporised of YRAN as distributed resource scheduler and HDFS as distributed file system. To evaluate the effectiveness for massive remote sensing processing, the multi-iteration Singular Value Decomposition (SVD) has been implemented in the proposed platform by both Spark and MapReduce models. The experiments show that the time cost on the Spark-based platform is far less than that on the MapReduce platform.

Acknowledgement. This work was supported in part by Natural Science Foundation of China under contract 71331005, in part by Shanghai Science and Technology Development Funds (13dz2260200, 13511504300), and in part by the Open Foundation of Second Institute of Oceanography (SOA).

References

1. Bilotta, G., Sánchez, R.Z., Ganci, G.: Optimizing satellite monitoring of volcanic areas through gpus and multi-core cpus image processing: An opencl case study. Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of **6**(6), 2445–2452 (2013)
2. Borthakur, D.: The hadoop distributed file system: architecture and design. Hadoop Project Website **11**, 21 (2007)
3. Callico, G., Lopez, S., Aguilar, B., Lopez, J., Sarmiento, R.: Parallel implementation of the modified vertex component analysis algorithm for hyperspectral unmixing using opencl (2014)
4. CUDA: http://www.nvidia.com/object/cuda_home_new.html/
5. Dagum, L., Menon, R.: Openmp: an industry standard api for shared-memory programming. IEEE Comput. Sci. Eng. **5**(1), 46–55 (1998)
6. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
7. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. In: ACM SIGOPS Operating Systems Review, vol. 37, pp. 29–43. ACM (2003)
8. Golpayegani, N., Halem, M.: Cloud computing for satellite data processing on high end compute clusters. In: IEEE International Conference on Cloud Computing, 2009. CLOUD 2009, pp. 88–92. IEEE (2009)
9. Grauer-Gray, S., Kambhamettu, C., Palaniappan, K.: Gpu implementation of belief propagation using cuda for cloud tracking and reconstruction. In: IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS 2008), vol. 4, p. 2 (2008)

10. Johnpaul, C., Thampi, N.S.: Distributed in-memory cluster computing approach in scala for solving graph data applications. In: 2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC), pp. 1–6. IEEE (2014)
11. Programming Language, S.: http://www.scala-lang.org
12. Lin, X., Wang, P., Wu, B.: Log analysis in cloud computing environment with hadoop and spark. In: 2013 5th IEEE International Conference on Broadband Network & Multimedia Technology (IC-BNMT), pp. 273–276. IEEE (2013)
13. Marchal, S., Jiang, X., State, R., Engel, T.: A big data architecture for large scale security monitoring. In: 2014 IEEE International Congress on Big Data (BigData Congress), pp. 56–63. IEEE (2014)
14. Pan, X., Zhang, S.: A remote sensing image cloud processing system based on hadoop. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), vol. 1, pp. 492–494. IEEE (2012)
15. Tan, Y.K.A., Tan, W.J., Kwoh, L.K.: Fast colour balance adjustment of ikonos imagery using cuda. In: IEEE International Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008, vol. 2, pp. II-1052. IEEE (2008)
16. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, pp. 10–10 (2010)
17. Zhao, J., Zhou, H.: Design and optimization of remote sensing image fusion parallel algorithms based on cpu-gpu heterogeneous platforms. In: 2011 4th International Congress on Image and Signal Processing (CISP), vol. 3, pp. 1623–1627. IEEE (2011)