Spark Platform Based Video Transcoding

Yunyu Liu and Jinpeng Yuan*
School of Computer and Information, Qiannan Normal University for Nationalities,
558000, Duyun, Guizhou Province, China
18285470037@139.com

Abstract. The HTML5 based videos play an important role in promoting the communication on national culture with the rapid development of the mobile internet. However, considering that the HTML5 based videos support Theora, H.264 and MPEG4 video coding formats only and there are various existing video formats on national culture, it is needed to conduct fast conversion on video files so as to adapt to HTML5 video labels. Therefore, a Spark platform based transcoding system is proposed in this article. The HDFS is adopted for storage, and the RDD (Resilient Distributed Dataset) and FFMPEG of Spark are utilized for distributed transcoding. It conducts detailed discussion on segmentation strategy for the distributed storage of videos, and makes comparisons on the thought of the MapReduce and that of the RDD. In addition, it proposes the RDD programming framework based distributed transcoding scheme. According to the comparisons on time consumed for transcoding between the MapReduce framework and the Spark framework with the same size of file block and cluster, compared with the MapReduce transcoding, the time used for transcoding of the Spark framework can be reduced by 25%.

Keywords: Spark; FFMPEG; MapReduce; video transcoding.

1 Introduction

Video transmission and interaction on mobile terminals have been widely applied due to the emerging video websites. However, due to different network environments and video coding formats, special video transcoding is necessary so as to adapt to the requirements on environment of mobile terminals. In conventional practice, a single server is adopted for video transcoding, but it costs the server very long time in condition of large video files.

2 Current Situations

Video transcoding means converting the video stream after compressed encoding to another video stream [1], so as to adapt to different network bandwidths, terminal processing capacities and demands of different users [1]. With the development in high-definition videos, the size of a video file is increased to dozens of G from several

TRIDENTCOM 2017, September 28-29, Dalian, People's Republic of China

DOI 10.4108/eai.28-9-2017.2273775

hundred of MB, and higher requirements on storage capacity of video files and transcoding servers are proposed so as to adapt to mobile terminal devices.

At present, there are the following four transcoding modes [2-5]:

(1) The stand-alone mode: It conducts transcoding to videos with a single transcoding server. It transmits videos to the transcoding server for transcoding, to return to the video after transcoding. This mode has the advantage of easy realization; however, the time used for transcoding is limited by the performance of the transcoding server, and it is not applicable to transcoding tasks with high concurrency.

(2) The distributed mode: Multiple transcoding machines are adopted at the same time for transcoding of a video file. The video file is segmented on the video source, and each segment is transmitted to corresponding transcoding machine, which are combined into a video file after transcoding, to return the video file [6]. This mode has the advantage of parallel transcoding with low time cost, and it is applicable to transcoding tasks of high concurrency; but it has complicated transcoding steps, with problems such as segmentation and combination.

(3) The cloud based mode: The storage and calculation capabilities of cloud are utilized for transcoding of video files. For example, the S3 (Simple Storage Service) server of Amazon is adopted by Grio [6] for storage of video files and the EC2 (Elastic Compute Cloud) is utilized for transcoding. In which, an instance of EC2 is in charge of the transcoding task of a video file.

(4) The distributed system based Hadoop platform [7] is adopted for transcoding; the HDFS of the Hadoop platform is utilized for storage and the thought of the MapReduce and the FFMPEG are utilized for video transcoding [8].

The HDFS (Hadoop Distributed File System) storage mechanism of the Hadoop framework is adopted for storage of video files, and the Spark based RDD (Resilient Distributed Dataset) and FFMPEG are utilized for distributed transcoding in this article, so as to achieve faster transcoding for massive video files.

3 Spark and FFMPEG

FFMPEG is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations [9].

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming. Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program). Specifically, to run on a cluster, the

SparkContext can connect to several types of cluster managers (either Spark's own standalone cluster manager, Mesos or YARN), which allocate resources across applications. Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application. Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, Spark Context sends tasks to the executors to run [10].

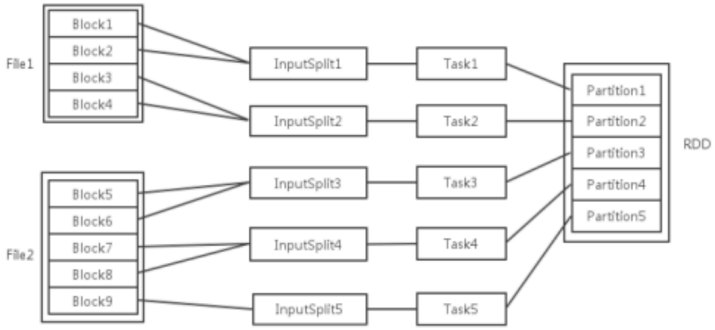

Fig. 1. Spark Distributing Computing Work Flow

4 System Architecture

As shown in Figure 2, this system is composed by WebServer, MySQL database and Spark cluster. WebServer is in charge of receiving users' request, storing the files uploaded by users to the file system and invoking the Spark cluster for conversion and storage of videos.

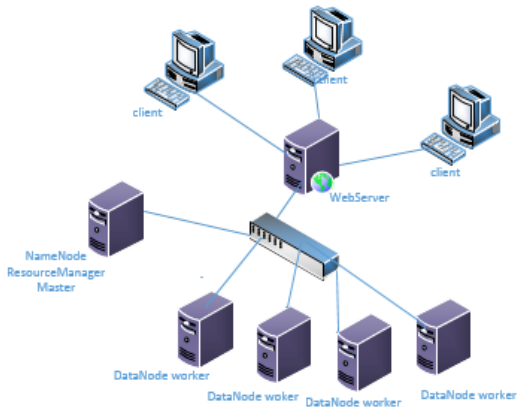

Fig. 2. System Architecture Diagram

The users' request on transcoding is processed by the system with the following steps:

(1) Users deliver their request through http to WebServer, which receives video files and acquires information such as the format and size of the files.

(2) WebServer submits video files to the video segmentation node, which segments video files into independent playable video files, and then submits to hdfs for distributed storage.

(3) Spark cluster conducts distributed transcoding to the distributed file.

(4) Completion on transcoding: After the end of the transcoding, a piece of record (path, name and format of the source file as well as those of the object file) is inserted into the mysql database. Webserver returns the record to the user.

(5) The users then download the object file after transcoding from the file system.

5 Segmentation Strategy for Distributed Storage of Videos

The Hadoop system stores a file with numerous data blocks. Both the MapReduce platform and the Spark platform conduct video transcoding in a concurrent way. Considering of the nonstructural data of video files, there is a correlation between frames in videos, and the simple storage of files into Hadoop may lead to damage to the data information of video files. Therefore, the FFMPEG is utilized firstly for undamaged segmentation, with the size of that of the HDFS block such as 64MB. The size of the video file is generally not the integral multiple of that of HDFS blocks. If the size of the last file is less than that of the HDFS block, it is combined with the previous file. In this way, it is available to reduce the quantity of tasks during concurrent transcoding, so as to improve the performance of the system.

6 The Spark based Scheme

Compared with the MapReduce programming model [11-12], the Spark provides more flexible DAG (Directed Acyclic Graph) programming model, as shown in Figure 3, which includes not only conventional map and reduce interfaces but also filter, flatMap and union operation interfaces, achieving more flexible and convenient video transcoding of the Spark. The logic for the Spark video transcoding business is as shown in Figure 6. It is available to simplify 11 MapReduces to a Spark operation through the DAG programming model of the Spark. The Spark segments the video transcoding process into 3 stages, each of which includes multiple tasks of concurrent execution. The data among stages is transmitted through shuffle. Finally, it is only needed to read and write HDFS once. It eliminates 9 times of HDFS read-write by reducing HDFS read-write by 80%. It applies for needed executor resources to yarn after the start of the park operation, and all the stages operate in a way of thread by sharing executors. Compared with the MapReduce mode, the Spark mode reduces the times for application for resources by nearly 90%. Considering that RDD (Resilient Distributed Dataset) model is utilized for video transcoding, the middle data is stored in a way of RDD, which is stored in contents of slave nodes, reducing the times for disk read-write during the computational process.

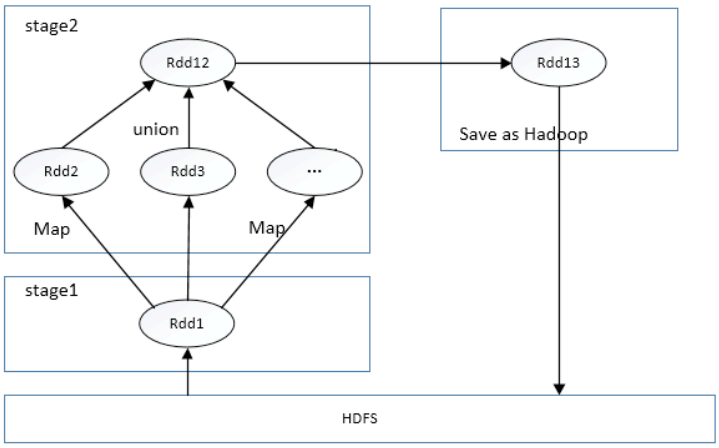

Fig. 3. Directed Acyclic Graph

7 Experimental Results and Analysis

This article constructs 5 sets of virtual machines in an environment of a server with 6-core CPU, 64 GB internal storage, 6TB hard disk through the virtualized software VMware vSphere. The uniform configuration of the virtual machine is listed as follows: 4-core CPU, 8GB internal storage, 500 GB hard disk, CentOS 6 64-bit operating system. The role of the virtual machine is as shown in Table 1.

7.1 Experimental Enviroment

Table 1. Node Roles.

IP address	Node roles
192.168.2.170	Namenode, resourcemanager, Master
192.168.2.171	Namenode, Datanode, resourcemanager, NodeManager, worker
192.168.2.172	Datanode, NodeManager, worker
192.168.2.173	Datanode, NodeManager, worker
192.168.2.174	Datanode, NodeManager, worker

The data adopted in the experiment is that of mov format of 3.02GB, with the following size and quantity of segmentation:

Table 2. Size of segmentation.

Size of segmentation/MB	Quantity of segmentation
64	48
128	24

256	12	

Table 3. Video parameters before and after the transcoding.

Video type	Video size	Video format coding	Bit rate	Resolution
Input video	3.02GB	mov	6972	2560*1600
Output video	476MB	MPEG-4	782	1280*720

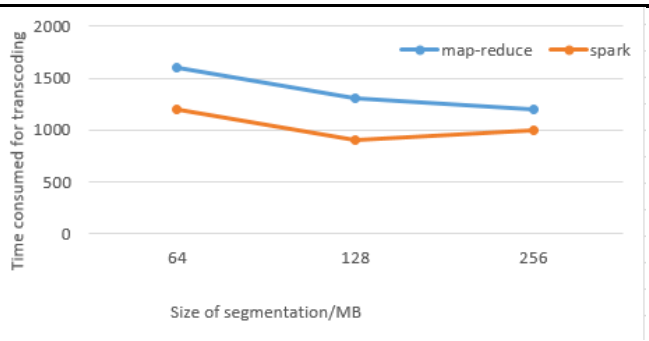

Fig. 4. Size of Segmentation

7.2 Experimental analysis

The time consumed for video transcoding with the FFMPEG tool on a single node is totally 2280s. On the nodes of 4 sets of ApplicationMaster, in condition of the segmentation size of 64MB, it consumes the time of 1600s with the MapReduce. It consumes 1200s on the 4 workers on the spark platform with the segmentation size of 64MB, which is significantly less than that of the MapReduce. The time consumed by Spark is the least with the segmentation size of 128MB. The segmentation size of 256MB consumes more time than that of 128MB, but is still less than that of the MapReduce. Based on the experimental result, the Spark operates faster than MapReduce, with the following main reasons:

(1) It has quicker start-up time for the task; Spark is a fork-out thread while MR is the start-up of a new process;

(2) It achieves faster shuffles; the Spark stores the data in the disk only during shuffle while MapReduce does not;

(3) It achieves faster workflow: The typical MapReduce workflow is composed by many MapReduce tasks, the data interaction among which is realized through data persistence to the disk. The Spark supports DAG and pipelining, and it is available to not store the data in the disk in condition of without shuffle.

During the transcoding with the Spark platform, the time consumed for the segmentation of 128MB is less than that of 256MB, with the following main reason: With the data size of certain degree, the data needed to be loaded into the internal storage during the same operation moment is only a subset of the whole data, and certain time is needed for disc read-write during the process.

8 Conclusion

A distributed scheme with Hadoop based data storage and Spark transcoding mode is proposed in this article. According to experimental results, it saves 25% of time for transcoding when utilizing the Spark platform of 4 sets of worker node cluster than that of application Master of the same nodes. It greatly improves the transcoding efficiency. However, there is a disadvantage in this scheme: it is needed to wait until the end of all the rdds during the reduce stage, which leads to time damage. Aiming at this disadvantage, it is necessary to start the combination between adjacent video segments immediately after the end of the processing to further improve the transcoding efficiency in the future.

Acknowledgements. This work is supported by the Joint Funds of Department of Science & Technology of Guizhou Province (No.qiankehe LH[2015]7725) and the Natural Science Research Project of Education Department of Guizhou Province (No.qianjiaohe KY[2015]348).

References

1. Ahmad I, Wei XH, Sun Y, Zhang YQ. Video Transcoding: An Overview of Various Techniques and Research Issues. IEEE Trans. on Multimedia, 2005,7(5).
2. Barlas G. A Taxonomy and DLT-based analysis of Cluster- based Video Trans/Encoding. 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2006,388–395.
3. Cardellini V, Colajanni M, Lancellotti R, Yu PS. A distributed architecture of edge proxy servers for cooperative trans- coding. The 3rd IEEE Workshop on Internet Applications, 2003,66-70.
4. Guo JN, Bhuyan L. Load Sharing in a Transcoding Cluster. Distributed Computing, 2003, 835.
5. Sambe Y, Watanabe S, Yu D, Nakamura T. Distributed video transcoding and its application to grid delivery. Proc. ITC-CSCC2003, 2003, 921-924.
6. Grio African American Breaking News and Opinion, http://www.thegrio.com/.
7. Borthakur D. The Hadoop Distributed File System: Architecture and Design. http://hadoop.apache.org/core/ docs/current/hdfs_design.html, 2007.
8. Yang Fan and Shen Qiwei. Video Transcoding of the Distributed Hadoop Platform. Computer Systems & Applications, 2011, 20 (11).
9. Project Description[EB/OL], http://www.ffmpeg.org
10. http://spark.apache.org
11. The Apache Hadoop project. http://Hadoop. apache. org/.
12. Dean J, Ghemawat S. MapReduce: Simplied Data Pro- cessing on Large Clusters. Proc. of OSDI'04: 6th Sympo- sium on Operating System Design and Implemention, San Francisco, CA, Dec. 2004.