IEEE Access
The journal for rapid open access publishing

Received November 15, 2015, accepted December 3, 2015, date of publication December 11, 2015, date of current version December 22, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2507788

A Distributed Video Management Cloud Platform Using Hadoop

XIN LIU[1], DEHAI ZHAO[2], LIANG XU[2], WEISHAN ZHANG[2], JIJUN YIN[1], AND XIUFENG CHEN[1]

[1]Hisense TransTech, Ltd., Qingdao 266071, China
[2]Department of Software Engineering, China University of Petroleum, Qingdao 266580, China

Corresponding author: W. Zhang (zhangws@upc.edu.cn)

This work was supported by the International Science and Technology Cooperation Program of China under Grant 2013DFA10980.

ABSTRACT Due to complexities of big video data management, such as massive processing of large amount of video data to do a video summary, it is challenging to effectively and efficiently store and process these video data in a user friendly way. Based on the parallel processing and flexible storage capabilities of cloud computing, in this paper, we propose a practical massive video management platform using Hadoop, which can achieve a fast video processing (such as video summary, encoding, and decoding) using MapReduce, with good usability, performance, and availability. Red5 streaming media server is used to get video stream from Hadoop distributed file system, and Flex is used to play video in browsers. A user-friendly interface is designed for managing the whole platform in a browser–server style using J2EE. In addition, we show our experiences on how to fine-tune the Hadoop to get optimized performance for different video processing tasks. The evaluations show that the proposed platform can satisfy the requirements of massive video data management.

INDEX TERMS Hadoop, J2EE, video processing, MapReduce, video management.

I. INTRODUCTION

With the development of Internet of things (IOT), streaming media applications such as security and traffic video surveillance are deployed rapidly. Streaming media data contains considerable useful information. These massive audio and video data are unstructured data that traditional approaches can't handle storage and processing efficiently. The majority of existing approaches use centralized storage,[1] such as disk array, which is not scalable for efficient massive video data processing.

In order to alleviate these problems, researchers are beginning to make use of the relatively new cloud computing technologies to explore video storage and processing. For example, work on using Hadoop[2] [1], [2] to accelerate video processing time utilizing MapReduce programming style [3], [4], and similar work called Split & Merge architecture was proposed in [5] for high performance video compression. However, these existing work lack details on how to design a versatile video management platform in a user friendly way, and how to effectively use Hadoop to fine tune performance of video processing in order to build a practical and usable video cloud platform.

In this paper, we propose a solution that seamlessly integrates J2EE, and Flex with Hadoop to provide a user friendly way for video management, and also we integrate an open source media server to make our solution economical. H.264 video files stored on HDFS [6], [7] are transcoded to common video format using FFmpeg.[3] Users can download the transcoded files and play it with a video player directly. Furthermore, Gaussian mixture model (GMM) is used to implement video background subtraction. A web based interface is designed using J2EE to provide users an easy way to manage the platform, where Struts2, Spring and Hibernate are integrated.

The contributions of this paper include:

- We propose an universal video data management architecture that integrates J2EE technologies with Red5 media server, and Flex technology, running on top of Hadoop platform.
- We share our experiences on implementing this architecture, and on fine tuning Hadoop in order to achieve good video processing performance.
- We evaluate the platform for its concurrent IO performance, file uploading and downloading, to show the effectiveness of the proposed platform.

[1]http://nffinc.com/sites/default/files/Unified%20VSS%20white%20paper.pdf
[2]http://hadoop.apache.org/
[3]https://www.FFmpeg.org/

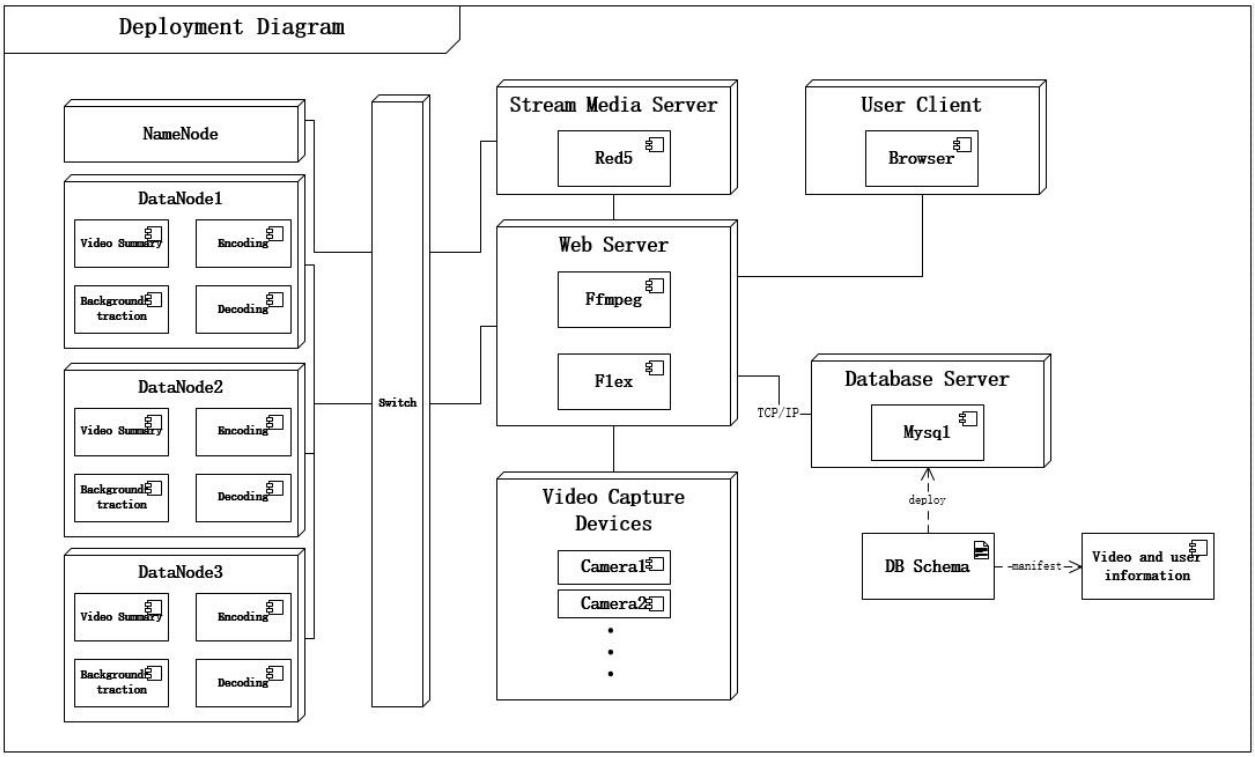


FIGURE 1. Deployment diagram of the platform.

The outline of the paper is as follows: section 2 presents the architecture of the distributed cloud platform, with implementation details discussed in Section 3. Section 4 evaluates the proposed video cloud platform. And section 5 presents some related work. Conclusions and future work end the paper.

II. ARCHITECTURE AND DESIGN OF THE VIDEO MANAGEMENT PLATFORM

As discussed in the introduction, the video management platform should have a browser server style which can improve usability for normal users. Also it will make use of Hadoop platform to make use of the parallel and distributed computing and storage capabilities. The architecture of the video platform described with a UML deployment diagram is shown in Figure 1.

The distributed cloud platform mainly consists of four parts: Hadoop clusters, Red5[4] streaming server clusters, Web servers and database servers. Video stream from video capture devices (e.g. cameras) is stored on HDFS via Web server [8]. Large amount of continuous video data (e.g. in smart surveillance applications) need a huge storage space, but it's not necessary to store all the video from cameras, so some processing is done on the DataNode. For example, video summary can extract specified video that we care about and remove useless video that contain little information. The background subtraction can separate background and foreground effectively, which makes it easier to implement video summary. Meanwhile, video encoding and decoding is an important method to ensure efficient video transmission. Red5 streaming media server uses RTMP, RTMPT, and RTMPE to play audio and video when a users' request is sent to a Web server. There are two important components with a Web server. The first one is FFmpeg, which is a transcoding tool to transcode RTSP stream from cameras to RTMP stream. Another one is Flex, which can get real time stream from a Red5 server. All user information is stored on MySQl database servers. Finally, a user-friendly graphical interface is designed for an easier operation in a browser.

The package diagram of the platform is shown in Figure 2. We can see that the video cloud platform is using a typical three-tier architecture style. The Data package is used for database accessing. The Server package contains service logic components and there are two sub-packages: stream server package and Web server package. The Processing package is used for various video processing features which depend on a OpenCV[5] package. The User package sends requests to services and the responses are displayed by GUIs in a browser.

The class diagram of the platform is shown in Figure 3. Cameras form a group of video capture devices. The View class implements GUIs including displaying user information, playing video, etc. The Management class is the core class

[4]http://www.Red5server.org/

[5]http://opencv.org/

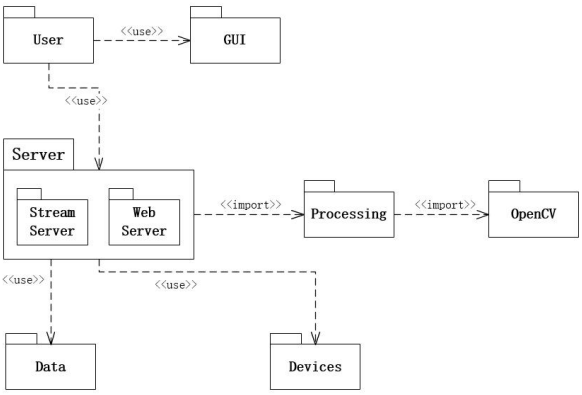


FIGURE 2. Package diagram of the platform.

which plays a role of resource scheduling, user requests response, video stream transcoding, log generating, etc. User class is a set of user information and operation. Similarly, the video class contains video information and operation. Four main video processing classes of the platform are video summary, background subtraction, encoding and decoding. And each genetic class extends the video processing class as its parent class. Lastly, the Database class stores various data and information, like user login data, user group, video time, etc.

III. IMPLEMENTING THE VIDEO CLOUD PLATFORM

There are three important modules in Hadoop: HDFS, YARN and MapReduce. HDFS is the Hadoop distributed file system which can provide high reliability, high scalability and high throughput data services. YARN is responsible for unify management and scheduling of cluster resources. And MapReduce is a distributed computing framework that provides efficient,fault-tolerant, and scalable parallel processing capabilities.

A. DISTRIBUTED STORAGE

HDFS uses a master-slave configuration which includes a NameNode and a number of DataNodes. The NameNode (master) is a central server, and it manages the file system name space and clients' access to files. The DataNode (slave) is the storage space that stores various data. Additionally, Zookeeper[6] cluster is used to manage the active NameNode and the standby NameNode to ensure reliable hot-swap between two NameNode when the active NameNode is

[6]http://zookeeper.apache.org/

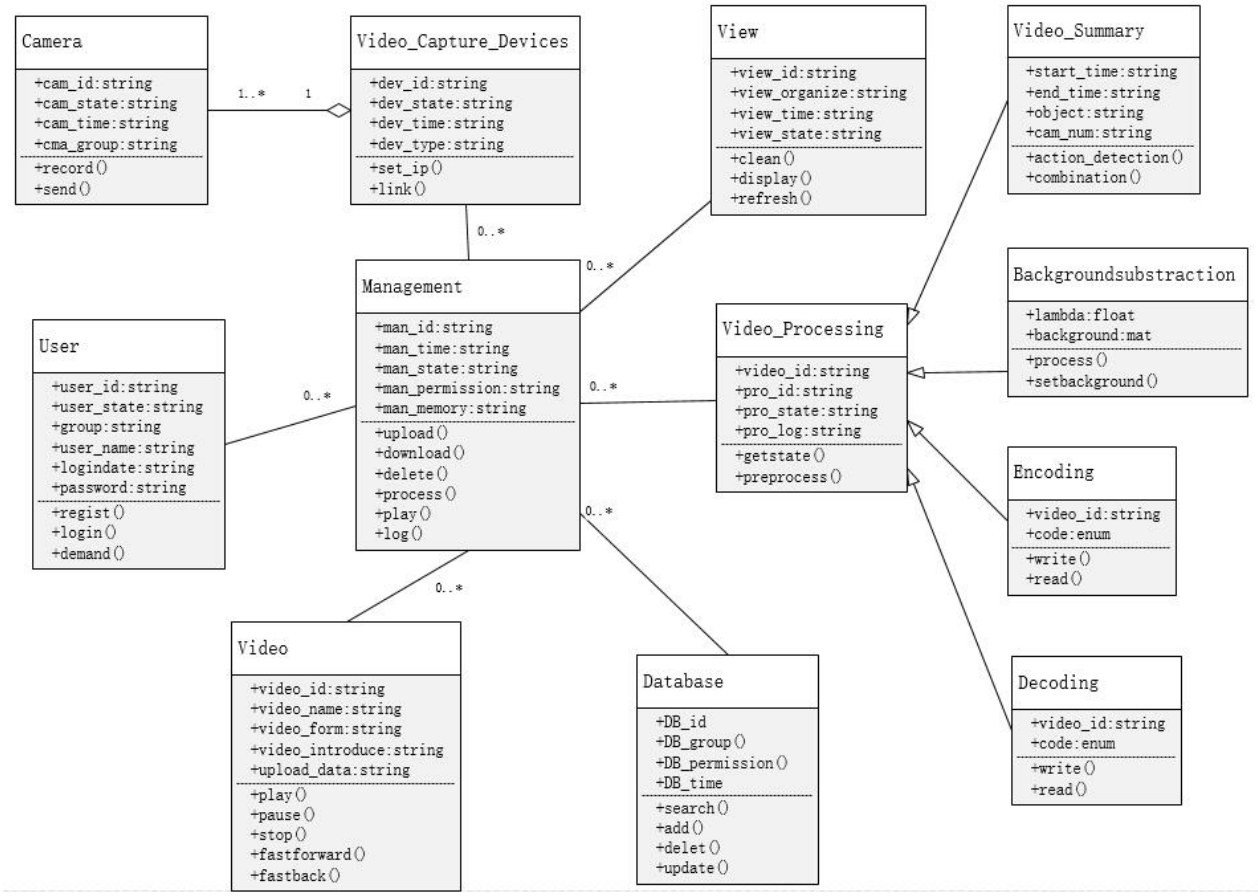


FIGURE 3. Class diagram of the platform.

not working. HDFS has a strong system scalability which supports adding or removing nodes dynamically. Whenever a larger storage space is needed, new nodes can be added online like nothing happens and data can be redistributed to old and new nodes automatically, i.e., the storage space of Hadoop cluster can be infinite large theoretically. Meanwhile, HDFS stores files on blocks with limited size, which can be adjusted according to your own configuration. So the storage strategy of HDFS is to cut those large files to little pieces and scatter them over the storage space in the cluster. HDFS stores files on partition with automatic block replication to implement high fault tolerance and its backup number can also be configured.

To access video files stored in HDFS, users should login to the platform through a client browser. Then they can manage videos, like upload, download, and delete video files if they have such rights. It's hard for users to operate HDFS directly. A user-friendly interface is designed using the browser server architecture style, and J2EE technology is used to implement it to provide a simple and easy operation environment. When a user' requests are sent to a Web server via http protocol, this Web server will call APIs to send reading and writing requests to the HDFS cluster. Then Web server receives the response from HDFS and display it in the browser. As shown in Figure 4, the Web server acts as a relay in this process.

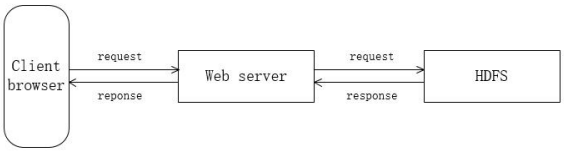


FIGURE 4. Work flow of user requests.

When uploading video files to HDFS cluster, a logged in user will upload video files to the the Web server, and then a *UploadMovieToHDFS()* method is used to upload files to the HDFS cluster. After this, the Web server deletes the cached files. When downloading video files from a HDFS cluster, the Web server queries files address information and apply *DownloadFromHDFS()* method to get files from the HDFS cluster. When deleting video files, the request is sent to the Web server, then the persistence layer query the file address information and apply *DelMovieFromHDFS()* method to delete the file from the HDFS cluster.

B. VIDEO PLAYING

Red5 is an open source streaming media server [9] based on Flash streaming media service [10] developed in Java. It uses RTMP, RTMPT and RTMPE to play audio (MP3) and video (FLV, MP4, etc.) [11], [12]. When the request of playing video is sent by users, the Web server will transcode RTSP stream from cameras to RTMP stream using FFmpeg and send it to Red5 streaming media server. Then the Web server can get real time video stream from Red5 streaming media server using Flex. The process is shown in Figure 5.

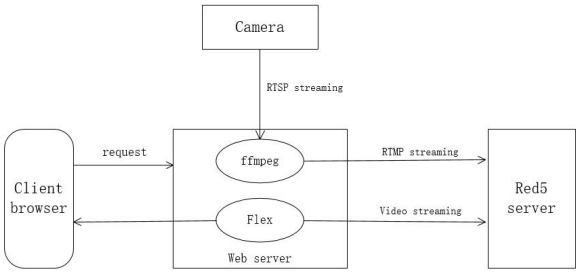


FIGURE 5. Work flow of video transcoding.

The transcoded video stream from camera is stored on HDFS. A mount is made using fuse-dfs between Red5 streaming media server and HDFS cluster and a directory named *Streams* for the mount directory is created. The process is shown in Figure 6.

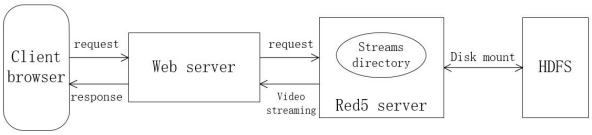


FIGURE 6. Work flow of video playing.

The Red5 streaming media server scans FLV files from the *Streams* directory and transcodes them to video stream. The Web server receives video stream from the Red5 streaming media server and plays it in a browser.

C. BACKGROUND SUBTRACTION

One of the most important video processing task is background subtraction. There are lots of algorithms to achieve this. The mixture of Gaussians (MOG) is used widely. We use an improved Gaussian model designed by Zivkovic [13]. The advantages of this algorithm are that the number of Gaussian components is constantly adapted per pixel and that it can perform shadow detection.

IV. EVALUATION

We have built a distributed cloud cluster to test our the video cloud platform. The cluster contains a gigabit Ethernet switch and five Dell XPS 8700-R37N8 computers with the label of node1 to node5. Some necessary softwares such as OpenCV and CUDA are installed. Four roles are distributed on these five nodes. They are NameNode (NN), DataNode (DN), Resourcemanager (RM) and Nodemanager (NM) as shown in Table 1.

TABLE 1. Distribution of each role on the test bed.

pc	node1	node2	node3	node4	node5
role	NN, RM	DN, NM	DN, NM	DN, NM	DN, NM

A. PERFORMANCE EVALUATION

The first test is Hadoop benchmark test (system I/O test). We use MapReduce to test the concurrent reading and writing

TABLE 2. Results of Hadoop benchmark test.

backup number	number of Map	writing			reading		
		Throughput (mb/sec)	Average IO rate (mb/sec)	exec time (sec)	Throughput (mb/sec)	Average IO rate (mb/sec)	exec time (sec)
1	4	45.035	45.035	588.505	27.35	27.352	951.76
	64	28.716	31.5	200.11	14.474	18.344	302.135
3	4	23.864	23.865	1087.683	67.14	77.524	463.534
	64	5.446	5.744	722.01	16.092	19.475	284.191

of the platform in order to show the throughput performance and HDFS I/O performance. In the test, we upload and download a file of size 100G. The results are shown in Table 2.

Throughput is the average throughput of each Map task. **Average I/O rate** is the average I/O rate of each file, and **exec time** shows the total time of the whole process. We can see clearly that for the reading process, it costs more time when the backup number is three and it costs less time when the backup number is one. But for the writing process, it is a different situation. Therefor the backup number should be adjusted according to actual needs. When the backup number is fixed, the more concurrency there are, the larger throughput and less processing time the platform can achieve. It is important to note that the writing speed is higher than the reading speed when the backup number is one, but the reading speed is higher than the writing speed when the backup number is three. The reason why we get this result is the difference between writing method and reading method of HDFS. The writing process is a pipeline, that is to say, when writing a video file, the video file is segmented to blocks and the blocks are segmented to smaller blocks called package. The package is written on the first DataNode and copied to the second DataNode from the first DataNode. Then the package on the second DataNode is copied to the third DataNode and so on. So the writing time increases with the increasing of backup number. However, the reading process is parallel, that is, the data blocks can be read from each DataNode which stores them and combined as a complete video file. So the reading speed will be higher if the backup number becomes larger.

In addition, a test of uploading a file of size 36G with single process has been done to verify the conclusion of Table 2 that the writing speed is higher than the reading speed when the backup number is one, but the reading speed is higher than the writing speed when the backup number is three. The results are shown in Table 3, with different number of backups.

B. TUNING THE PERFORMANCE

In this test, we will conduct a video summary with a 17.5 hours video file, whose size is 37G bytes with a frame rate of 25 FPS and the resolution is 1280*720 pixels. There are two approaches to do this. The first one is segmenting raw video data to clips and using the clip names as the input of map, then reading video files according to the clip address and processing them in the map function. The other one is also segmenting raw video data to clips, but using clip address as key and video files as value, then combining them into SequenceFile and processing it as the input of MapReduce. The second one is a good choice due to potentials to reduce data transmissions. We compare the above two approaches and the results are shown in Table 4. We can see that for the second approach, the performance is better than the first one.

TABLE 3. Results of backup number test.

backup number	upload time(sec)	download time(sec)
1	337.184	418.788
3	524.721	399.593

TABLE 4. Results of video summary.

scheme	system1	system2	system3
1	4h11m5s	3h37m25s	2h42m39s
2	3h44m18s	3h17m58s	2h35m23s

All block sizes of the Hadoop platform are set to 64M. The only difference is the memory size of Nodemanager. They are 8G, 10G, 12G for system1, system2 and system3 respectively.

In a practical application, we should optimize the platform for a better performance. In general, video processing needs a lot of memory. Hadoop provides *usemapreduce.map.memory.mb* and *mapreduce.reduce.memory.mb* to modify the memory of each map task and reduce task depending on actual requirements.

There are more than 190 parameters in a Hadoop configuration file and 25 of them have remarkable effect on the efficiency of Hadoop applications. We test some parameters which are very important for video processing. They are:

- *yarn.nodemanager.resource.memory-mb* represents the total physical memory that is available to YARN on one node. YARN can not detect the total available physical memory of one node automatically. So we should adjust it according to the available memory resource of a node.
- *dfs.block.size* represents the block size on HDFS. Meanwhile, it determines the number of maps in the default MapReduce framework.
- *dfs.replication* represents the backup number. Increasing backup number can improve map's hit probability of local data. However, this figure should be not be big for big data processing applications.

The parameter optimization results are shown as follows, in which Table 5 is a detail illustration of schema 2 in Table 4. It can be seen obviously that the processing speed increases with the increasing of memory. And Table 6 shows the effects of different block size on processing time. We can see that it has the shortest processing time when the block size is 64M,

TABLE 5. Effects of different memory size.

yarn.nodemanager. resource.memory-mb	8G	10G	12G
time	3h44m18s	3h17m58s	2h35m23s

TABLE 6. Effects of different block size.

dfs.block.size	32M	48M	64M
time	2h41m39s	2h37m11s	2h35m15s

because the video files have been segmented to 64M. So the the clip size should be close to the HDFS block size to reduce processing time.

V. DISCUSSION

We have encountered many problems during the development of the video platform.

- During the development of the video data storage module, we can not upload video files to HDFS cluster with HDFS APIs directly. The solution to the problem is uploading video files to the cache of Web server. Then the video files will be upload to the HDFS cluster from the Web server using HDFS APIs.
- For retrieving real-time streaming, we can not get streaming data from IP cameras directly. The solution to the problem is transcoding RTSP stream to RTMP stream with FFmpeg. Then we use Flex to receive video stream from the Red5 streaming media server.
- We find that Ajax is a good solution to enhance user experience because it can optimize the transmission between browser and Web server, and reduce data transmission footprint and load time of Web server.
- The parameter setting of Hadoop can affect video processing a lot, and we need to carefully adjust some of the parameters in order to reduce the data transmissions, and improve the performance.

VI. RELATED WORK

Myoungjin Kim et al. proposed a Hadoop-based distributed video transcoding system in a cloud computing environment, which can transcode various video format to MPEG-4 [14]. They design and implement the platform efficiently using MapReduce framework. Comparing with their work, besides transcoding, we develop a user-friendly interface to operate HDFS easily, details on how to use Hadoop efficiently by tuning different parameters are also presented.

In [15], the authors proposed a Hadoop-based storage architecture for massive MP3 files. They used classification algorithm in pre-processing module to merge small files into sequence files. They confirm that the introduced efficient indexing mechanism is a good solution to the problem of small files. Our work focus on Hadoop-based storage and processing architecture for big video data, and usability is one of our focuses.

Lin *et al.* [16] proposed a prototype of cloud based video recording system. The system can provide scalable video recording, data backup and feature detection. They also used HDFS to store video data, which is similar with ours. In our work, we apply J2EE technology to develop a user-friendly interface for HDFS, and we use Red5 to play video. Additionally we consider fine tuning Hadoop is an important task for practical big data applications.

In [17], the authors studied and analyzed the open source Red5 streaming media server and they used Flex and Red5 streaming media server for playing video successfully. In our research, we mainly concentrate on Hadoop based management of video data using Flex and Red5 streaming media server. In our work we combine Flex, Red5 and Hadoop together to implement our platform.

Liu *et al.* [18] present a framework for video playing and video storage based on Hadoop. Their framework provides high availability services, which could support concurrent access and playing streaming media in mobile terminals. Our work is similar with their work, but we target a user friendly big video data management platform and we also present our experiences on using J2EE for the management of video data, and experiences on optimizing the video management platform with Hadoop.

In [19], the authors presented a video monitoring system that can meet the users' demands of searching video, uploading video, downloading video and transcoding video. They also used FFmpeg to transcode video. In our work, we focused on the usability of the video management platform, plus the experiences on integration of J2EE, Flex, and Hadoop.

In our previous work [20], we did some preliminary research on video monitoring platform, where Hadoop based processing was explored. In this paper, we focus on a user friendly video management platform with its architecture, and sharing our experiences on fine tuning the performance of Hadoop based video processing.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we present a distributed cloud platform for storing and managing massive video data based on the integration of Hadoop, Red5 streaming media server and J2EE. The platform is economical because the cluster can be built by personal computer, and the software packages are open source. We have proposed to use Red5 streaming media technology to realize the video playback while downloading. We have also evaluated the performance of the platform extensively and it shows that the platform has good performance. We have also found that the parameter setting of Hadoop can greatly affect the performance of the video processing.

We are experimenting to use the presented approach for smart transportation system in Qingdao City, China. Additionally we are adopting an online processing and offline processing combined approach [4] in order to make full of the cloud computing potentials. Another direction is that we plan to use deep learning to do knowledge mining from video data [21].

REFERENCES

[1] J. Venner, *Pro Hadoop*. New York, NY, USA: Apress, 2009.

[2] T. White, *Hadoop: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2012.

[3] H. Tan and L. Chen, "An approach for fast and parallel video processing on Apache Hadoop clusters," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2014, pp. 1–6.

[4] W. Zhang, L. Xu, P. Duan, W. Gong, Q. Lu, and S. Yang, "A video cloud platform combing online and offline cloud computing technologies," *Pers. Ubiquitous Comput.*, vol. 19, no. 7, pp. 1099–1110, Oct. 2015.

[5] R. Pereira, M. Azambuja, K. Breitman, and M. Endler, "An architecture for distributed high performance video processing in the cloud," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2010, pp. 482–489.

[6] X. Wang and J. Su, "Research of distributed data store based on HDFS," in *Proc. 5th Int. Conf. Comput. Inf. Sci. (ICCIS)*, Jun. 2013, pp. 1457–1459.

[7] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. Oper. Syst. Design Implement.*, 2006, pp. 307–320.

[8] F. Azzedin, "Towards a scalable HDFS architecture," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, May 2013, pp. 155–161.

[9] G. Shi, Z.-P. Li, and Z.-L. Xu, "Design and implementation of real-time video streaming system based on Red5," *Instrum. Technol.*, vol. 6, p. 6, Jun. 2010.

[10] J. Emigh, "New Flash player rises in the Web-video market," *Computer*, vol. 39, no. 2, pp. 14–16, Feb. 2006.

[11] M. Lu and X. D. Wang, "Design of video conference system based on techniques of FLV streaming media," *J. Jilin Univ. Inf. Sci. Ed.*, vol. 28, no. 2, pp. 186–190, 2010.

[12] D. Wang and K. Xu, "Red5 Flash server analysis and video call service implementation," in *Proc. IEEE 2nd Symp. Web Soc. (SWS)*, 2010, pp. 397–400.

[13] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 2. Aug. 2004, pp. 28–31.

[14] M. Kim, Y. Cui, S. Han, and H. Lee, "Towards efficient design and implementation of a Hadoop-based distributed video transcoding system in cloud computing environment," *Int. J. Multimedia Ubiquitous Eng.*, vol. 8, no. 2, pp. 213–224, 2013.

[15] X.-Y. Zhao, Y. Yang, L.-L. Sun, and Y. Chen, "Hadoop-based storage architecture for mass MP3 files," *J. Comput. Appl.*, vol. 32, no. 6, pp. 1724–1726, 2012.

[16] C.-F. Lin, S.-M. Yuan, M.-C. Leu, and C.-T. Tsai, "A framework for scalable cloud video recorder system in surveillance environment," in *Proc. 9th Int. Conf. Ubiquitous Intell. Comput., 9th Int. Conf. Auto. Trusted Comput. (UIC/ATC)*, Sep. 2012, pp. 655–660.

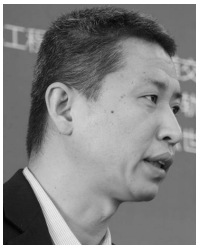
[17] D. Wang and K. Xu, "Red5 Flash server analysis and video call service implementation," in *Proc. IEEE 2nd Symp. Web Soc. (SWS)*, Aug. 2010, pp. 397–400.

[18] C. Liu, K. Fan, Z. Yang, and J. Xiu, "A distributed video share system based on Hadoop," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Intell. Syst. (CCIS)*, Nov. 2014, pp. 587–590.

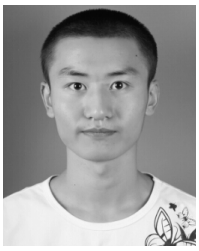
[19] Y. Liu, S. Liang, M. Zhao, and Y. Sun, "Hadoop applications in video monitoring," in *Advanced Materials Research*, vols. 1030–1032. Dürnten, Switzerland: Trans Tech Publications, 2014, pp. 1717–1720.

[20] W. Zhang, X. Ding, and R. Hou, "Hadoop and PaaS collaborative practice research in video monitoring platform," in *Applied Mechanics and Materials*, vols. 543–547. Dürnten, Switzerland: Trans Tech Publications, 2014, pp. 3549–3554.

[21] W. Zhang, P. Duan, Z. Li, Q. Lu, W. Gong, and S. Yang, "A deep awareness framework for pervasive video cloud," *IEEE Access*, vol. 3, pp. 2227–2237, Nov. 2015.



XIN LIU is currently an Assistant Chief Engineer with Hisense TransTech Company, Ltd. His main research interests include smart transportation, video processing, and smart cities.

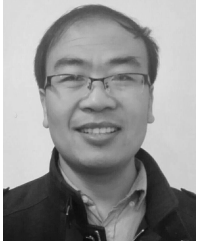


DEHAI ZHAO is currently pursuing the master's degree with a focus on big data processing systems, food recognition using deep learning, and big data mining topics. He has published two papers on these areas.

LIANG XU is currently pursuing the master's degree with a focus on big data processing systems, food recognition using deep learning, and big data mining topics. He has published five papers on these areas, including reputed journal like *Personal and Ubiquitous Computing*.

WEISHAN ZHANG was a Research Associate Professor/Senior Researcher with the Computer Science Department, University of Aarhus (till 2010), where he had been working on the EU FP6 Hydra pervasive middleware project, for which he was a Technical Manager (2008–2009). He was a Visiting Scholar with the Department of Systems and Computer Engineering, Carleton University, Canada (2006–2007). He was an Associate Professor with the School of Software Engineering, Tongji University, Shanghai, China (2003–2007). He was a NSTB Post-Doctoral Research Fellow with the Department of Computer Science, National University of Singapore (2001 to 2003). He is currently a Full Professor, the Deputy Head of research with the Department of Software Engineering, School of Computer and Communication Engineering, China University of Petroleum. He is the Director of the Big Data Intelligent Processing Innovation Team of Huangdao District, the Director of the Smart City Research Center of Fuwode Electronic Corporation, and the Founding Director of the Autonomous Service Systems Laboratory. He has an h-index of 13, according to Google Scholar, and the number of total citations is around 490 in 2015.

JIJUN YIN is currently a Researcher with Hisense TransTech Company, Ltd. His main research interests include smart transportation, video processing, and smart cities.



XIUFENG CHEN received the Ph.D. degree from the University of Jilin, in 2013. He holds a postdoctoral position with Hisense TransTech Company, Ltd., Qingdao, China. His research interests include traffic system analysis and intelligent transportation technology.