future internet

Article

Turning Video Resource Management into Cloud Computing

Weili Kou *, Hui Li and Kailai Zhou

School of Computer and Information, Southwest Forestry University, Kunming 650224, China; yncq24212@163.com (H.L.); zkl2@163.com (K.Z.)
* Correspondence: kwl@swfu.edu.cn; Tel.: +86-137-006-8673

Academic Editor: Dino Giuli
Received: 13 December 2015; Accepted: 12 July 2016; Published: 21 July 2016

Abstract: Big data makes cloud computing more and more popular in various fields. Video resources are very useful and important to education, security monitoring, and so on. However, issues of their huge volumes, complex data types, inefficient processing performance, weak security, and long times for loading pose challenges in video resource management. The Hadoop Distributed File System (HDFS) is an open-source framework, which can provide cloud-based platforms and presents an opportunity for solving these problems. This paper presents video resource management architecture based on HDFS to provide a uniform framework and a five-layer model for standardizing the current various algorithms and applications. The architecture, basic model, and key algorithms are designed for turning video resources into a cloud computing environment. The design was tested by establishing a simulation system prototype.

Keywords: video resources; big data; cloud computing; HDFS

1. Introduction

With the development of information technologies, tremendous challenges have emerged in video resource management. Transferring this work into online service systems is a feasible solution for resolving the problems. Video resources with huge amounts of volumes and complex data types are a kind of typical big data, and are difficult for processing. Cloud services have been regarded as a significant trend of technical industries and applications after the Web services era. Generally, the framework of cloud services consists of infrastructures, operating systems, virtual machines, storages, and cloud web application services. With the improvement of global network performance over the past few years, research on sharing video resources with large volumes and complex data types online based on cloud computing has attracted more and more attentions [1]. Video processing is a notably data-intensive, time-consuming, and computing-intensive application. Upfront infrastructure investment is usually costly, especially when dealing with applications where time-to-market is a crucial requirement [2]. Traditionally, a great volume of video resources is generated, stored, and managed on local servers daily. That obviously incurs some problems in keeping daily video resources in data centers, such as limitations of bandwidth, storage space, overloading, reliability, and scalability [3]. When the number of users online reaches a certain scale, the limitation of bandwidth will greatly influence the accessing speeds of response servers. Meanwhile, data servers need to deal with the loading balances between huge volumes of video resources and users, and bring buffer a phenomenon. This paper proposes a novel architecture based on well-developed peer-to-peer (P2P) technology and emerging cloud computing aiming to solve these issues. The architecture exploits inherent characteristics of P2P and cloud computing to provide an economic, scalable, reliable, and efficient model to manage video resources. This paper focused on architecture, components, operation flows, and implementation of video resource management systems. Video resources always being unavailable to users due to their huge volume makes the access speed slow and overloads the storage media. The emergence of cloud computing provides a new solution to efficiently manage video resources.

Hadoop is an open-source software framework that supports data-intensive distributed applications, running off applications on large clusters of commodity hardware. It also transparently provides both reliability and data motion to applications. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in computing clusters. In addition, it provides a distributed file system that stores data across data nodes with very high aggregate bandwidth. Both Map/Reduce and distributed file systems are designed to automatically handle problems of node failures. It enables applications to work with thousands of computation-independent computers and petabytes of data.

2. Related Work

Web systems based on traditional storage servers are mainly used for video resources management. Low efficiency is the bottleneck of these systems. For improving the management efficiency, clusters are used in the systems. With the increasing volumes, data types, and complex application scenarios of video data (especially single video with a large size), current systems cannot meet their management needs. At present, Hadoop-based cloud-computing technologies are extensively used in various fields, such as video transcoding systems [4], data processing cross-platforms [5], and multimedia conferences [6]. Scalable and fault-tolerant architecture that supports the parallel processing of large volumes of video data is becoming increasingly necessary for flexible, robust, and efficient processing of large volumes of data [2]. However, current video resource management systems make it difficult to manage videos in big volumes. Additionally, these applications lack uniform architecture and layer models to conveniently share resources and efficiently communicate each other. This paper focuses on building a video resource management system serving various users in a cloud-computing environment based on Hadoop Distributed File System (HDFS).

This paper tries to design a system that utilizes HDFS based on a cloud server. Thereby, HDFS and the Map/Reduce framework are briefly introduced. Hadoop inspired by Google's Map/Reduce and Google File System (GFS) [7] is a software framework that supports data-intensive distributed applications handling thousands of nodes and petabytes of data [8]. It can perform scalable and timely analytical processing of large data sets to extract useful information. Hadoop consists of two important frameworks: (1) (HDFS) (Figure 1), like GFS, is a distributed, scalable and portable file system written in the Java programming language; (2) Map/Reduce (Figure 2) which is the first framework developed by Google for processing large volumes of data sets [9].

The Map/Reduce paradigm is a framework for processing huge datasets of certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster [10]. It consists of an initial Map stage, where a master node takes the input, chops it into smaller or sub-problems, and distributes the parts to worker nodes, which process the information; next is the Reduce stage, where the master node collects the answers to all the sub-problems and combines them to produce the job output. A popular Map/Reduce implementation is Apache's Hadoop, which consists of one Job Tracker, to which client applications submit Map/Reduce jobs. The Job Tracker pushes work out to available Task Tracker nodes in the cluster, which execute the Map and Reduce tasks [11]. This paper proposes architecture for video resource processing and online management in a HDFS-based cloud-computing environment, which is sufficiently flexible to be deployed in either a private or public cloud environment.

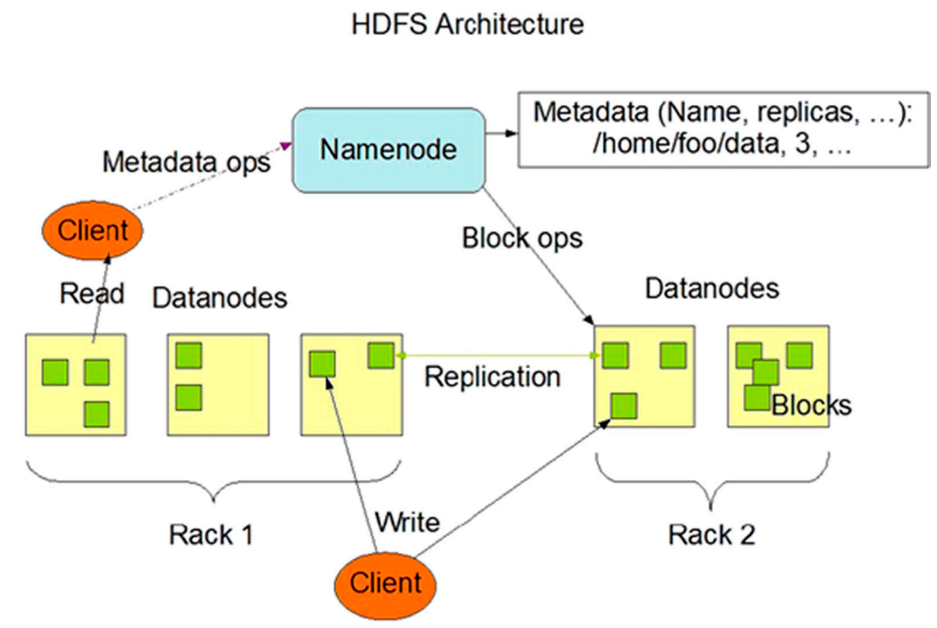


Figure 1. The architecture of Hadoop File System (HDFS).

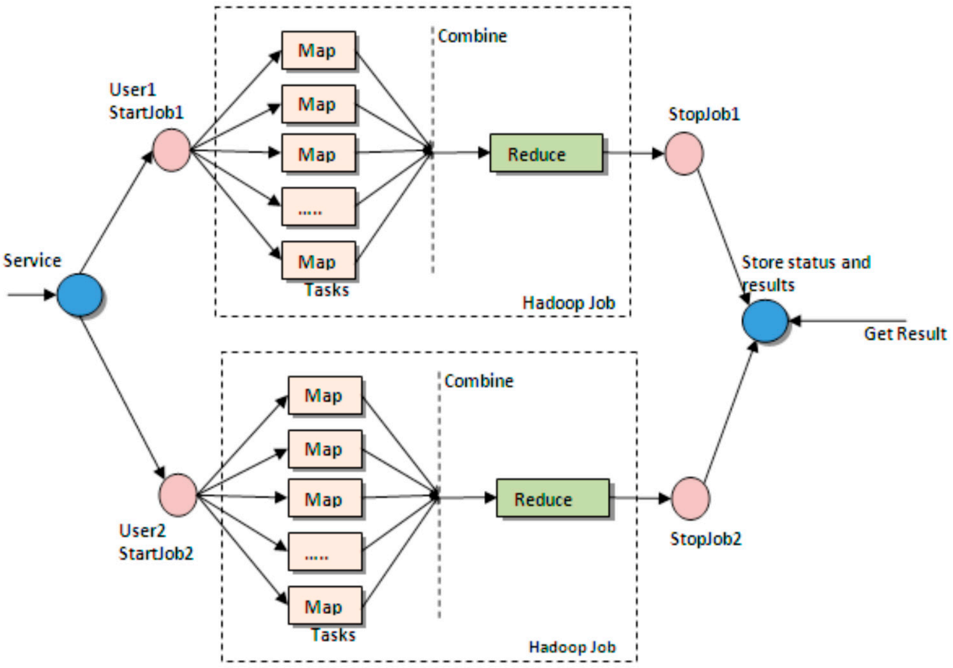


Figure 2. The Map/Reduce paradigm.

3. The Video Resource Management System in HDFS-Based Cloud Computing

3.1. System Layers

The model of video resource management service systems (Figure 3) consists of five layers from top to bottom: the client, middleware, application server, storage, and infrastructure layers. (1) The client layer is responsible for receiving various requests of users and displaying results on available web browsers, such as Internet Explorer (IE), Firefox, and Chrome. (2) The middleware layer is used to manage computer resources and network communications. It is a connector between the client and application layers, so applications can run across heterogeneous hardware and software platforms.

(3) The application service layer can implement all kinds of services including user management, video searching, and video playing. (4) The storage layer virtualizes the infrastructure resources as a file system and provides distributed data storage services for users. HDFS is employed to implement the function of this layer. (5) The infrastructure layer consists of devices for computing, storing, and communications (such as the hard disk and memory).

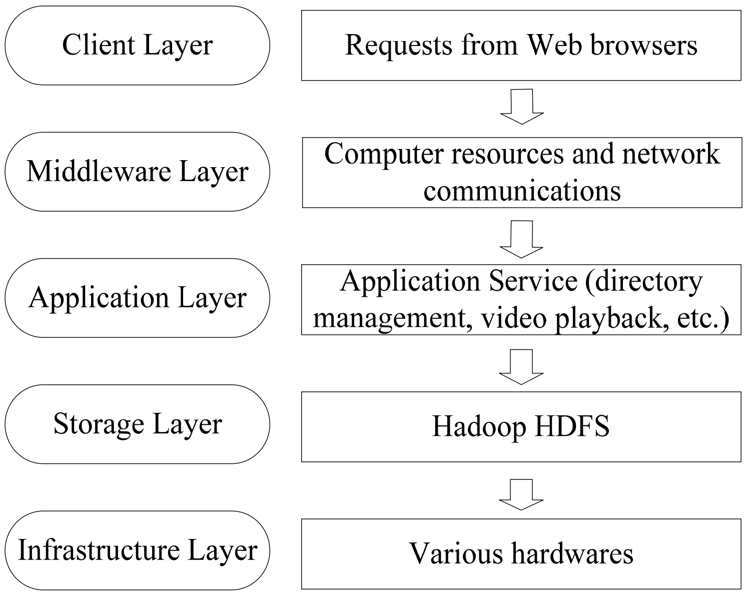

Client Layer
Requests from Web browsers
Middleware Layer
Computer resources and network communications
Application Layer
Application Service (directory management, video playback, etc.)
Storage Layer
Hadoop HDFS
Infrastructure Layer
Various hardwares

Figure 3. HDFS-based five-layer model for video resource management.

3.2. HDFS-Based Architecture of Video Resource Management

The model of video resource management systems (Figure 4) consists of clients, web clusters, streaming media servers, and HDFS clusters. These components work cooperatively. Firstly, a user sends a video playing request to web clusters by selecting a video listed on a web page through the HyperText Transfer Protocol (HTTP); secondly, web clusters search the video information according to the user's request from the MySQL database, and return the search results to the HDFS clusters to retrieving the video files; thirdly, video HDFS clusters send the video information to streaming media servers; fourthly, streaming media servers read, compress, and fetch the video streaming data from the HDFS clusters; finally, clients get the continuous video stream from the streaming media servers.

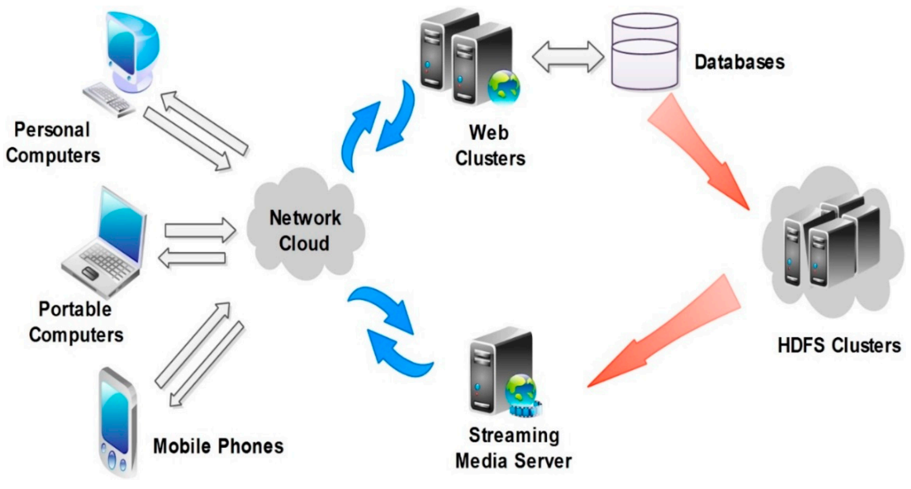


Figure 4. HDFS-based architecture of video resource management.

3.2.1. The Clients

The clients provide functions of video information searching, video playing control, etc. Video detailed attributes are stored in Web Clusters, and maintained (such as update, upload, and delete manipulations) by administrators. Users can get continuous streaming media information that is encoded, compressed, and cached through sending a request to the HDFS server for access to videos.

3.2.2. The Web Service Clusters

The web service cluster is a crucial part of the video resource service systems, as it is a direct interface for users to search, upload, and play videos. The web clusters consist of Apache and Tomcat, and load balancing is implemented by Apache, Tomcat and mod_jk. The advantages of the integration of Apache and Tomcat are as follows: (1) It improves the performance of the whole video resource service system. The dynamic web pages are processed by Tomcat, and the static web pages are managed by Apache; (2) The integration of Apache and Tomcat can realize better load balancing of video resource service systems, where the clients get responses from clusters instead of traditional servers. For improving the performance of web servers, Apache as a proxy distributes client requests to every Tomcat, and the web server as a cluster service processes client requests; (3) Applications and fault tolerance are upgraded seamlessly. In the web server clusters environment, if a web server is suspended, another available one will instantly instead and continue its affairs. The web server with the newest version applications will automatically upgrade the one with the out-of-date in background, and client users do not feel the updating process. The functions of the web clusters include releasing, uploading, and playing videos online, etc. (1) The video-playing module provides video searching and online playing services. When users submit a playing request to the video service system, the video information is retrieved from MySQL database servers according to user requests and the results are returned; (2) The video uploading function is improved. Users upload videos to web servers and the system automatically stores the video attributes into MySQL database servers that can respond to manipulation requests for retrieving video information from web servers such as query, update, and delete. The videos are audited and stored in the HDFS clusters.

3.2.3. HDFS-Based Video Resource Clusters

The feature of writing once and reading many times is a main advantage of HDFS clusters, and is very fit to applications needs of video resource management. A HDFS cluster is a distributed file system that is made from a large amount of computers with low costs in resources and costs. HDFS video resource clusters are used to store and manage the huge number of video resources, also a data source of streaming media servers.

3.2.4. Streaming Media Servers

Streaming media servers are mainly used to provide video storage, playback, and related controls. Reading video data from HDFS clusters according to user requests and sending them to the clients after the streaming processing is the main task of streaming media servers. Streaming media servers that are employed are the Routing Table Maintenance Protocol (RTMP), Red5 and Flowplayer to real-time control video streaming, which includes playback, fast-forward, pause and so on. Interactions between clients and streaming media servers are the main functions of streaming media servers in online video service systems. (1) Client users get video playback resources from streaming media servers through sending video playback requests to the web servers which retrieve video information from MySQL and HDFS servers; (2) Streaming media servers fetch caches and play the video according to the client user's requests; (3) When streaming media servers receive a control command request (such as playback, pause, and fast forward), it controls video streams in terms of commands of users;

(4) When video streams are completely transferred or a stop command is sent by users, the streaming media server shuts the connection and ends a playback period.

3.3. Key Algorithms

3.3.1. Pseudo Codes for Uploading Video Files from Local Servers to HDFS Clusters

```
public void copytoHDFS()
{
//Create configuration property object
     Configuration conf = new Configuration();
//Fetch Hadoop configuration information
      conf.addResource(new Path(str_conf));
//Create file sysytem object
      FileSystem hdfs = FileSystem.get(conf);
//Get absolute path of local files
      Path src = new Path(src_conf);
//Upload files to specified HDFS directories
      Path dst = new Path(dst_conf);
//Upload files
      hdfs.copyFromLocalFile(src, dst);
}
```

3.3.2. Pseudo Codes for Reading Video Information from HDFS Clusters by Streaming Media Servers

```
public void readFromHDFS
{
    //Create configuration property object
      Configuration conf = new Configuration();
//Create file system object
      FileSystem hdfs = FileSystem.get(conf);
//Call FSDataInputStream function
      FSDataInputStream hdfsInStream = fs.open();
//Declare a array
    byte[] ioBuffer = new byte[10240];
//Read the length of the array
      int readLen = hdfsInStream.read(ioBuffer);
// Start an array for writing data
    while(readLen!=-1)
      {
          System.out.write(ioBuffer, 0, readLen);
          readLen = hdfsInStream.read(ioBuffer);
      }
//Close data stream
    hdfsInStream.close();
//Close HDFS
    fs.close();
}
```

4. Experiments and Results

To verify the performance of this design, an experimental prototype of the video service system was established in laboratories.

4.1. Facilities and Configurations

In the experiment, four personal computers were used to implement the design; three of them were configured as data nodes of the HDFS for web clusters and streaming media servers. The details of the configuration are shown in Tables 1 and 2, and the software installed on the web cluster servers are listed in Table 3.

Table 1. Hardware and their parameters.

ID	CPU	Memory	Hard Disk	Node Type	Roles
1	Intel(R) Pentium(R)4 CPU 3.06 GHz	2G	80GB	Namenode master	Web server for streaming medias
2				Datanode slave	Web server
3					/
4					/

Table 2. Network configurations.

Node Type	IP Address	Host Name
Namenode, master	172.16.10.11/24	Hadoop1.com
Datanode, slave	172.16.10.12/24	Hadoop2.com
Datanode, slave	172.16.10.13/24	Hadoop3.com
Datanode, slave	172.16.10.14/24	Hadoop4.com

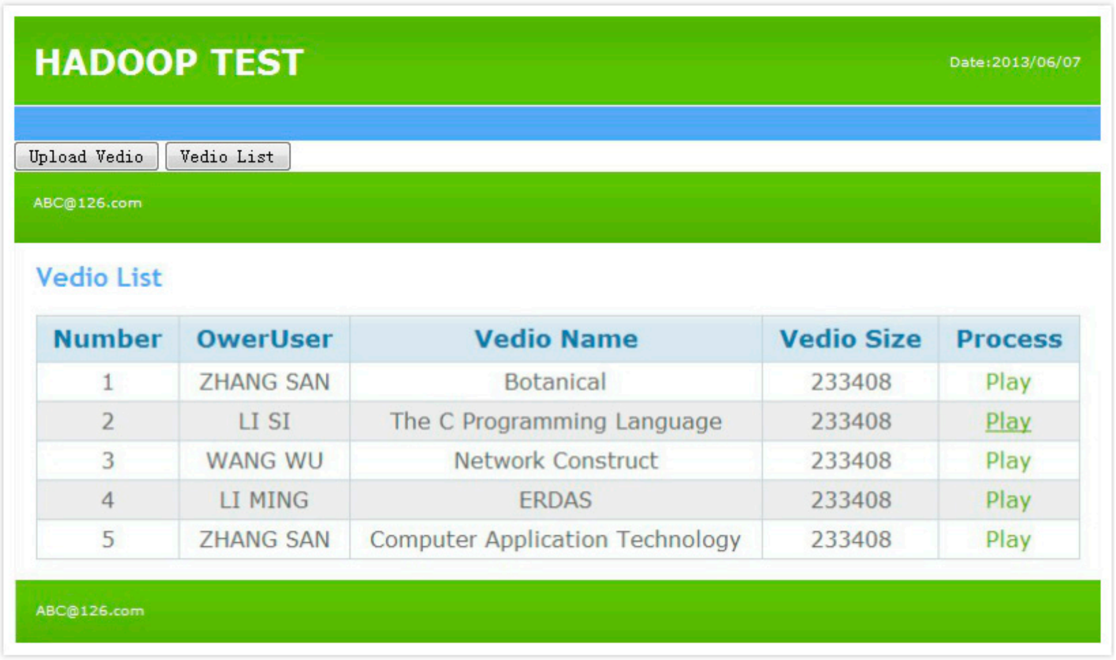
Table 3. Software installed on servers.

Host name	IP address	Software	Version
Hadoop1.com	172.16.10.11/24	apache-tomcat-7.0.39.tar.gz	7.0.39
		httpd-2.2.24.tar.gz	2.2.24
		tomcat-connectors-1.2.37-src.tar.gz	1.2.37
		jdk-6u35-linux-i586-rpm	6u35
Hadoop2.com	172.16.10.12/24	apache-tomcat-7.0.39.tar.gz	7.0.39
		tomcat-connectors-1.2.37-src.tar.gz	1.2.37
		jdk-6u35-linux-i586-rpm	6u35

4.2. The Experimental Results

4.2.1. Functional Testing

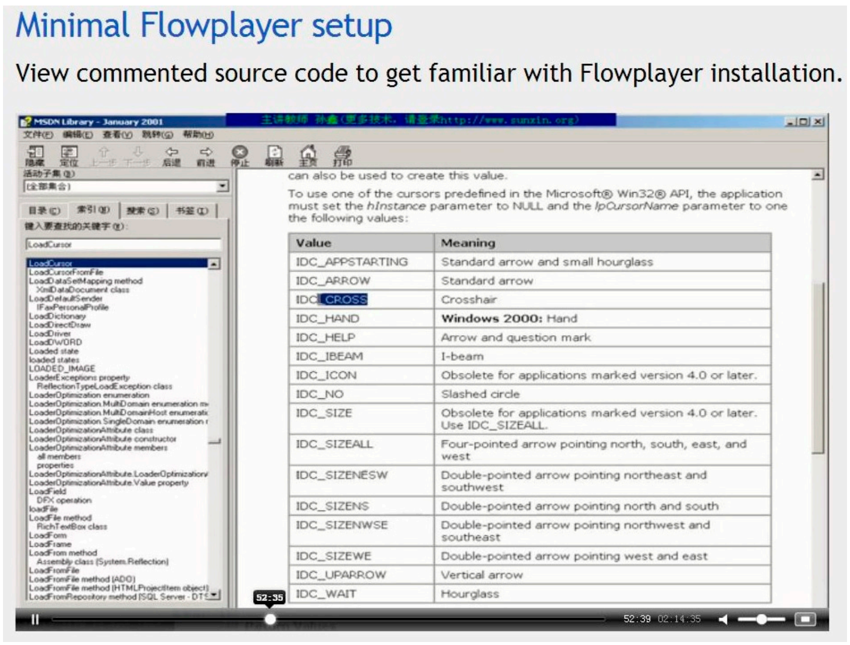
Figure 5 is a playlist of videos required by client users through the online video service system based on HDFS. Figure 6 is a video playback testing screenshot (00:52:35). The experiment shows that the design proposed in this paper is feasible and easy implementable.



Vedio List

Number	OwerUser	Vedio Name	Vedio Size	Process
1	ZHANG SAN	Botanical	233408	Play
2	LI SI	The C Programming Language	233408	Play
3	WANG WU	Network Construct	233408	Play
4	LI MING	ERDAS	233408	Play
5	ZHANG SAN	Computer Application Technology	233408	Play

Figure 5. A playlist of the video service testing.



Minimal Flowplayer setup

View commented source code to get familiar with Flowplayer installation.

can also be used to create this value.

To use one of the cursors predefined in the Microsoft® Win32® API, the application must set the *hInstance* parameter to NULL and the *lpCursorName* parameter to one the following values:

Value	Meaning
IDC_APPSTARTING	Standard arrow and small hourglass
IDC_ARROW	Standard arrow
IDC_CROSS	Crosshair
IDC_HAND	**Windows 2000:** Hand
IDC_HELP	Arrow and question mark
IDC_IBEAM	I-beam
IDC_ICON	Obsolete for applications marked version 4.0 or later.
IDC_NO	Slashed circle
IDC_SIZE	Obsolete for applications marked version 4.0 or later. Use IDC_SIZEALL.
IDC_SIZEALL	Four-pointed arrow pointing north, south, east, and west
IDC_SIZENESW	Double-pointed arrow pointing northeast and southwest
IDC_SIZENS	Double-pointed arrow pointing north and south
IDC_SIZENWSE	Double-pointed arrow pointing northwest and southeast
IDC_SIZEWE	Double-pointed arrow pointing west and east
IDC_UPARROW	Vertical arrow
IDC_WAIT	Hourglass

Figure 6. The video screenshot at time 00:52:35.

4.2.2. Performance Testing

Considering that the video resource service system is frequently used for data reading, the download time ate calculated by Formula (1) was measured in different data sizes (MB) to verify whether the HDFS-based system is fit to manage huge volumes of video resources. The testing result (Figure 7) shows the download time rate of videos is decreasing with data size increasing, that is, the HDFS-based video resource management system performed better in managing big data than small, and videos are already very big in size. When the data size is more than 1000 MB, the download time rate stabilizes at a certain level.

$$P = t_i/t_0 \tag{1}$$

where P represents the download time rate, t_i is the average downloadtime (ms/MB) of the i^{th} data groupB, and t_0 is the download time (ms) of 1 MB.

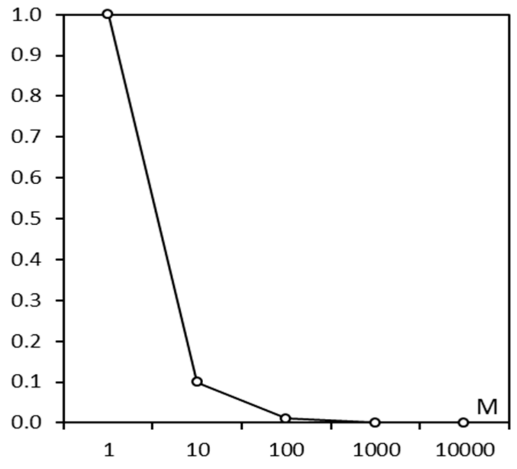


Figure 7. The download time rate of videos decreases as the data size increases.

5. Conclusions

With the volume of video resources is becoming larger and larger and data types are more various and complex, many difficulties have emerged in their management work. Fortunately, Hadoop poses new chances to resolve these problems. This paper proposes architecture to manage video resources based on Hadoop under a HDFS-based cloud-computing environment, and an experiment was done to test the new design. The system is able to ensure uniform video resource management and a high-performance processability by applying HDFS and Map/Reduce; thus, our system overcomes the difficulties related to emerging and merging policies in distributed video resource management service systems as well as to fault tolerance and load balancing management in large-scale distributed systems by obeying Hadoop policies. Based on the experimental results, we also suggest optimal Hadoop options for video resource management service systems in our cloud-based cluster servers. The study shows that using Hadoop to build a distributed storage system for video resources can enhance access speeds and save energy.

This paper is specifically interested in architecture that deals with huge volumes of video resource management tasks, as most of the existing video resource management techniques do not consider parallel computing. The architecture should adapt to different users working on different platforms, and it should have a convenient, friendly, and simple user interface. The requirements of the architecture are portability, being service oriented, flexibility and easy extension. Users can easily use it through a web portal with a standard service-oriented interface. In the study, Map/Reduce could achieve process optimization by distributing tasks among available computing resources. In what follows, we will focus on the dynamic deployment of additional computer resources, as the means to handle seasonal load variations.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (No. 31400493), the Yunnan Provincial Research Foundation for Basic Application Research, China (Grant No. 2011FB070), the Research Center of Kunming Forestry Information Engineering Technology (Grant No. 2015FIB04), the Natural Science Research Foundation of Education Bureau of Yunnan Province, China (Grant No. 2011Y282), and the Teaching Research Foundation of Southwest Forestry University (No.YB201120).

Author Contributions: Weili Kou, Kailai Zhou, and Hui Li designed the concept of the article. Hui Li and Weili Kou did the experiments and created the graphics. Weili Kou wrote the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yang, C.-T.; Huang, K.-L.; Liu, J.-C.; Chen, W.-S. Construction of Cloud IaaS Using KVM and Open Nebula for Video Services. In Proceedings of the 41st International Conference on Parallel Processing Workshops (ICPPW), Pittsburgh, PA, USA, 10–13 September 2012; pp. 212–221.
2. Pereira, R.; Azambuja, M.; Breitman, K; Endler, M. An Architecture for Distributed High Performance Video Processing in the Cloud. In Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, 9–11 July 2010; pp. 482–489.
3. Wu, Y.-S.; Chang, Y.-S.; Juang, T.-Y.; Yen, J.-S. An Architecture for Video Surveillance Service based on P2P and Cloud Computing. In Proceedings of the 9th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2012), Fukuoka, Japan, 4–7 September 2012; pp. 661–666.
4. Garcia, A.; Kalva, H.; Furht, B. A study of transcoding on cloud environments for video content delivery. In Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing, New York, NY, USA, 25–29 October 2010; pp. 13–18.
5. Sun, B.-J.; Wu, K.-J. Research on Cloud Computing Application in the Peer-to-Peer Based Video-on-Demand Systems. In Proceedings of the 3rd International Workshop on Intelligent Systems and Applications (ISA), Wuhan, China, 28–29 May 2011; pp. 1–4.
6. Li, J.; Guo, R.; Zhang, X. Study on Service-Oriented Cloud Conferencing. In Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, 9–11 July 2010; pp. 21–25.
7. Ghemawat, S.; Gobioff, H.; Leung, S.-T. The Google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 19–22 October 2003.
8. Srirama, S.N.; Jakovits, P.; Vainikko, E. Adapting scientific computing problems to clouds using MapReduce. *J. Future Gener. Comput. Syst.* **2012**, *28*, 184–192. [CrossRef]
9. Kim, M.; Cui, Y.; Han, S.; Lee, H. Towards Efficient Design and Implementation of a Hadoop-based Distributed Video Transcoding System in Cloud Computing Environment. *Int. J. Multimed. Ubiquitous Eng.* **2013**, *8*, 213.
10. Web Technologies—Distributed Computing/Big Data 2016. Available online: http://www.bogotobogo.com/WebTechnologies/distributedcomputing.php (accessed on 14 July 2016).
11. Pereira, R.; Azambuja, M.; Breitman, K.; Endler, M. An Architecture for Distributed High Performance Video. In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), Miami, FL, USA, 5–10 July 2010; pp. 482–489.