Towards Efficient Design and Implementation of a Hadoop-based Distributed Video Transcoding System in Cloud Computing Environment

Myoungjin Kim[1], Yun Cui[1], Seungho Han[1] and Hanku Lee[1,2,*]

[1]*Department of Internet and Multimedia Engineering, Konkuk University, Seoul 143-701, Korea*
[2]*Center for Social Media Cloud Computing, Konkuk University, Seoul 143-701, Korea*
{tough105, ilycy, shhan87, hlee}@konkuk.ac.kr
[*]*Corresponding Author: Hanku Lee (hlee@konkuk.ac.kr)*

Abstract

In this paper, we propose a Hadoop-based Distributed Video Transcoding System in a cloud computing environment that transcodes various video codec formats into the MPEG-4 video format. This system provides various types of video content to heterogeneous devices such as smart phones, personal computers, television, and pads. We design and implement the system using the MapReduce framework, which runs on a Hadoop Distributed File System platform, and the media processing library Xuggler. Thus, the encoding time to transcode large amounts of video content is exponentially reduced, facilitating a transcoding function. For performance evaluation, we focus on measuring the total time to transcode a data set into a target data set for three sets of experiments. We also analyze the experimental results, providing optimal Hadoop Distributed File System and MapReduce options suitable for video transcoding. Based on the experiments performed on a 28-node cluster, the proposed distributed video transcoding system provides excellent performance in terms of speed and quality.

Keywords: *Video Transcoding, MapReduce, Hadoop, Cloud Computing, Distributed Processing, Cloud Platform*

1. Introduction

Owing to the explosive growth of mobile devices and improvements in mobile device communication technologies, media streaming services and applications based on video content have gained remarkable popularity and interest from users. Therefore, video transcoding techniques [3, 7, 11] are required for delivering large amounts of video data in multiple formats to heterogeneous mobile devices.

The video transcoding process for sharing video content imposes a heavy burden on the existing Internet infrastructure and computing resources. Recent video files have changed from low capacity and definition to high capacity and definition; hence, massive storage servers are required for the storage of such files. In addition, all video transcoding processes necessarily consist of three intensive sub-processes: decoding, resizing and encoding. Therefore, immense computation power from the CPU resource and efficient approaches for transcoding are needed.

To overcome these limitations, researchers have focused on distributed and cluster-based video transcoding approaches [2, 6, 9, 10] that reduce processing time and maintenance costs of building and expanding computing resources. However, these approaches focus on procuring computing resources for a video transcoding process by simply increasing the number of cluster machines in a parallel and distributed computing environment. In addition, they do not consider load balancing, fault tolerance and a data replication method to ensure data protection and expedite recovery. Furthermore, there has been limited progress in research related to splitting and merging policies that are considered significant in distributed transcoding.

In this paper, we propose a Hadoop-based Distributed Video Transcoding System (HDVTS) in a cloud computing environment that can transcode a variety of video coding formats into the MPEG-4 video format. Improvements in quality and speed are achieved by adopting Hadoop Distributed File System (HDFS) [1] for storing large amounts of video data created by numerous users, MapReduce [1] for distributed and parallel processing of video data, and Xuggler [4] for transcoding based on open source. In addition, our system improves the distributed processing capabilities and simplifies system design and implementation by incorporating data replication, fault tolerance, load balancing, file splitting and merging policies provided by Hadoop. Furthermore, our system provides Infrastructure as a Service (IaaS) aimed at deploying logical computing resources, and offering automatic installation and configuration of HDFS and MapReduce customized for distributed transcoding processing.

In the experiments section, we measured the total transcoding time for three sets of experiments, and experimentally verified the excellent performance of our system in video transcoding processing using distributed techniques. Based on the experimental results, we also provide optimal Hadoop options for video transcoding in our cloud-based cluster server.

The remainder of this paper is organized as follows: Related works are presented in Section 2. In Section 3, the proposed overall system architecture and the implementation strategy are described. Section 4 consists of the details of the experimental environment, including hardware specification, data sets and the experimental method for measurement of video transcoding time. In Section 5, the results of several experiments performed on our cloud cluster are discussed. Section 6 comprises the conclusion and potential future research.

2. Related Work

Owing to the increasing popularity of mobile media services, the development of distributed video transcoding approaches that exponentially reduce transcoding time and ensure video quality has become a challenge. The efforts in the investigation of such approaches have thus increased. In this paper, our system that utilizes HDFS, MapReduce framework, and Xuggler based on a cloud server is presented. Therefore, in this section, we describe HDFS and MapReduce framework.

Hadoop, inspired by Google's MapReduce and Google File System [5], is a software framework that supports data-intensive distributed applications handling thousands of nodes and petabytes of data [1, 8]. It can perform scalable and timely analytical processing of large data sets to extract useful information. Hadoop consists of two important frameworks: 1) Hadoop Distributed File System (HDFS), like GFS, is a distributed, scalable and portable file system written in Java. 2) MapReduce is the first framework developed by Google for processing large data sets.

The MapReduce framework provides a specific programming model and a run-time system for processing and creating large data sets amenable to various real-world tasks [9]. This framework also handles automatic scheduling, communication, and synchronization for processing huge datasets and it has fault tolerance capability. The MapReduce programming model is executed in two main steps called *mapping* and *reducing*. Mapping and reducing are defined by *mapper* and *reducer* functions. Each phase has a list of key and value pairs as input and output. In the *mapping* step, MapReduce receives input data sets and then feeds each data element to the mapper in the form of key and value pairs. In the *reducing* step, all the outputs from the *mapper* are processed, and the final result is generated by the *reducer* using the merging process. Figure 1 shows the example of word counting using MapReduce framework.

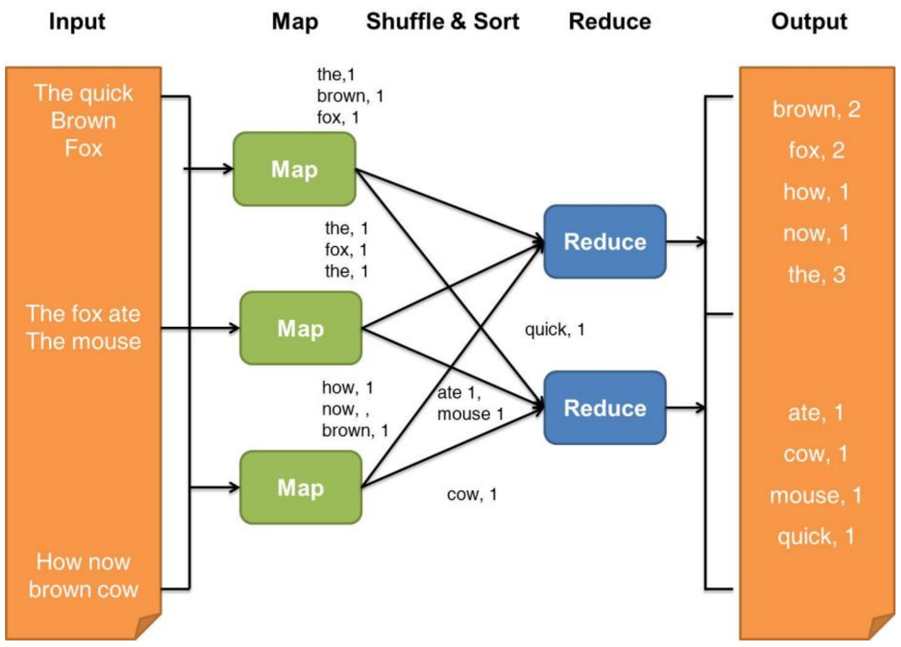


Figure 1. An Example of Word Counting using MapReduce

3. Proposed System Architecture

3.1 Characteristics of HDVTS

In this section, we describe our proposed system architecture. The characteristics of our HDVTS are as follows: (1) It consists of a codec transcoding function and a function with a configurable display size, codec method, and container format. (2) It focuses mainly on the batch processing of large video files collected over a fixed period of time rather than the processing of small video files collected in real time. (3) HDFS is used in our system in order to avoid the high cost of communication of the video file during data transfer for distributed processing. HDFS is also used owing to the large chunk size (64 MB) policy suitable for processing video files, and the user-level distributed system. (4) Our system incorporates load balancing, fault tolerance, and merging and splitting policies provided by MapReduce for distributed processing.

3.2 Overall System Architecture

HDVTS is mainly divided into four domains: Video Data Collection Domain (VDCD), HDFS-based Splitting and Merging Domain (HbSMD), MapReduce-based Transcoding

Domain (MbTD) and Cloud-based Infrastructure Service Domain (CbISD). Figure 2 shows the system architecture.

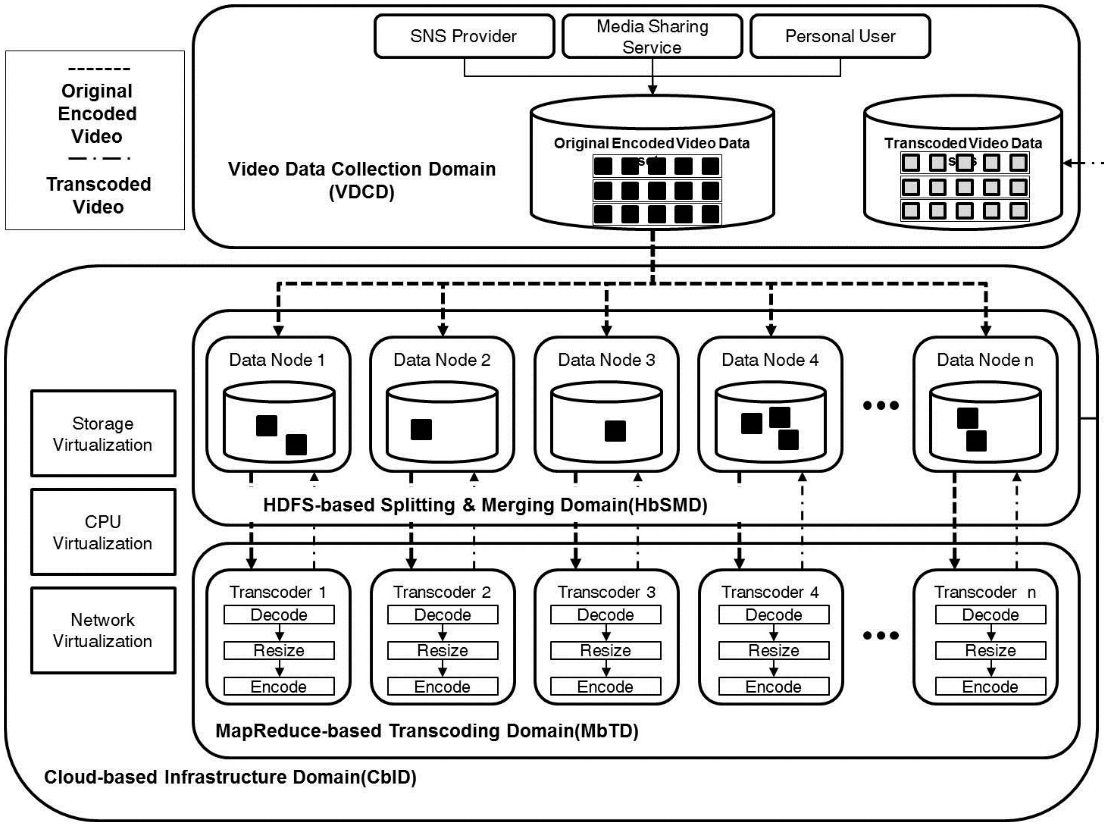


Figure 2. The Overall Architecture of HDVTS

First, the main contribution of VDCD is the collection of different types of original encoded video files created by media creators such as SNS providers, media sharing services, and personal users, and the storage of these files on our local file system. It also collects transcoded video data sets converted to a target format file through a transcoding processing step based on MapReduce in MbTD, and stores them on the local file system. The period for collecting original encoded video data sets can be set by administrators and users according to a data set size and acquisition time.

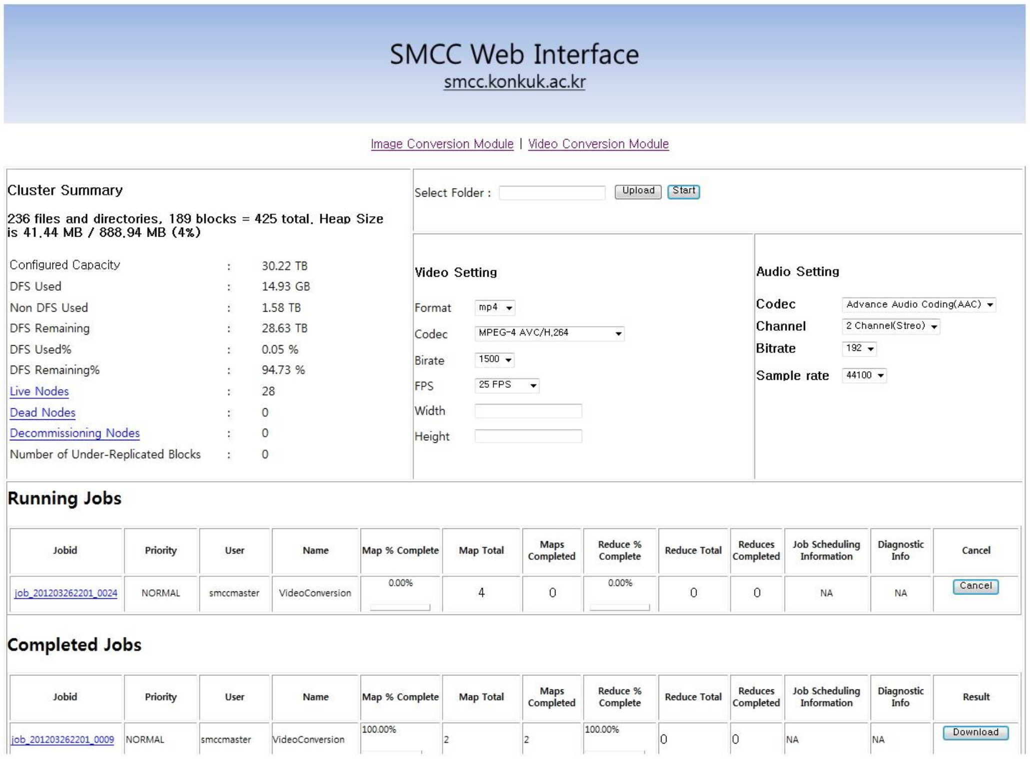
Second, the main role of HbSMD, which runs on HDFS, is to split collected original video data sets into blocks of a configured size, and to automatically distribute all blocks over the cluster. In HbSMD, the default block size is set to 64 MB, but it is changed by administrators and users to various other values, such as 16 MB, 32 MB, 128 MB, 256 MB, etc. When a block is distributed, it is replicated at three data nodes according to the Hadoop distribution policy, thus complying with the entire distributed processing procedure and enabling recovery from a system failure caused by data loss. The other role of HbSMD is to merge blocks transcoded by transcoders in MbTD into target video files, and to transmit the video files to VDCD. The number of block replicas is set to 1, 2, 4, 5, *etc.*

Third, MbTD performs several tasks that transcode distributed blocks in each data node by using a MapReduce-based transcoding module with Xuggler. A data node 1 and a transcoder 1 are located in the same physical machine. First, the transcoders implement the decoding step. Next, the resizing step is implemented if the users and administrators require a change in the resolution of a video file. If such a change is not required, the transcoders skip this step.

Finally, the transcoders encode the decoded blocks into a target file based on the requirements of the user. The details will be presented in Section 3.3. Last, CbISD offers infrastructure services in a cloud computing environment via server, storage, CPU, and network virtualization techniques. Because of the massive storage space and enormous computing resource requirements of such systems, small service vendors are unable to afford the cost of building them. When users require logical computing resources to build and implement this system, CbISD automatically deploys a virtualized cluster environment. CbISD allows users to select a specific configuration of memory, CPU, storage, and the number of clusters. In addition, it provides the easy installation and configuration environment of HDFS and MapReduce without much effort from the user. In this paper, we present the idea and concept of CbISD; its implementation is not considered.

3.3 Prototype of HDVTS

Figure 3 shows our first prototype of an HDVTS. In this prototype, users and administrators can select video transcoding options such as format, codec, bitrate, width, and height, and audio transcoding options such as codec, bitrate, and sample rate. Further, the summary information of the system including the available storage capacity of HDFS, the activation state of data nodes, and the progress status report for MapReduce job are monitored.



SMCC Web Interface
smcc.konkuk.ac.kr

Image Conversion Module | Video Conversion Module

Cluster Summary

236 files and directories, 189 blocks = 425 total. Heap Size is 41.44 MB / 888.94 MB (4%)

Configured Capacity	:	30.22 TB
DFS Used	:	14.93 GB
Non DFS Used	:	1.58 TB
DFS Remaining	:	28.63 TB
DFS Used%	:	0.05 %
DFS Remaining%	:	94.73 %
Live Nodes	:	28
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

Select Folder : Upload Start

Video Setting

Format	mp4
Codec	MPEG-4 AVC/H.264
Birate	1500
FPS	25 FPS
Width	
Height	

Audio Setting

Codec	Advance Audio Coding(AAC)
Channel	2 Channel(Streo)
Bitrate	192
Sample rate	44100

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info	Cancel
job_201203262201_0024	NORMAL	smccmaster	VideoConversion	0.00%	4	0	0.00%	0	0	NA	NA	Cancel

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info	Result
job_201203262201_0009	NORMAL	smccmaster	VideoConversion	100.00%	2	2	100.00%	0	0	NA	NA	Download

Figure 3. Prototype of a Hadoop-based Distributed Video Transcoding System

4. Experimental Environment

In this section, we provide the hardware specification of our cloud cluster, data sets and the experimental methods for measuring video transcoding time. Performance evaluation is conducted on a 28-node HDFS cluster consisting of 1 master node and 27 slave nodes (data nodes). Each node running on the Linux OS (CentOS 5.5) is equipped with two Intel Xeon 4 core 2.13 GHz processors with 4 GB registered ECC DDR memory and 1 TB SATA-2. All nodes are interconnected by a 100 Mbps Ethernet adapter. We also use Java 1.6.0_23, Hadoop-0.20.2, and Xuggler 3.4.

In addition, to verify the performance evaluation for encoding very large sizes of video files into target files, we create and use six types of video data sets (Table 1), including several 200 MB original video files, according to their volume sizes. Table 2 lists the parameters for each original and target transcoded video file.

Table 1. Video Data Sets for Performance Evaluation

Size of file	1GB	2GB	4GB	8GB	10GB	50GB
Number of video files	5	10	20	40	50	250

Table 2. Parameters for Each Original and Transcoded Video File

Parameter	Original video file	Transcoded video file
Codec	XviD	MPEG-4
Container	AVI	MP4
Size	200 MB	60MB
Duration	3 min 19 s	3 min 19 s
Resolution	1280 ×720	320 ×240
Frame rate	29.97 fps	29.97 fps

5. Performance Evaluation

In this section, we discuss and analyze the experimental results, presenting optimal Hadoop options for processing video files. We focus on measuring the total time to transcode the original data set (Table 1) into the target data set (Table 1) for three sets of experiments. 1) Change in cluster size for speedup performance 2) Different MapReduce options for block size (default: 64 MB) 3) Different Hadoop options for block replication factor (default: 3).

5.1 Changing Cluster Size for Speedup Performance

The objective of the first set of experiments is to measure the total transcoding time and speedup for various cluster sizes, such as 1, 4, 8, 12, 16, 20, 24 and 28 nodes with

Hadoop default options. Speedup is used to evaluate the effect of parallelism. It is defined as: Speedup (n) = transcoding time on 1 node / transcoding time on n nodes.

Table 3 lists the total transcoding time for various cluster sizes. Table 4 and Figure 4 show the calculated speedup results.

Table 3. Total Transcoding Time for Various Cluster Sizes (s)

Nodes	Video Data Set Size					
	1GB	2GB	4GB	8GB	10GB	50GB
1	916	1712	3465	6934	8727	43183
4	278	516	950	1810	2268	11040
8	183	276	512	967	1188	5615
12	170	196	364	677	793	3747
16	166	188	282	517	620	2824
20	170	171	247	412	498	2240
24	161	168	203	363	421	1841
28	124	169	207	311	375	1623

Table 4. Speedup Results for Various Cluster Sizes

Nodes	Video Data Set Size					
	1GB	2GB	4GB	8GB	10GB	50GB
1	1	1	1	1	1	1
4	3.29	3.32	3.65	3.83	3.85	3.91
8	5.01	6.20	6.77	7.17	7.35	7.69
12	5.39	8.73	9.52	10.25	11.01	11.52
16	5.52	9.11	12.29	13.41	14.08	15.29
20	5.39	10.01	14.03	16.83	17.52	19.28
24	5.69	10.19	17.07	19.10	20.73	23.46
28	7.39	10.13	16.74	22.30	23.27	26.61

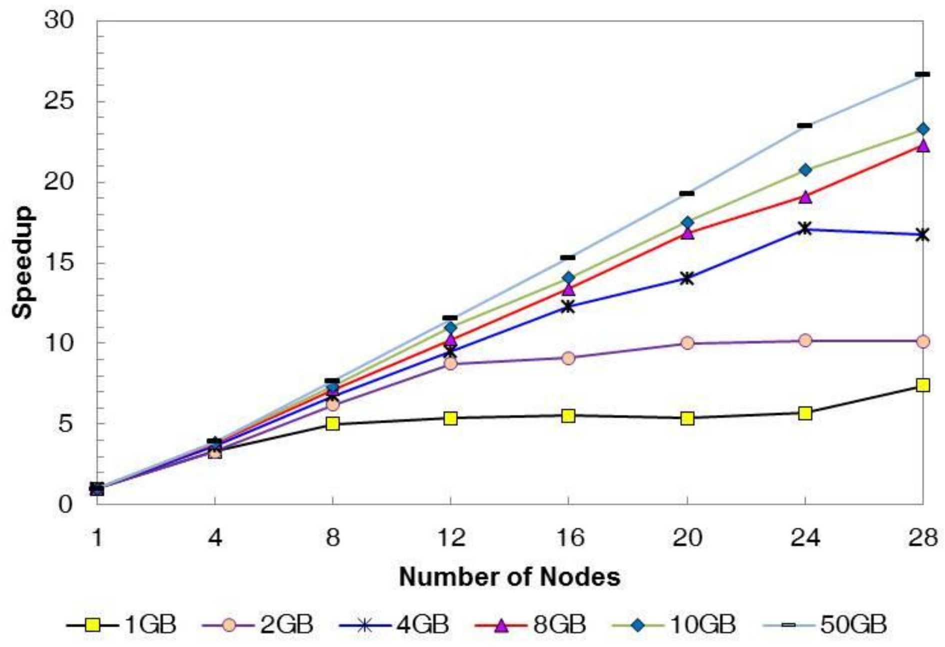


Figure 4. Speedup Results for Various Cluster Sizes

According to Table 3, our system provides excellent transcoding time for very large sizes of video files. For example, our system takes approximately 1600 s (about 27 min) to complete the transcoding process for a 50 GB video data set with the default Hadoop options.

From the speedup results, it can be observed that: 1) Our system has excellent performance in terms of its parallel and distributed characteristic. 2) Speedup performances for 10 and 50 GB data sets are better compared to the speedup performances for 1, 2, 4, 8 GB datasets, implying that our system exhibits good performance when the size of the data set increases.

5.2 Changing Block Size and Block Replication Factor

In the second and third sets of experiments, we measure the total time to transcode each data set with the different MapReduce options for block size (default: 64 MB) and block replication factor (default: 3). 5 block size options, 32, 64, 128, 256 and 512 MB are used in the experiments. 5 values of block replication factor, 1, 2, 3, 4 and 5 are used. Table 5 and Figure 5 show the measured transcoding times in seconds for different block size. Table 6 and Figure 6 show the total transcoding times in seconds for different replication factor values.

In this experiment, it is clearly observed that our system performs best when the block size is set to 256 MB and 512 MB, or when the block replication factor is set to three. Hence, it can be concluded that to ensure the best distributed video transcoding performance in our system, the block size option should be set to a value closer to the original file size. Furthermore, the block replication factor should be set to three. This value provides the best performance and makes the distributed systems that process massive media files reliable and robust in terms of recovery from system failure when data loss occurs and one node fails.

Table 5. Total Transcoding Time for Various Values of Block Size (s)

Block Size	Video Data Set Size					
	1GB	2GB	4GB	8GB	10GB	50GB
32 MB	173	226	335	532	666	2837
64MB	124	169	207	311	375	1623
128MB	103	108	120	199	209	820
256MB	102	103	106	106	116	443
512MB	102	105	105	111	109	444

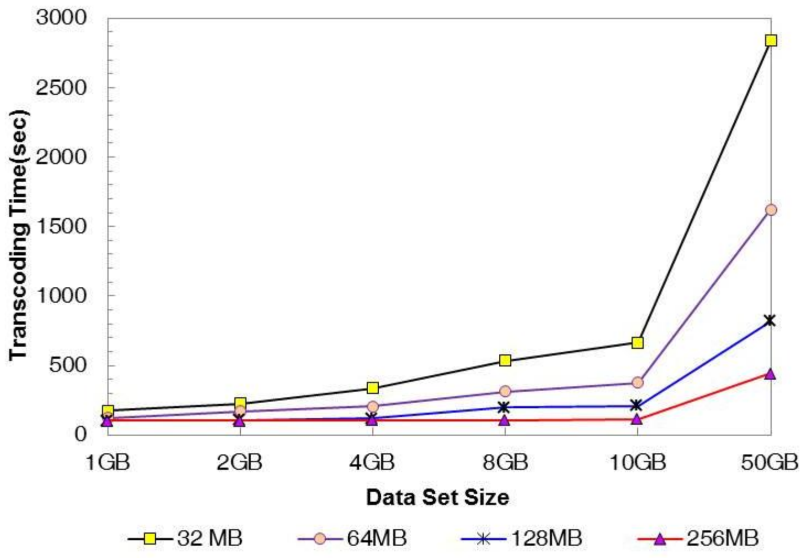


Figure 5. Total Transcoding Time According to Changes in the Block Size

Table 6. Total Transcoding Time for Various Values of Block Replication Factor (s)

Block Replication	Video Data Set Size					
	1GB	2GB	4GB	8GB	10GB	50GB
32 MB	173	226	335	532	666	2837
64MB	124	169	207	311	375	1623
128MB	103	108	120	199	209	820
256MB	102	103	106	106	116	443
512MB	102	105	105	111	109	444

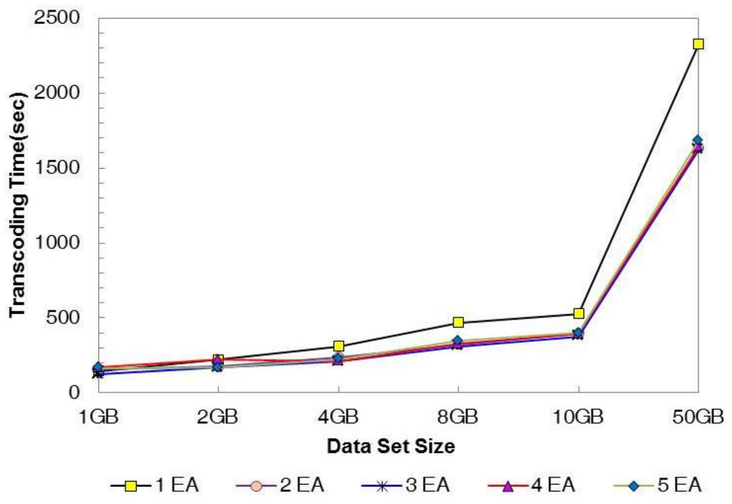


Figure 6. Total Transcoding Time According to Changes in the Block Replication

6. Conclusion and Future Work

We proposed a Hadoop-based Distributed Video Transcoding System (HDVTS) in a cloud computing environment that transcodes various video codec formats into the MPEG-4 video format. Our system ensures uniform transcoded video quality and a fast transcoding process by applying HDFS and MapReduce, the core techniques in cloud computing enabling technologies. Moreover, our system overcomes the difficulties related to emerging & merging policies in distributed video processing as well as fault tolerance and load balancing management in large-scale distributed systems by obeying Hadoop policies. In the experiments section, we measured the total transcoding time for three sets of experiments, and experimentally verified the excellent performance of our system in video transcoding processing using distributed techniques. Based on the experimental results, we also suggest optimal Hadoop options for video transcoding in our cloud-based cluster server.

We will leverage this HDVTS to implement Cloud-based Infrastructure Service Domain (CbISD) and to improve strategies for load balancing, advanced splitting & merging policies for media processing, and quality of service for delivering mobile media service. We also plan to add distributed video streaming services optimized for the cloud computing environment to our system.

Acknowledgements

This work was supported by the Konkuk University

References

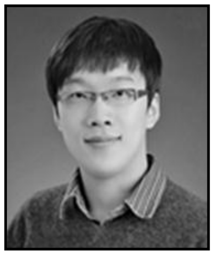
[1] Wikipedia, http://en.wikipedia.org/wiki/Apache_Hadoop, **(2012)**.
[2] Z. Tian, J. Xue, W. Hu, T. Xu and N. Zheng, “High performance cluster-based transcoder”, Proceedings of 2010 International Conference on Computer Application and System Modeling, **(2010)** October 22-24; Shanxi, China.

[3] I. Ahmad, X. Wei, Y. Sun and Y.-Q. Zhang, IEEE Transactions on Multimedia, vol. 7, no. 5, **(2005)**.
[4] Xuggler Java library, http://www.xuggle.com/xuggler/.
[5] S. Ghemawat, H. Gobioff and S. -T. Leung, “The Google file system”, Proceedings of the nineteenth ACM symposium on Operating Systems principles, **(2003)** December, New York, USA.
[6] Y. Sambe, S. Watanabe, D. Yu, T. Nakamura and N. Wakamiya, IEICE Transactions on Information and Systems, vol. E88-D, no. 8, **(2005)**.
[7] S. Moiron, M. Ghanbari, P. Assuncao and S. Faria, Studies in Computational Intelligence, vol. 231, **(2009)**.
[8] S. N. Srirama, P. Jakovits and E. Vainikko, Journal of Future Generation Computer Systems, vol. 28, no. 1, **(2012)**.
[9] D. Seo, J. Kim and I. Jung, “Load distribution algorithm based on transcoding time estimation for distributed transcoding servers”, Proceedings of 2010 International Conference in Information Science and Applications, **(2010)** April 21-23, Seoul, Korea.
[10] H. Sanson, L. Loyola and D. Pereira, “Scalable distributed architecture for media transcoding”, Proceedings of 12th International Conference on Algorithms and Architectures for Parallel Processing, **(2012)** September 4-7, Fukuoka, Japan.
[11] C. Lee, Journal of Supercomputing, **(2012)**, pp. 1-20.

Authors

Myoungjin Kim

Myoungjin Kim received B.S. degree in computer science from Daejin University in 2007 and M.S. degree from Konkuk University, Seoul, Korea, in 2009. Currently, He is a Ph.D. student in the department of Internet and Multimedia Engineering at the same university and also assistant researcher at the Social Media Cloud Computing Research Center. His research interest includes distributed computing, real-time programming, MapReduce, Hadoop, Media transcoding and cloud computing



Yun Cui

Yun Cui received M.S degree in the division of Internet and Multimedia Engineering at Konkuk University, Korea and currently he is Ph.D. student His current research interests are cloud computing, social network service, home network service and distributed computing systems.

Seungho Han

Seungho Han is a M.S course student in the department of Internet and Multimedia Engineering at University of Konkuk. He is also and also assistant researcher at the Social Media Cloud Computing Research Center. He is current research interests are UPnP, cloud computing system, Hadoop.

Hanku Lee

Hanku Lee is the director of the Social Media Cloud Computing Research Center and an associate professor of the division of Internet and Multimedia Engineering at Konkuk University, Seoul, Korea. He received his Ph.D. degree in computer science at the Florida State University, USA. His recent research interests are in cloud computing, distributed real-time systems, distributed and compilers.