A Hadoop-based Multimedia Transcoding System for Processing Social Media in the PaaS Platform of SMCCSE

Myoungjin Kim[1], Seungho Han[1], Yun Cui[1], Hanku Lee[1,*] and Changsung Jeong[2]
[1] Department of Internet and Multimedia Engineering, Konkuk University
Gwangjin-gu, Seoul 143-701 – Republic of Korea
[2] Department of Electrical Engineering, Korea University
Seongbuk-gu, Seoul 136-713 – Republic of Korea
[e-mail: tough105, shhan87, ilycy, hlee@konkuk.ac.kr, csjeong@korea.ac.kr]
*Corresponding author: Hanku Lee

Received April 9, 2012; revised July 9, 2012; accepted August 16, 2012; published November 30, 2012

Abstract

Previously, we described a social media cloud computing service environment (SMCCSE). This SMCCSE supports the development of social networking services (SNSs) that include audio, image, and video formats. A social media cloud computing PaaS platform, a core component in a SMCCSE, processes large amounts of social media in a parallel and distributed manner for supporting a reliable SNS. Here, we propose a Hadoop-based multimedia system for image and video transcoding processing, necessary functions of our PaaS platform. Our system consists of two modules, including an image transcoding module and a video transcoding module. We also design and implement the system by using a MapReduce framework running on a Hadoop Distributed File System (HDFS) and the media processing libraries Xuggler and JAI. In this way, our system exponentially reduces the encoding time for transcoding large amounts of image and video files into specific formats depending on user-requested options (such as resolution, bit rate, and frame rate). In order to evaluate system performance, we measure the total image and video transcoding time for image and video data sets, respectively, under various experimental conditions. In addition, we compare the video transcoding performance of our cloud-based approach with that of the traditional frame-level parallel processing-based approach. Based on experiments performed on a 28-node cluster, the proposed Hadoop-based multimedia transcoding system delivers excellent speed and quality.

Keywords: Hadoop, mapreduce, multimedia transcoding, cloud computing, paas

A preliminary version of this paper appeared in the ICONI (International Conference on Internet) 2011, December 15-19, 2011. This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) [NIPA-2012-H0301-12-3006].

http://dx.doi.org/10.3837/tiis.2012.10.005

1. Introduction

In these days of rapid technological change, cloud computing [20][22][24][31][32] has achieved remarkable interest from researchers and the IT industry for providing a flexible dynamic IT infrastructure, QoS guaranteed computing environments, and configurable software services [11]. Due to these advantages, many service providers who release Social Network Services (SNS) [8][12][14] utilize cloud computing in order to reduce maintenance costs of building and expanding computing resources for processing large amounts of social media data, such as video, image, and text formats.

In order to develop a SNS based on large amounts of social media, scalable mass storage for social media data created daily by users is needed. For example, the amount of data generated by Twitter every day reaches up to 7TB. Facebook also produces around 10TB because media files have recently changed from low capacity, low definition to high capacity and high definition formats. In addition to transferring social media data to end-users, media transcoding approaches [3][7][9] are required for delivering a variety of video data in multiple formats to heterogeneous mobile devices. Moreover, distributed and parallel data processing models such as Hadoop [2][13], Google's MapReduce model [1][21] and the Message Passing Interface (MPI) standard [4] are also needed for data processing in a parallel and distributed computing environment.

In an earlier publication [5], we described the new concept of a Social Media Cloud Computing Service Environment (SMCCSE) that enables cloud computing technologies, approaches in the intensive use of computing resources, and cloud services for developing social media-based SNS. In particular, we focused on designing a social media PaaS platform as the core platform of our SMCCSE in [17]. The main role of the social media PaaS platform is to provide a distributed and parallel processing system for media transcoding functions and delivering social media, including video and audio files, to heterogeneous devices such as smart phones, personal computers, televisions, and smart pads. This platform is composed of three parts: A social media data analysis platform for large scalable data analysis; a cloud distributed and parallel data processing platform for storing, distributing, and processing social media data; and finally, a cloud infra management platform for managing and monitoring computing resources.

In this paper, we focus on designing and implementing a Hadoop-based multimedia transcoding system for delivering image and video files in a SNS by adopting a social media PaaS platform in SMCCSE. Our transcoding system consists of two modules, including a video transcoding module for converting video data and an image transcoding module for converting image data. Our video transcoding module can transcode a variety of video coding formats into the MPEG-4 video format. The image transcoding module can transcode large amounts of image data sets into a specific format.

In the traditional multimedia transcoding approaches, many researchers have focused on distributed and cluster-based video media approaches, such as those found in [6][15][28] that reduce processing time and maintenance costs for building a computing resource infrastructure. 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 order to overcome these limitations, we apply a cloud computing environment to our Hadoop-based multimedia transcoding system. Improvements in quality and speed are achieved by adopting Hadoop Distributed File System (HDFS) [18][29] for storing large amounts of video data created by numerous users, MapReduce [10] for distributed and parallel processing of video data, and Xuggler [25] and Java Advanced Imaging (JAI) [27] for media 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.

Our paper is organized as follows: in Section 2 we describe the basic idea of cloud computing, HDFS, MapReduce, and media transcoding approaches. In Section 3, we introduce the PaaS platform in SMCCSE, describing three sub-platforms in detail. In Section 4, we propose a Hadoop-based media transcoding system in the PaaS platform. Design and implementation strategies of our system are provided in Section 5. In Section 6, we discuss the results of several experiments conducted on our cloud cluster by presenting optimal Hadoop options suitable for media transcoding; in addition, we compare the transcoding performance of our Hadoop-based transcoding approach implemented in Java with that of the traditional frame-based parallel transcoding approach implemented in C and C++. Section 7 comprises the conclusion and potential future research.

2. Related Work

2.1 Hadoop and MapReduce

Hadoop, inspired by Google's MapReduce and Google File System [33], is a software framework that supports data-intensive distributed applications handling thousands of nodes and petabytes of data. 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 [30]. 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.

2.2 Media transcoding

The term media transcoding is defined in many publications, such as [23][29]. In [11], to bring multimedia contents and service to the numerous heterogeneous client devices while retaining the ability to go mobile, multimedia information must be adapted, that is referred to as media transcoding technology.

Fig. 1 illustrates the architecture of a legacy transcoding system. First, the client requests a transcoding function from a transcoding server. The transcoding sever reads the original media data from the media server, proceeds to transcode the data depending on user requested resolution, bit-rate, and frame rate. The transcoding server then sends the transcoded media data to the client [30]. However, this media transcoding processing imposes a heavy burden on the existing internet infrastructure and computing resources because more recent media files, such as video and image files have changed to high capacity/high definition.

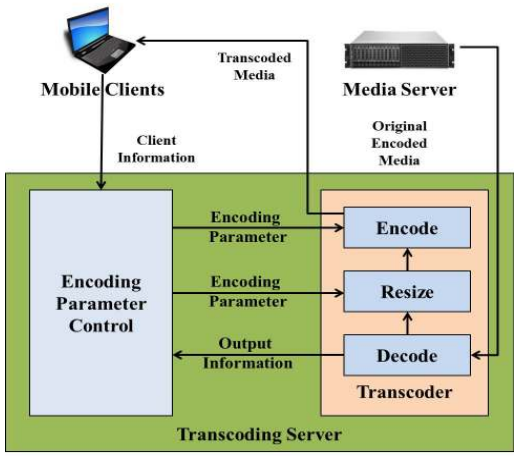


Fig. 1. The architecture of a legacy transcoding system [30]

Therefore, many researchers apply distributed and parallel computing to media transcoding methods. Jiani Guo et al. in [15] have proposed a cluster-based multimedia web server. This team has designed and implemented a media cluster that dynamically generates video units in order to satisfy the bit rate requested by many clients and has proposed seven load balance scheduling schemes for the MPEG transcoding service. Y. Sambe et al. in [16] have designed and implemented a distributed video transcoding system able to transcode a MPEG-2 video file into diverse video formats with different rates. The main idea behind transcoding a video file is that the transcoder chunks the MPEG-2 video file into small segments along the time axis and transcodes them in a parallel and distributed manner.

Zhiqiang et al. in [19] described a cluster-based transcoder that can transcode MPEG-2 format video files into MPEG-4 and H.264 format video files with faster transcoding speed. This system is composed of a master node and a number of worker nodes. The master node consists of six threads, a splitter, merger, sender, receiver, scheduler and an audio transcoder.

3. Brief Overview on PaaS Platform of SMCCSE

In this section, we briefly review the Social Media Cloud Computing Service Environment (SMCCSE), focusing on describing the social media PaaS platform.

3.1 SMCCSE (Social Media Cloud Computing Service Environment)

Our SMCCSE has a multiple service model using cloud computing to support SNSs such as Twitter and Facebook, social media services such as YouTube, and social game services like the social network games in Facebook.

First, our service model offers social media APIs, a web-based social SDK, and service delivery platforms for developing the SNS as the form of SaaS. Second, in order to provide social media data with reliable services to users, this service model also provides a distributed and parallel data processing platform for storing, distributing, and en/decoding large amounts of social data (including audio, video, and image formats) as a form of PaaS. Finally, the service model provides an IaaS based on virtualization in order to reduce the cost associated with building computing resources, such as servers, storage, and so on. **Fig. 2** summarizes this Social Media Cloud Computing service model.

Here we introduce the SMCCSE architecture. Designing a SMCCSE involves establishing an environment for supporting the development of SNS, addressing numerous SNSs, providing approaches for processing large amounts of social media data, and providing a set of mechanisms to manage the entire infrastructure.

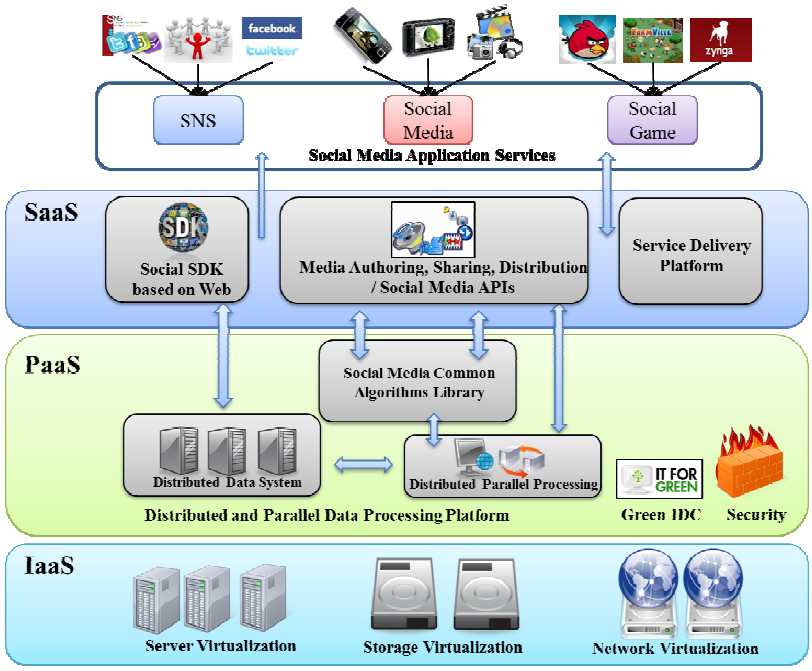


Fig. 2. Social Media Cloud Computing Service model

3.2 PaaS Platform in SMCCSE

The PaaS platform is likely to be the core platform of SMCCSE and IaaS, and provides the physical computing environment. **Fig. 3** shows the whole architecture of the PaaS platform in SMCCSE.

3.2.1 Social Media Data Analysis Platform

The role of a social media data analysis platform is to analyze social media data including text, images, audio, and videos and to provide various libraries that perform the functions of encoding, decoding, transcoding, and transmoding these different formats. In social media based SNSs, the analysis social media data is one of the most important elements for offering reliable services to users. In order to recommend and offer social media of specific types to users, our platform analyzes usage patterns, types, and correlation of social media shared, created, and published by users in advance. The other key function of a social media data

analysis platform is to provide a user friendly interface that conducts the functions of transcoding and transmoding so that users can easily create, share, and upload social media, especially image and video content via social media common algorithms libraries.

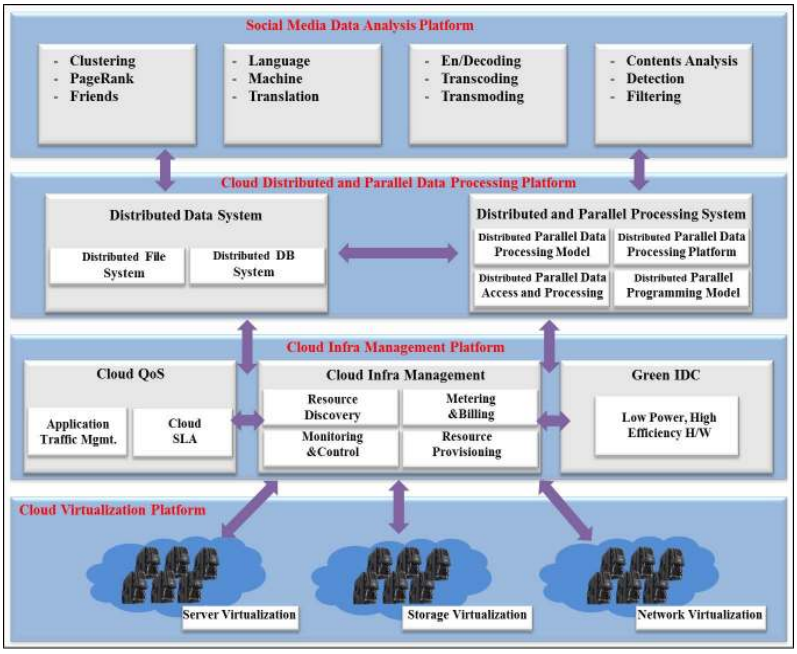


Fig. 3. Social Media Cloud Computing PaaS Platform

3.2.2 Cloud Distributed and Parallel Data Processing Platform

The main function of a cloud distributed and parallel data processing platform as a core platform in SMCCSE is to store, distribute, and process large amounts of social media data. The social media data are then transferred to user devices, such as mobile phones, smart pads, PCs, TVs, etc. Distributed and parallel data processing system are composed of two systems: a distributed data system and a distributed parallel processing system. The distributed data system adopts HDFS for a distributed file system and HBase (Hadoop Database) for a distributed DB system. In addition, we also select MapReduce as a distributed parallel programming model.

In practice, this platform carries out functions provided by social media common algorithms libraries in social media data analysis platforms. First, newly created social media data (text, images, audio, and video) are stored on HDFS. Stored data is processed in two steps using MapReduce. In the first step, our platform conducts analysis work for the execution of each core logic defined by social media APIs in the SaaS platform. For instance, if the SaaS platform defines a social media API that shows a list of video clips a particular group of users have seen most, MapReduce analyzes the social media data and returns any result to the social media API in order to provide the list to requestors. In the second step, encoding, decoding, transcoding, and transmoding functions are carried out to serve the QoS service to hundreds of heterogeneous smart devices.

Traditional approaches to media conversion are very time-intensive. However, our platform has reduced the media conversion time by using enabling cloud computing technologies and

large scalable computing resources in a cloud computing environment. Using the fixed size policy [26][34], a traditional file splitting technique, the content of a single image is split into small chunks and stored in HDFS. Each chunk is then encoded in parallel and subsequently combined into a single file again using the MapReduce programming model. MapReduce reduces run time for encoding work. The transcoding and transmoding functions are carried out using the same approach.

3.2.3 Cloud Infra Management Platform

The Cloud Infra Management Platform consists of cloud QoS, Green IDC, and Cloud Infra Management. Cloud Infra management manages and monitors computing resources that do not depend on a specific OS or platform. Cloud Infra management includes resource scheduling, resource information management, resource monitoring, and virtual machine management functions. These functions are provided on a web service based on Eucalyptus.

4. Hadoop-based Multimedia Transcoding System in PaaS of SMCCSE

Media transcoding is a very important function in the PaaS platform of SMCCSE for delivering social media content. Hence, we design and implement a Hadoop-based multimedia transcoding system, including image and video transcoding modules by utilizing the PaaS platform scheme of SMCCSE.

4.1 Overall System Architecture

The core processing for video and image transcoding is briefly explained as follows: The proposed system uses HDFS as storage for distributed parallel processing. The extremely large amount of collected data is automatically distributed in the data nodes of HDFS. For distributed parallel processing, the proposed system exploits the Hadoop MapReduce framework. In addition, Xuggler libraries for video resizing and encoding as well as JAI libraries for image are utilized in Mapper. The Map function processes each chunk of video data in a distributed and parallel manner. **Fig. 4** illustrates the overall architecture of a Hadoop-based multimedia transcoding system in PaaS platform.

4.2 Hadoop-based Multimedia Transcoding System Architectural Components

Our system is mainly divided into four domains: Video and Image Data Collection Domain (VIDCD), HDFS-based Splitting and Merging Domain (HbSMD), MapReduce-based Transcoding Domain (MbTD) and Cloud-based Infrastructure Domain (CbID).

4.2.1 VIDCD

The main contribution of VIDCD is the collection of different types of original encoded video and image 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 and image data sets converted to target format files 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 and image data sets can be set by administrators and users according to a data set size and acquisition time.

4.2.2 HbSMD

The main role of HbSMD, which runs on HDFS, is to split collected original video and image 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 and image files, and to transmit them to VIDCD. The number of block replicas is set to 1, 2, 4, 5, etc.

4.2.3 MbTD

MbTD performs several tasks that transcode distributed blocks in each data node by using a MapReduce-based transcoding module with Xuggler and JAI. 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 and an image 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.

4.2.4 CbID

CbID 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, CbID automatically deploys a virtualized cluster environment. CbID 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 CbID; its implementation is not considered.

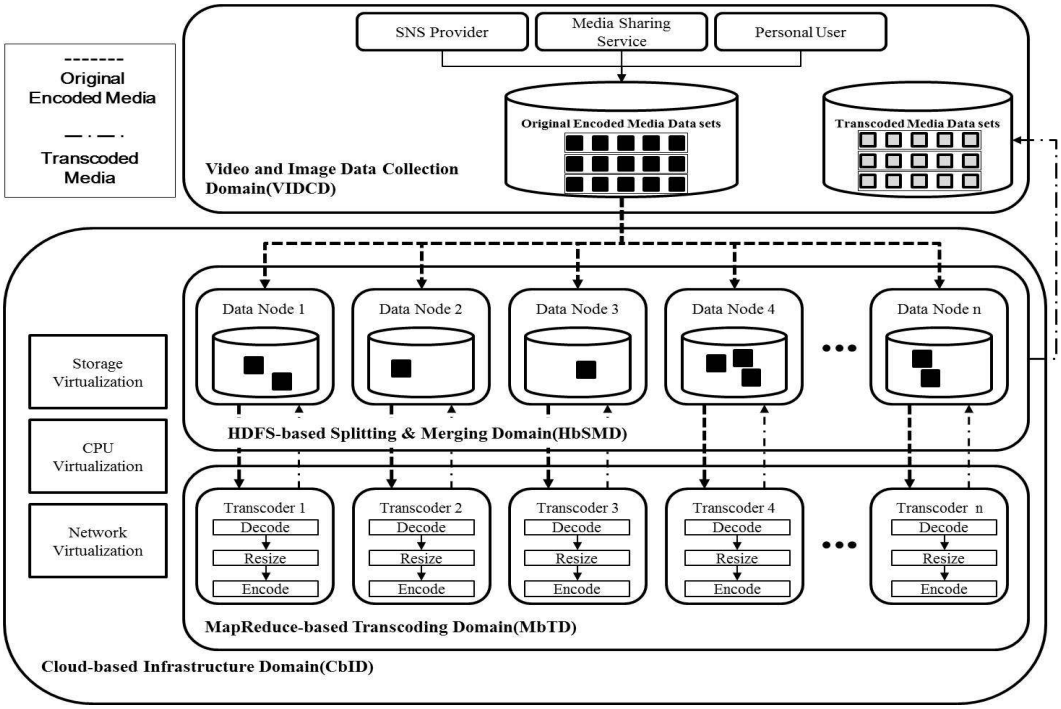


Fig. 4. Overall Architecture of a Hadoop-based Multimedia Transcoding System in the PaaS platform

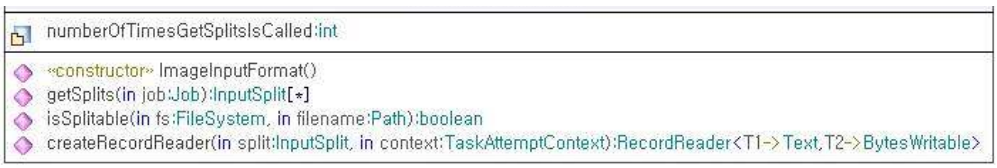
5. Design and Implementation

In this section, we discuss several design features of our Hadoop-based multimedia transcoding system. We describe the design and implementation of MbTD, which is responsible for image and video data processing through the MapReduce framework. The image and video transcoding modules are also designed based on the MapReduce framework. The MapReduce framework provides the `FileInputFormat` and `FileOutputFormat` classes for processing petabyte-scale text data in a parallel and distributed manner. However, these classes are not suitable for processing media data. Therefore, we designed new classes: `ImageInputFormat`, `VideoFileInputFormat`, `ImageOutputFormat`, and `VideoFileOutputFormat` suitable for performing image and video transcoding functions within the MapReduce framework. We also discuss implementation issues for MbTD based on Xuggler and JAI, as described in the previous section.

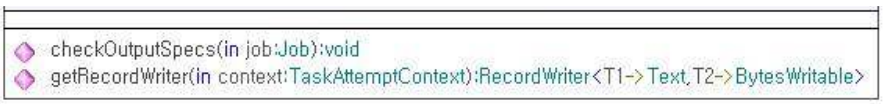
5.1 Design Strategy

We first discuss the design strategy of our image transcoding module. `FileInputFormat` is responsible for transferring stored data in the HDFS to *mapper*. The module is designed to read text data line by line. However, since our system deals with media data like images and video data, rather than text data, we design a new `ImageInputFormat` class that can read image data by transforming such data to byte stream form. In addition, the `ImageRecordReader` class is designed to read one line record transformed as a byte stream form from `ImageInputFormat` and pass it to *mapper*.

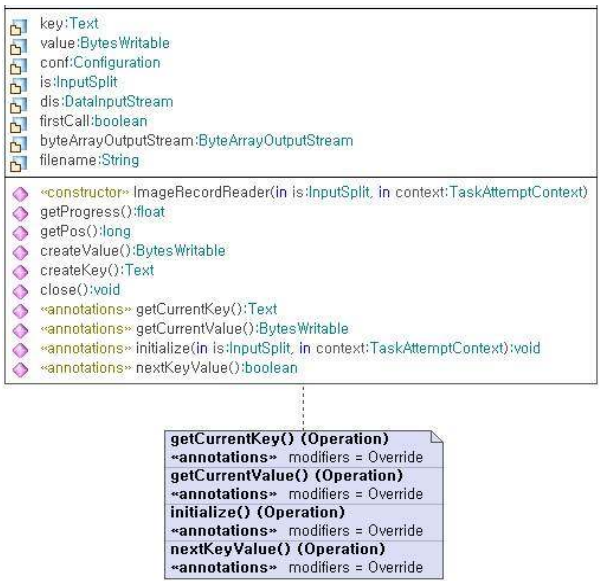
When one line record transfers to *mapper*, it is composed of a video file name and a byte stream of the image file as key and value pairs. For `FileOutputFormat`, the record is in the same state as for `FileInputFormat` explained above. Hence, we also design new `ImageOutputFormat` and `ImageRecordWriter` classes that receive key and value pairs in record form created as a result of *mapper* and *reducer*. Subsequently, these classes output the record in a specified directory. **Fig. 5** illustrates four class diagrams for image conversion function. Although, most MapReduce applications bring a result via *mapper* and/or *reducer*, we only implement *mapper* in our system since it is unnecessary to reduce a set of intermediate values using *reducer*. Moreover, the class `ImageResizer` using the JAI libraries is designed to perform a transcoding function by processing key and value pairs in the *mapper* phase. **Fig. 6** shows `ImageConversion` with `main ( )`, `Map`, and `ImageResizer` class diagrams.



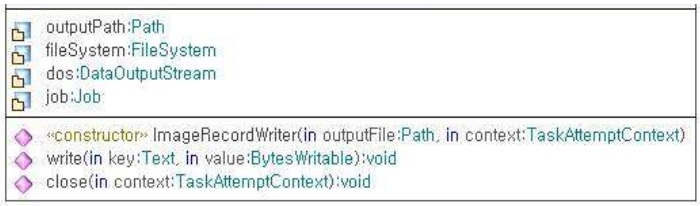
`ImageInputFormat` class diagram



`ImageOutputFormat` class diagram



ImageRecordReader class diagram



ImageRecordWriter class diagram

Fig. 5. New class diagrams designed for the image conversion function

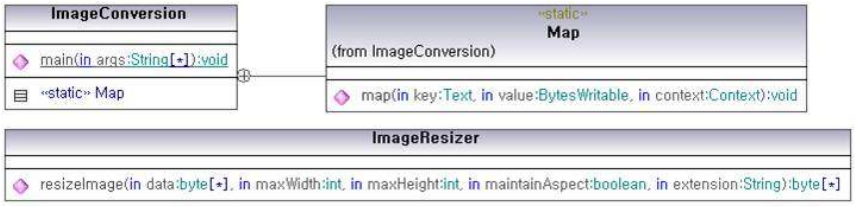


Fig. 6. ImageConversion, `Map`, and `ImageResizer` class diagrams

The design of the video conversion module is very similar to that for image conversion. We design `VideoFileInputFormat` and `VideoRecordReader` receiving a video file from HDFS as key and value pairs compatible with the MapReduce framework for processing. We also design `VideoFileOutputFormat` and `VideoRecordWriter` classes that write output data in HDFS. The `Resizer`, `MyVideoListener`, and `VideoConverter` classes are responsible for processing video transcoding in the video conversion module. These three classes are designed using Xuggler libraries. Method `convertVideo` in `VideoConverter` transcodes the input video data as a byte stream form according to the file size and format of the video as set by the users and administrators. **Fig. 7** illustrates the four class diagrams for the image conversion function. **Fig. 8** shows `Resizer`,

`MyVideoListener`, `VideoConverter`, `VideoConversion` with `main ( )`, and `Map` class diagrams.

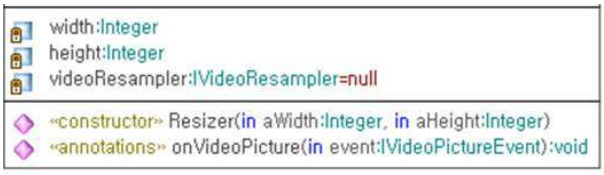
VideoFileInputFormat class diagram

VideoFileOutputFormat class diagram

VideoRecordReader class diagram

VideoRecordWriter class diagram

Fig. 7. Newly designed class diagrams for video conversion funciton



Resizer class diagram

MyVideoListener class diagram

VideoConverter class diagram

`VideoConversion` and ap class diagram

Fig. 8. Class diagrams for transcoding processing

5.2 Implementation

In this section, we focus on the detailed description of the implementation of MbTD, the component that plays an important role in our system. MbTD is responsible for the processing of video data through the MapReduce framework. **Fig. 9** shows the detailed MapReduce-based programming strategy of MbTD.

First, we implement `InputFormat` that transfers original video data sets stored on HDFS. The `InputFormat` plays two significant roles. The first role is to provide information about the number of map tasks for the MapReduce framework in advance. Therefore, the map tasks are prescheduled in the MapReduce framework. The second role is to read a record that is transferred to `map ()` of a map class from the original video datasets. This function is performed by `RecordReader`. The `RecordReader` provided from `FileInputFormat` is designed to read one line from a source file and pass it to `map ()`.

Next, we implement Mapper to process each record received from the `RecordReader`. Mapper receives a video file name and byte stream of the video file as key and value pairs from `RecordReader`. The key and value pairs are processed by `map ()` in a parallel and distributed manner. This transcoding processing is carried out by the Xuggler and JAI media libraries. The result of complete processing is transmitted in the form of converted key and value pairs to `OutputFormat`.

Finally, we also implement `OutputFormat` that writes the complete output data processed by Mapper to HDFS. That is, the `OutputFormat` rewrites key and value pairs as a file on HDFS. In this paper, we implemented MbTD, the main component of our system, by using the implementation strategy explained above.

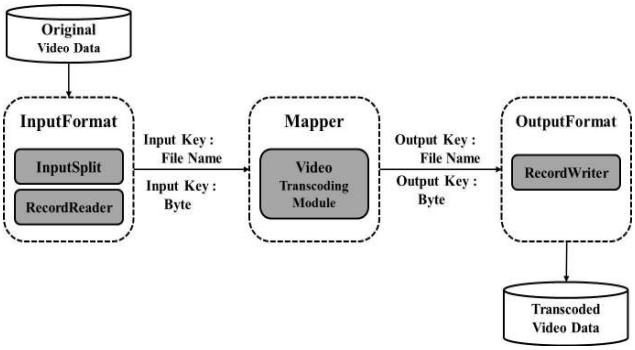


Fig. 9. The MapReduce-based programming strategy in MbTD

6. Performance Evaluation

In this section, we present the experimental results, describing optimal Hadoop options for processing image and video files. Moreover, we explain the hardware specifications of our cloud cluster, image and video data sets used in performance evaluation and the experimental methods for measuring media transcoding time.

6.1 Experiment Environment

Our performance evaluation is performed on a single enterprise scale cluster that is composed of 1 master node and 27 computational nodes (data nodes called in HDFS). The only way to access the cluster is through the master node. Each node running on the Linux OS (Ubuntu 10.04 LTS) 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, Xuggler 3.4 for video transcoding and JAI 1.1.3 for image transcoding. Because the structure of the cluster is homogeneous, it provides a uniform evaluation environment.

In order to verify the performance for our transcoding functions including image and video transcoding processing, we use six types of video data sets (**Table 1**), including several 200MB video files and six types of image data sets, (**Table 1**) including several approximately 20MB image files.

Table 1. Image and video data sets for performance evaluation

Video data sets						
Size of file	1GB	2GB	4GB	8GB	10GB	50GB
Number of video files	5	10	20	40	50	250
Image data sets						
Size of file	1GB	2GB	4GB	8GB	10GB	50GB
Number of video files	52	104	208	416	520	2594

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	200MB	60MB
Duration	3 min 19s	3 min 19s
Resolution	1280 x 720	320 x 240

We measure the total transcoding time of image and video transcoding modules. For evaluating the image transcoding function, we focus on measuring the total time to transcode large amounts of image data sets (JPG files) into a specific format (PNG files). For evaluating the video transcoding function, we measure the encoding time to transcode large sizes of video files into target files. The parameters for each original and target transcoded video file are listed in **Table 2**.

During the experiment, the following default options in Hadoop are used. (1) The number of block replications is set to 3. (2) The block size is set to 64MB. In order to verify the efficiency of our system, we conduct three sets of experiments: (1) examine how a change in cluster size affects performance speedup, (2) explore different Hadoop options for different block sizes

(32, 64, 128, 256, 512), and (3) explore different Hadoop options for different block replication factors (1, 2, 3, 4, 5).

6.2 Changing Cluster Size for Speedup Performance

In the first set of experiments, we measure the total transcoding time for image and video transcoding functions under varying cluster size, including 1, 4, 8, 12, 16, 20, 24, 28 nodes. Hadoop default options are explained in the experiment environment section.

Table 3. Total image and video transcoding time for various cluster size and speedup (s)

The total image transcoding time (s)		Image data set size					
		1GB	2GB	4GB	8GB	10GB	50GB
Cluster size (node)	1	247	473	933	1852	2311	11507
	4	75	129	246	473	590	2887
	8	48	75	130	247	301	1454
	12	39	58	94	168	210	976
	16	31	48	75	130	158	735
	20	30	39	66	111	131	590
	24	30	40	57	93	112	508
	28	23	32	49	84	102	428
Speedup	1	1	1	1	1	1	1
	4	3.29	3.67	3.79	3.92	3.92	3.99
	8	5.15	6.31	7.18	7.50	7.68	7.91
	12	6.33	8.16	9.93	11.02	11	11.79
	16	7.97	9.85	12.44	14.25	14.63	15.66
	20	8.23	12.13	14.14	16.68	17.64	19.50
	24	8.23	11.83	16.37	19.91	20.63	22.65
	28	10.74	14.78	19.04	22.05	22.66	26.89
The total video transcoding time (s)		Video data set size					
		1GB	2GB	4GB	8GB	10GB	50GB
Cluster size (node)	1	863	1710	3465	6842	8498	42471
	4	272	506	952	1804	2268	11040
	8	183	276	512	953	1188	5583
	12	170	196	363	640	793	3720
	16	166	188	282	517	620	2824
	20	165	171	247	412	498	2226
	24	161	166	203	356	414	1841
	28	124	169	207	311	375	1623
Speedup	1	1	1	1	1	1	1
	4	3.17	3.38	3.64	3.79	3.75	3.87
	8	4.72	6.20	6.77	7.18	7.15	7.66
	12	5.08	8.72	9.55	10.69	10.72	11.49
	16	5.20	9.10	12.29	13.23	13.71	15.13
	20	5.23	10.00	14.03	16.61	17.06	19.20
	24	5.36	10.30	17.07	19.22	20.53	23.22
	28	6.96	10.12	16.74	22.00	22.66	26.33

We also conduct parallel speedup measurements. Parallel speedup refers to how many times faster the parallel and distributed executions are compared to running the transcoding

functions implemented by the same MapReduce programming on a single node. If speed up is greater than 1, there is at least some gain from carrying out the work in parallel. If speedup is the same as the number of machines, our cloud server and MapReduce programming has a perfect scalability and ideal performance. Speedup is defined as: Speedup (n) = transcoding time on 1 node / transcoding time on n nodes.

Table 3 shows the transcoding time as a function of cluster size and speedup. **Fig. 10(a)** shows the result of the experiment for each cluster size in an image transcoding module. **Fig. 10(c)** illustrates the result of speedup in the same module. **Fig. 10(b)** and **Fig. 10(b)** show the effect of cluster size and speed up in a video transcoding module, respectively. According to **Table 3**, our Hadoop-based media transcoding system shows excellent performance in image and video transcoding functions for very large image and video files. For example, with image transcoding, our system takes 428 s (approximately 7 min) to conduct image transcoding processing for 50GB in 28 nodes. For video transcoding, it takes 1623 s (approximately 27 min) under the same conditions.

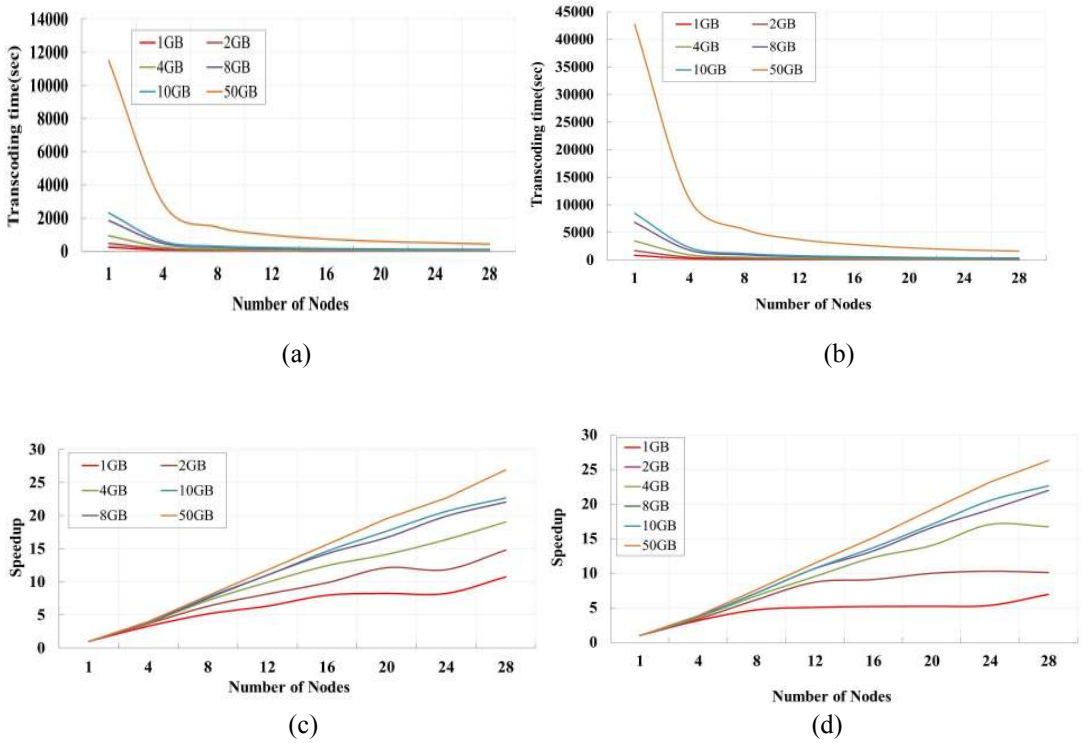


Fig. 10. (a) Transcoding time versus cluster size using the image transcoding module and (b) the video transcoding module (c) Speedup versus cluster size in the image transcoding module and (d) video transcoding module

From **Fig. 10 (a)** and **(b)**, for 4, 8 and 12 nodes the running time decreases dramatically and from 16 nodes to 28 nodes the transcoding time reduces gradually in both of the two modules. From **Fig. 10 (c)** and **(d)**, speedup performance for 10 and 50GB data sets are higher compared to speedup performances for 1, 2, 4, 8GB datasets, implying that our system exhibits good performance when the size of the data set increases.

6.3 Changing Block Size Factor

For the second set of experiment, we measure the total elapsed time with the different Hadoop options with respect to block size factor (default: 64MB). Hadoop processes large amounts of data sets in a parallel and distributed manner after data sets are chunked into a 64MB chunk size. However, users and programmers can change the block size options in order to improve the performance of data processing according to the size and type of unstructured data. Thus, in order to determine the optimal block size condition, we measure the total media transcoding time with five block size options, 32, 64, 128, 256, 512MB. **Table 4** lists the measured image and video transcoding time in seconds for different block sizes.

Table 4. Total image and video transcoding time for various values of block size (s)

The total image transcoding time (s)		Image data set size					
		1GB	2GB	4GB	8GB	10GB	50GB
Block size (MB)	32	23	30	49	84	102	437
	64	23	32	49	84	102	428
	128	23	31	50	84	102	436
	256	24	30	49	85	103	436
	512	23	32	50	85	102	436
The total video transcoding time (s)		Video data set size					
		1GB	2GB	4GB	8GB	10GB	50GB
Block size (MB)	32	173	226	335	532	657	2837
	64	124	169	207	311	375	1623
	128	103	108	120	199	209	820
	256	102	103	106	106	116	443
	512	102	105	105	111	109	444

As shown in **Table 4, Fig. 11 (a)**, and **Fig. 11 (b)**, there is no difference in performance for different block size options in the image transcoding module (**Fig. 11 (a)**), whereas the transcoding performance for video transcoding module with Hadoop block size options of 256 or 512MB are better compared to 32, 64, and 128MB. From the results, we find that when the block size option is set to a value greater than or close to the original file size, our system provides good performance for media transcoding processes. In fact, since one video data set has a file size of 200MB, 256 or 512 MB block sizes show the best performance for transcoding processing.

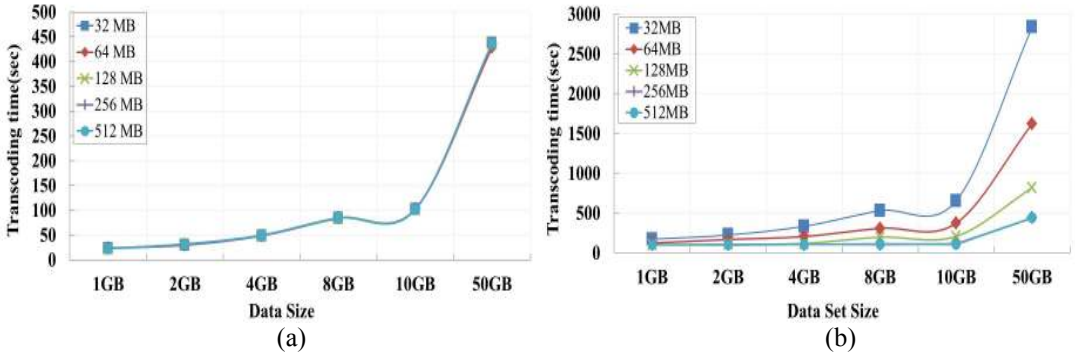


Fig. 11. (a) Total image transcoding time versus data size for various block sizes (b) Total video transcoding time versus data size for various block sizes

6.4 Changing Block Replication Factor

In the third set of experiments the total transcoding time with different Hadoop options with respect to block replication factor (default: 3) is measured. When large amounts of data sets are stored in HDFS, HDFS splits the data set into fixed size blocks for quick searching and processing. In fact, with the Hadoop default option for block replication, replicated data is stored on three data nodes of HDFS in order to rebalance, move copies around, and continue data replication when system faults such as disk failures or network connection problems occur. Hence, in order to verify how block replication factors affects performance, we measure the elapsed time in order to complete media transcoding processing. Five values of block replication factor, 1, 2, 3, 4 and 5 are used in the experiment. **Table 5** lists the measured image and video transcoding time in seconds for different block replication factor values.

Table 5. Total image and video transcoding time for various values of block replication factor (s)

The total image transcoding time (s)		Image data set size					
		1GB	2GB	4GB	8GB	10GB	50GB
Block replication (EA)	1	28	40	63	114	135	1092
	2	22	30	50	84	103	436
	3	23	32	49	84	102	428
	4	24	30	48	85	102	435
	5	23	31	48	84	102	435
The total video transcoding time (s)		Video data set size					
		1GB	2GB	4GB	8GB	10GB	50GB
Block replication (EA)	1	144	220	309	467	517	2330
	2	160	176	239	323	384	1616
	3	124	169	207	311	375	1623
	4	160	165	212	326	389	1643
	5	166	170	224	347	400	1492

According to **Table 5**, **Fig. 12 (a)**, and **Fig. 12 (b)**, two modules show the best performance when the block replication factor value is set to three. The worst performance occurs when the block replication factor is set to one, since new blocks with problems should be copied and transferred to a new data node on HDFS when disk failure and data loss occur. If the block replication factor is set to more than two, the process delay for performing fault tolerance does not occur. In addition, this performance degradation is caused by rescheduling job tasks in the master node in order to cope with recovering system failures.

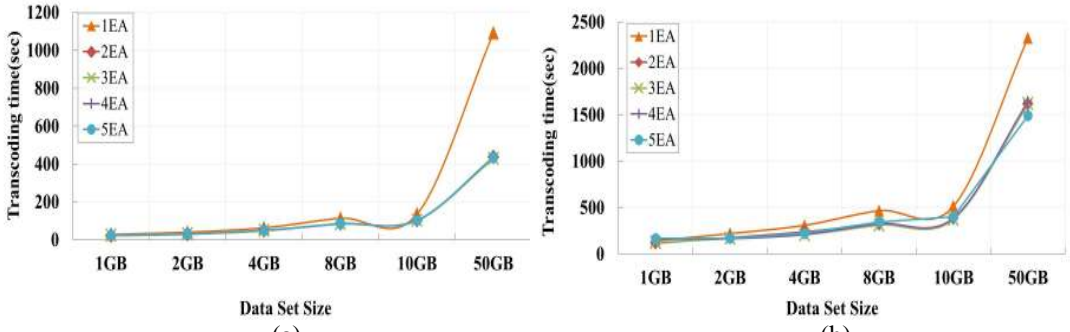


Fig. 12. (a) Total image transcoding time and total video transcoding time (b) versus data set size for various values of block replication factor

Although a similar performance is provided in both modules when the block size option value is set to more than three, we highly recommend that the block replication factor to be set to three, since many block replicas result in an unnecessary waste of storage space and more processing time for copying and transferring a large amounts of blocks.

6.5 Comparing two versions of video transcoding module

To compare the transcoding performance of our Hadoop-based approach with that of the traditional parallel processing approach, we implemented two versions of the video transcoding module running on the same cluster. The first version is our Hadoop-based transcoding module, whereas the other version is the traditional frame-based parallel transcoding module. We exploit the Media Encoding Cluster [35] for the traditional approach.

The Media Encoding Cluster written in C and C++ is the first open source project that deals with frame-based encoding in a distributed and parallel manner in commodity hardware to reduce the encoding time for a file. That is, to encode original media files into target files, our Hadoop-based transcoding approach splits media files into fixed blocks, while the Media Encoding Cluster splits media files into frame units.

We test and compare both approaches using the same video data sets. In case of the Hadoop-based version, we use the default Hadoop options explained in the experiment environment section. **Table 6** lists the total transcoding time of the two versions with a speedup calculation that is different from the speedup used in Section 6.2. The speedup used in this section shows how many times the Hadoop-based transcoding executions are faster than the traditional parallel-based transcoding executions. Speedup is defined as: Speedup = traditional frame-based parallel transcoding time / Hadoop-based transcoding time.

Table 6. Total transcoidng time results of two versions of video transcoding module

Video data set size	Traditional parallel frame-based transcoding time (s)	Hadoop-based transcoding time (s)	Speedup
1GB	264	124	2.12
2GB	463	169	2.73
4GB	928	207	4.48
8GB	1865	311	6.00
10GB	2327	375	6.20
50GB	11601	1623	7.14

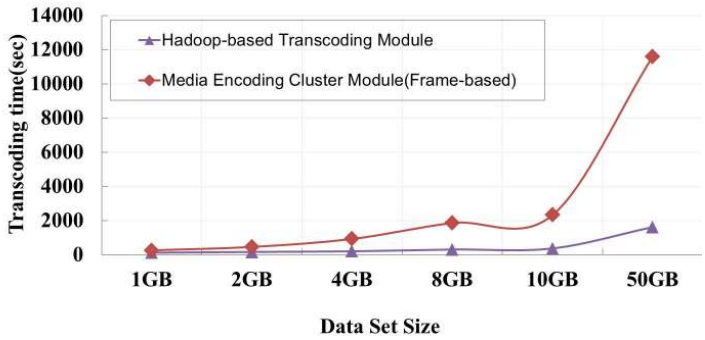


Fig. 13. Comparison of transcoding time between Hadoop-based trnascoding and Meida Encoding Cluster Module (frame-based)

According to **Table 6** and **Fig. 13**, the Hadoop-based transcoding module exhibits better

performance than the Media Encoding Cluster in terms of execution time in all the data sets. For instance, the total transcoding times for completing the transcoding process for 50 GB of the Hadoop-based transcoding module and the Media Encoding Cluster are approximately 3 h and 20 min and 27 min, respectively. From the speed up results, it is observed that the difference in performance between the two versions increases when the data set size increases. This means that there is much to be gained from our approach when processing data sets of larger sizes. The Media Encoding Cluster exhibits lower performance than our module because it involves greater overhead due to the steps for splitting and merging the original media files. Our approach splits original video files into 64 MB blocks and the blocks are merged after the transcoding process in MbTD, whereas the Media Encoding Cluster splits original video files into a significantly larger number of frame units compared to the Hadoop-based transcoding module's blocks and merges the chunked frames into target video files. In fact, in case of a 1GB data set (200 MB, 29 frames, 3 min 19 s), our module creates 20 chunked blocks of 64 MB, while the Media Encoding Cluster produces approximately 29000 chunked frames.

7. Conclusion and Future work

In this paper, we briefly review our social media cloud computing service environment (SMCCSE) and a social media cloud computing PaaS platform. In order to implement social media transcoding functions for transcoding image and video content into a specific format according to user transcoding requirements, we proposed a Hadoop-based multimedia transcoding system in the PaaS platform of SMCCSE. In order to reduce the transcoding time and ensure transcoded image and video quality, we apply a HDFS and MapReduce framework to our system, which are emerging technologies in the cloud computing field. Our system overcomes the difficulties related to emerging and merging policies in distributed video processing and fault tolerance and load balancing management in large-scale distributed systems by obeying Hadoop policies.

In the performance evaluation section, we focus on measuring the total transcoding time for various sets of experiments: (1) a change in cluster size for speedup of performance, (2) different Hadoop options with respect to block size (32, 64, 128, 256, 512MB), (3) different Hadoop options for different block replication factors (1, 2, 3, 4, 5). Through these experiments, we experimentally verified the excellent performance of our system in media transcoding processing and identified the ideal Hadoop options suitable for media transcoding processing. When the block size option is set to a value greater than or close to the original file size and the block replication factor value is set to three, our system delivers good performance for media transcoding processes. Moreover, in terms of transcoding execution time, our Hadoop-based transcoding approach implemented using Java exhibits better performance than the traditional frame-based parallel approach implemented using C and C++.

In the future, we will plan to improve the advanced splitting and merging algorithms and a load balancing scheme specified in media transcoding processing in Hadoop. We will also implement distributed video transcoding streaming services optimized for our system.

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communication of the ACM*, vol.51, no.1, pp.107-113, Jan. 2008. Article (CrossRef Link)

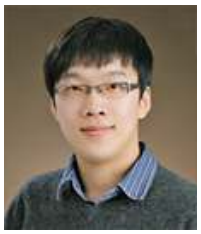
[2] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop distributed file system," in *Proc.*

of 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, pp.1-10, May. 2010. Article (CrossRef Link)
[3] S. Islam and J.-C.Gregoire, "Giving users an edge: A flexible cloud model and its application for multimedia," *Future Generation Computer Systems*, vol.28, no.6, pp.823-832, Jun. 2012. Article (CrossRef Link)
[4] W. Gropp, et al., "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing*, vol.22, no.6, pp.789-828, Sep. 1996. Article (CrossRef Link)
[5] M. Kim and H. Lee, "SMCC: Social media cloud computing model for developing SNS based on social media," *Communications in Computer and Information Science*, vol.206, pp.259-266, 2011. Article (CrossRef Link)
[6] G. Barlas, "Cluster-based optimized parallel video transcoding," *Parallel Computing*, vol.38, no.4-5, pp.226-244, Apr. 2012. Article (CrossRef Link)
[7] I. Ahmad, X. Wei, Y. Sun and Y.-Q. Zhang," Video transcoding: An overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol.7, no.5, pp.793-804 Oct. 2005. Article (CrossRef Link)
[8] Y.-K. Lee, et al., "Customer requirements elicitation based on social network service," *KSII Transactions on Internet and Information Systems*, vol.5, no.10, pp.1733-1750, Oct. 2011. Article (CrossRef Link)
[9] S. Mirri, P. Salomoni and D. Pantieri, "RMob: Transcoding rich multimedia contents through web services," *in Proc. of 3rd IEEE Consumer Communications and Networking Conference*, vol.2, pp.1168-1172, Jan. 2006. Article (CrossRef Link)
[10] Hadoop MapReduce project, http://hadoop.apache.org/mapreduce/
[11] Z. Lei, "Media transcoding for pervasive computing," in *Proc. of 5th ACM Conf. on Multimedia*, no4, pp.459-460, Oct. 2001. Article (CrossRef Link)
[12] D.M. Boyd and N.B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol.13, no.1, pp.210-230, Oct. 2007. Article (CrossRef Link)
[13] Apache Hadoop project, http://hadoop.apache.org/
[14] C.L. Covle, and H. Vaughn, "Social Networking: Communication revolution or evolution?," *Bell Labs Technical Journal*, vol.13, no.2, pp.13-17, Jun. 2008. Article (CrossRef Link)
[15] J. Guo, F. Chen and L. Bhuyan, R. Kumar, "A cluster-based active router architecture supporting video / audio stream transcoding service," in *Proc. of Parallel and Distributed Processing Symposium*, Apr. 2003. Article (CrossRef Link)
[16] Y. Sambe, S. Watanabe, D. Yu, T. Nakamura, N. Wakamiya, "High-speed distributed video transcoding for multiple rates and formats," *IEICE Transactions on Information and Systems*, vol.E88-D, no.8, pp.1923-1931, Aug. 2005. Article (CrossRef Link)
[17] M.-J. Kim, H. Lee, H. Lee, "SMCCSE: PaaS Platform for processing large amounts of social media," in *Proc. of the 3rd international Conf. on Internet*, pp.631-635, Dec. 2011. Article (CrossRef Link)
[18] J. Shafer, S. Rixner and A.L. Cox, "The Hadoop distributed file system: Balancing portability and performance," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software*, pp.122-133, Mar. 2010. Article (CrossRef Link)
[19] Z. Tian, J, Xue, W. Hu, T. Xu and N. Zheng, "High performance Cluster-based Transcoder," in *Proc. of 2010 International Conf. on Computer Application and System Modeling*, vol.2, pp. 248-252, Oct. 2010. Article (CrossRef Link)
[20] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol.25, no.6, pp.599-616, Jun. 2009. Article (CrossRef Link)
[21] R. Lammel, "Google's MapReduce programing model - Revisited," *Science of Computer Programming*, vol.68, no.3, pp.208-237, Oct. 2007. Article (CrossRef Link)
[22] M.A. Vouk, "Cloud Computing – Issues, research and implementations," in *Proc. of 30th International Conf. on Information Technology Interfaces*, pp.31-40, Jun. 2008. Article (CrossRef

Link)
[23] C. Poellabauer and K. Schwan, "Energy-aware media transcoding in wireless systems," in *Proc. of Second IEEE Annual Conf. on Pervasive Computing and Communications*, pp.135-144, Mar.ch, 2004. Article (CrossRef Link)
[24] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "A view of cloud computing," *Communication of the ACM*, vol.58, no.4, pp.50-58, April, 2010. Article (CrossRef Link)
[25] Xuggler Java library, http://www.xuggle.com/xuggler/index
[26] Shenoy, J. Prashant, Vin, M. Harrick, "Efficient striping techniques for multimedia file servers," in *Proc. of the IEEE International Workshop on Network and Operating System Support for Digital Audio and Video*, pp.25-36, 1997. Article (CrossRef Link)
[27] M.A. Weifeng, and M. Keji, "Research on java imaging technology and its programming framework," *Lecture Notes in Electrical Engineering*, vol.72, pp.61-68, 2010. Article (CrossRef Link)
[28] D. Seo, J. Lee, C. Choi, H. Choi, I. Jung, "Load distribution strategies in cluster-based transcoding servers for mobile clients," *Lecture Notes in Computer Science*, vol.3983, pp.1156-1165, May. 2006. Article (CrossRef Link)
[29] S. Roy, B. Shen, V. Sundaram, R. Kumar, "Application level hand off support for mobile media transcoding sessions," in *Proc. of the 12th International Workshop on Network and Operating Systems for Digital Audio and Video*, pp.95-104, May. 2002. Article (CrossRef Link)
[30] D. Seo, J. Kim and I. Jung, "Load distribution algorithm based on transcoding time estimation for distributed transcoding servers," in *Proc. of 2010 Conf. on Information Science and Applications*, article no.5480586, Apr. 2010. Article (CrossRef Link)
[31] R.L. Grossman, "The Case for Cloud Computing," *IT Professional Journal*, vol.11, no.2, pp.23-27, Mar. 2009. Article (CrossRef Link)
[32] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol.1, no.1, pp.7-18, May. 2010. Article (CrossRef Link)
[33] S. Ghemawat, H. Gobioff and S.-T. Leung, "The google file system," *Operating Systems Review (ACM)*, vol.37, no.5, pp.29-43, Oct. 2003. Article (CrossRef Link)
[34] X. Liao and H. Jin, "A new distributed storage scheme for cluster video server," *Journal of Systems Architecture*, vol.51, no.2, pp.79-94, Feb. 2005. Article (CrossRef Link)
[35] Media Encoding Cluster Project, http://www.encodingcluster.com

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

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



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.

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.

Changsung Jeong is a professor at the Department of Electrical Engineering at Korea University. He received his M.S.(1985) and Ph.D(1987) from Northwestern University, and B.S.(1981) from Seoul National University. Before joining Korea University, he was a professor at POSTECH during 1982-1992. He was on editorial board for Journal of Parallel Algorithms and Application in 1992-2002. Also, he has been working as a chairman of Collaborative Supercomputing Group in GFK(Global Forum in Korea) since 2001, and a chairman of Computer Chapter at Seoul Section of IEEE region 10. His research interests include distributed concurrent computing, grid computing, cloud computing, and collaborative ubiquitous computing.