A Study on Efficient Design of a Multimedia Conversion Module in PESMS for Social Media Services

Jongjin Jung[1], Myoungjin Kim[2], Hanku Lee[3]
[1]Korea Electronics Technology Institute, Center for Social Media Cloud Computing, Konkuk University, South Korea
[2,3]Center for Social Media Cloud Computing, Konkuk University, South Korea
Email: mozzalt@keti.re.kr[1], tough105@konkuk.ac.kr[2], hlee@konkuk.ac.kr[3]

Article Info

Article history:

Received Jun 12, 2015
Revised Aug 20, 2015
Accepted Aug 26, 2015

Keyword:

PESMS
Image Conversion
MapReduce
Hadoop
Media Transcoding

ABSTRACT

The main contribution of this paper is to present the Platform-as-a-Service (PaaS) Environment for Social Multimedia Service (PESMS), derived from the Social Media Cloud Computing Service Environment. The main role of our PESMS is to support the development of social networking services that include audio, image, and video formats. In this paper, we focus in particular on the design and implementation of PESMS, including the transcoding function for processing large amounts of social media in a parallel and distributed manner. PESMS is designed to improve the quality and speed of multimedia conversions by incorporating a multimedia conversion module based on Hadoop, consisting of Hadoop Distributed File System for storing large quantities of social data and MapReduce for distributed parallel processing of these data. In this way, our PESMS has the prospect of exponentially reducing the encoding time for transcoding large numbers of image files into specific formats. To test system performance for the transcoding function, we measured the image transcoding time under a variety of experimental conditions. Based on experiments performed on a 28-node cluster, we found that our system delivered excellent performance in the image transcoding function.

Corresponding Author:

Jongjin Jung and Hanku Lee
Smart Media Research Center, Korea Electronics Technology Institute, and
Center for Social Media Cloud Computing, Konkuk University
Jong Jin, Jung Seoul, South Korea
Email: mozzalt@keti.re.kr

1. INTRODUCTION

Today, nearly everyone uses smart devices such as smart phones, tablet computers, and smart TVs. In addition, they use social networking service (SNS) applications that enable them to take high quality pictures, record videos, and upload and download them. These images and videos have high resolution; therefore, the files are quite large. Users also want seamless, real-time service; furthermore, they want all multimedia services to be independent of the platform operating system (OS), the display device resolution, and so on. In order to meet these requirements, the development environments of SNSs need to be supported with larger storage systems, a larger database system, more elastic computing resources, and data processing techniques for numerous multimedia devices. Service providers (SPs) must provide increasingly massive storage spaces to support the volume of social media created daily by users. SPs must also install more powerful hardware, only to have to change it again some time later. To handle the range of user device types, SPs must have all kinds of multimedia. However, this may not be the best solution.

This paper proposes the Platform-as-a-Service (PaaS) Environment for Social Multimedia Service (PESMS), which was derived as a PaaS model of the Social Media Cloud Computing Service Environment (SMCCSE) [1], [2]. The main objective of PESMS is to support development environments for developing or implementing social media services [3], [4], [5], [6] using cloud computing techniques and elastic computing resources in a cloud computing environment. In this study, we focused on the implementation of PESMS. We designed and implemented partially functional social media modules for converting images into JPEG, GIF, PNG, BMP, TIFF, and other formats and modules for resizing images using PESMS.

2. RELATED WORKS

Cloud computing: Cloud computing technology has already become a popular service for efficient system maintenance, thin-client devices, powerful software services and other applications. According to the National Institute of Standards and Technology (NIST), cloud computing is defined as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [7]. Thus, the developer, the system manager, or a common user can easily use the platform or service without the need for complex installation and associated major constraints of time, place, and occasion.

Social-media-service model using cloud computing: Social media–based social networking services (SNSs) have also become popular. Most people use one or more social service applications, and they want elastic social media service. People share their news, photographs, and multimedia with social media services, and they want to enjoy high-quality multimedia and high-speed service. Furthermore, because people have different devices, several versions are needed, even though the content is the same. More specifically, there may be different versions of the same content that differ in resolution, quality (e.g., thumbnail or mobile phone version), or codec. For example, it is known that Facebook had 70 billion images in 2012. To find the most appropriate content and deliver it to the user (or more precisely, the user’s device), accurate information about each multimedia object is very important. To obtain this information, social multimedia retrieval tools (to extract features such as color, texture, and histogram) could be required. To address these issues, the system or platform for a social media service must have more powerful computing equipment. However, some time later, as more and more people use social media services, more multimedia data will be generated, thus necessitating more storage, larger database systems, and much speedier networks. As a result, the trend is for many service providers to implement social media services with cloud computing.

Hadoop: Hadoop, developed by Apache, is an open source software project. It enables distributed processing of large data sets across clusters of commodity servers [8]. It is designed to scale up from a single server to thousands of machines, with very high degree of fault tolerance. Rather than relying on high-end hardware, the resiliency of these clusters comes from the software’s ability to detect and handle failures at the application layer. Thus, it is useful for data-intensive distributed applications, which are capable of handling thousands of nodes and petabytes of data. Hadoop changes the economics and the dynamics of large-scale computing. Its impact can be summarized in four notable characteristics, listed in Table 1. These four characteristics enable computing solutions that are scalable, cost effective, flexible, and fault tolerant.

Table 1. Characteristics of Hadoop

Characteristic	Description
Scalability	A cluster can be expanded by adding new servers or resources without having to move, reformat, or change the dependent analytic workflows or applications.
Cost-effectiveness	Hadoop brings massively parallel computing to commodity servers. The result is a sizeable decrease in the cost per terabyte of storage, which in turn makes it affordable to model all your data.
Flexibility	Hadoop is schema-less and can absorb any type of data, structured or not, from any number of sources. Data from multiple sources can be joined and aggregated in arbitrary ways enabling deeper analyses than any one system can provide.
Fault tolerance	When you lose a node, the system redirects work to another location of the data and continues processing without missing a beat.

Hadoop has four main components: Hadoop Common, Hadoop Distributed File System (HDFS), MapReduce, and “Yet Another Resource Negotiator” (YARN). These are explained in Table 2.

Two of these components, Hadoop Distributed File System (HDFS) and MapReduce, are utilized in PESMS. As mentioned previously, Hadoop has scalability. MapReduce is the heart of Hadoop, and it is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce framework provides a specific programming model and a runtime system for processing and creating large data sets that are suitable for various real-world tasks [9]. This framework also handles automatic scheduling, communication, and synchronization during the processing of huge data sets, and it has a fault tolerance capacity. The MapReduce programming model is executed in two main steps, called *mapping* and *reducing*, performed by the *mapper* and *reducer* functions, respectively. Each phase requires a list of key and value pairs as the input and output. In the mapping step, MapReduce receives the input data set and feeds each data element to the mapper in the form of key and value pairs. In the reducing step, all of the outputs from the mapper are processed, and the final result is generated by the reducer using the merging process. The HDFS component further contributes to Hadoop's scalability. All data in Hadoop are configured with HDFS; data in a Hadoop cluster are broken down into smaller pieces (called blocks) and distributed throughout the cluster. In this way, the mapping and reducing functions can be executed on smaller subsets of the user's larger data sets, and this provides the scalability that is needed for processing of big data.

Table 2. The Four Core Components of Hadoop Architecture

Core Component	Description
Hadoop Common	A module containing the utilities that support the other Hadoop components.
Hadoop Distributed File System (HDFS)	A file system that provides reliable data storage and access across all the nodes in a Hadoop cluster. It links together the file systems on many local nodes to create a single file system.
MapReduce	A framework for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of thousands of machines, in a reliable, fault-tolerant manner.
Yet Another Resource Negotiator (YARN)	The next-generation MapReduce, which assigns CPU, memory and storage to applications running on a Hadoop cluster. It enables application frameworks other than MapReduce to run on Hadoop, opening up a wealth of possibilities.

3. SMCCSE (SOCIAL MEDIA CLOUD COMPUTING SERVICE ENVIRONMENT)

SMCCSE is a development environment with which an SP can easily develop or implement cloud-based social media services [5] [10][11]. It provides an environment for supporting the development of SNSs, addressing numerous SNSs and providing more effective algorithms for processing high volumes of social media data and a set of mechanisms to manage the entire infrastructure [1], [2]. It is composed of four layers: the social media applications layer, the social service software library layer, the distributed social data processing layer, and the cloud virtualization platform layer. Figure 1 shows the SMCCSE architecture.

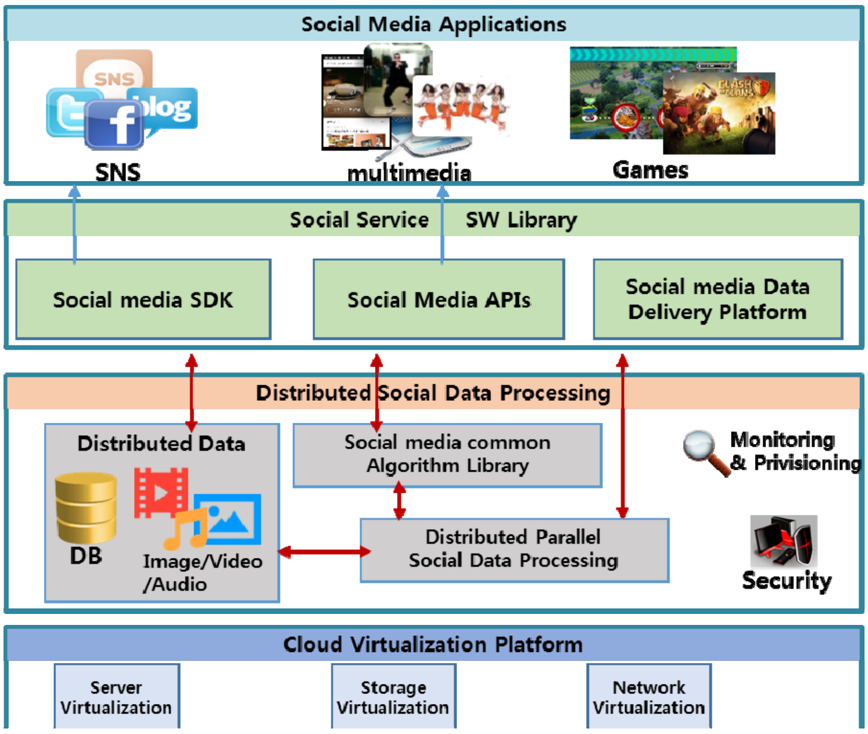
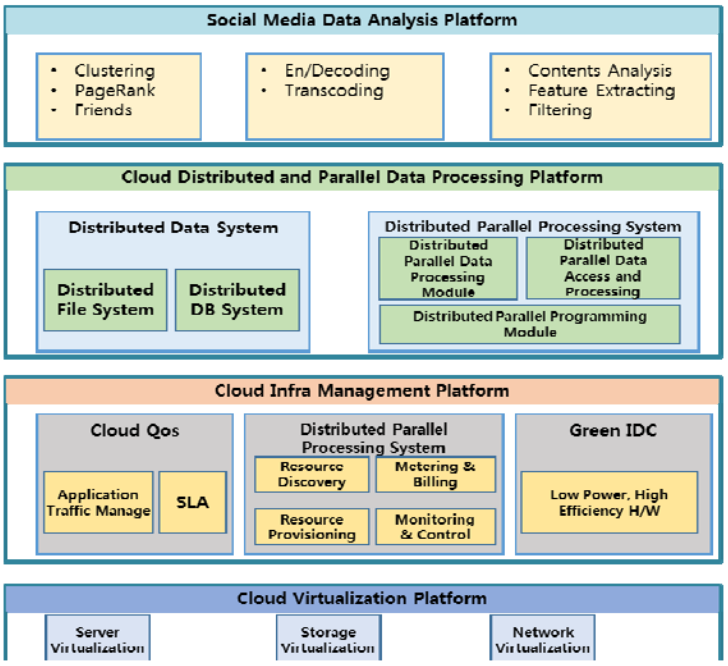

Social Media Applications
SNS
multimedia
Games
Social Service
SW Library
Social media SDK
Social Media APIs
Social media Data Delivery Platform
Distributed Social Data Processing
Distributed Data
DB
Image/Video /Audio
Social media common Algorithm Library
Distributed Parallel Social Data Processing
Monitoring & Privisioning
Security
Cloud Virtualization Platform
Server Virtualization
Storage Virtualization
Network Virtualization

Figure 1. The architecture of the Social Media Cloud Computing Service Environment (SMCCSE)

In the social media applications layer, many social applications (SNS, multimedia, games, and so on) can be developed using modules of the layer below, which is the social service software library. This layer has a software development kit (SDK) and many APIs, functions that are provided to the developer or SP as a form of software as a service (SaaS) for developing the various social media applications. In the distributed social data processing layer, many module-related distributed social data processes are provided as a form of PaaS [12]. In the cloud virtualization platform layer, huge hardware equipment is provided as a virtualized platform as a form of infrastructure as a service (IaaS).

4. PESMS (PAAS ENVIRONMENT FOR SOCIAL MULTIMEDIA SERVICE)

As mentioned in the Introduction, PESMS is a PaaS model of SMCCSE. PESMS is composed of a social media data analysis platform (social common algorithms library), a cloud distributed and data processing platform, and a cloud infra management platform, as shown in Figure 2.



Social Media Data Analysis Platform
Clustering
PageRank
Friends
En/Decoding
Transcoding
Contents Analysis
Feature Extracting
Filtering
Cloud Distributed and Parallel Data Processing Platform
Distributed Data System
Distributed File System
Distributed DB System
Distributed Parallel Processing System
Distributed Parallel Data Processing Module
Distributed Parallel Data Access and Processing
Distributed Parallel Programming Module
Cloud Infra Management Platform
Cloud Qos
Application Traffic Manage
SLA
Distributed Parallel Processing System
Resource Discovery
Metering & Billing
Resource Provisioning
Monitoring & Control
Green IDC
Low Power, High Efficiency H/W
Cloud Virtualization Platform
Server Virtualization
Storage Virtualization
Network Virtualization

Figure 2. The architecture of the Platform-as-a-Service (PaaS) Environment for Social Multimedia Service (PESMS)

Social media data analysis platform: The main role of the social media data analysis platform is to analyze usage patterns and relationships between the users and the social media data they demand and to provide encoding, decoding, transcoding, and transmoding functions in a library form. Transcoding is the conversion of a media file into file types suitable for numerous digital devices, and transmoding is the conversion of a media file into multiple files of more suitable sizes.

Cloud distributed and parallel data processing platform: This is a core part of PESMS. It can store, distribute, and process social media data created by users by applying HDFS, MapReduce, and a Hadoop database system (HBase) [13][14][15]. The social media data are delivered to various user devices such as mobile phones, smart pads, PCs, and TVs. The distributed and parallel data processing system has two subsystems: a distributed data system and a distributed parallel processing system. In the first, HDFS is adopted for a distributed social multimedia data system. In the second, MapReduce is adopted for a distributed parallel programming model. All functions of this platform are performed by social media common algorithms from the libraries in the social media data analysis platform [16][17]. The generated social media data (text, images, audio, and video) and database are stored in HDFS or HBase. Then, the stored data are processed in two steps using MapReduce.

Cloud infra management platform: The cloud infra management platform involves the concepts of cloud quality of service (QoS) and a green Internet data center (IDC), and it is used to manage and monitor computing resources that do not depend on a specific OS or platform. It includes resource scheduling, resource information management, resource monitoring, and virtual machine management. These functions are provided on Web services based on Eucalyptus. In addition, our IaaS is designed to offer flexible computing resources including servers, storage, and bandwidth using virtualization techniques based on Xen [18].

The most important function of PESMS is image and video conversion via transcoding and transmoding; by this means, large volumes of social media objects created by SNS users are efficiently transmitted to end-user devices. To accomplish this, we designed and implemented a partially functional image conversion module based on Hadoop. Figure 3 shows the architecture of the image conversion module. The transcoding and transmoding of images using Hadoop is as follows. First, user-created images are automatically distributed and stored in each node running on HDFS [19], [20]. Next, batch processing by MapReduce converts images stored in HDFS to appreciate another images. Our conversion module uses only a map step because it is not necessary to conduct a merging process for the results from the reduce step [21]. The map function is implemented by the SequenceFiles method in the map phase; this is used to input the file names and the file contents themselves as the keys and values, respectively, in a set of intermediate key/value pairs. Then, the image conversion module sets the file contents into byte type using a BytesWritable class. Finally, image data sets are processed in parallel in each node. Figure 4 shows the programming elements of the image conversion module and the implementation of the proposed module.

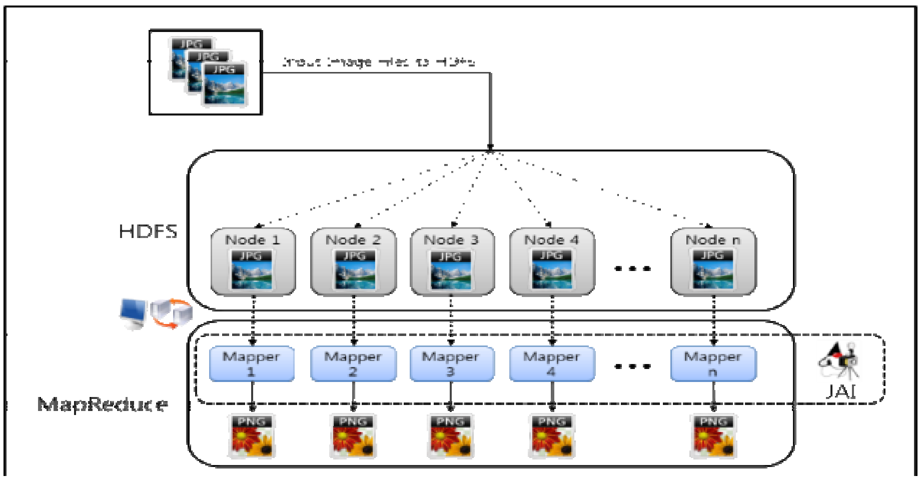


Figure 3. Image Conversion Module using PESMS

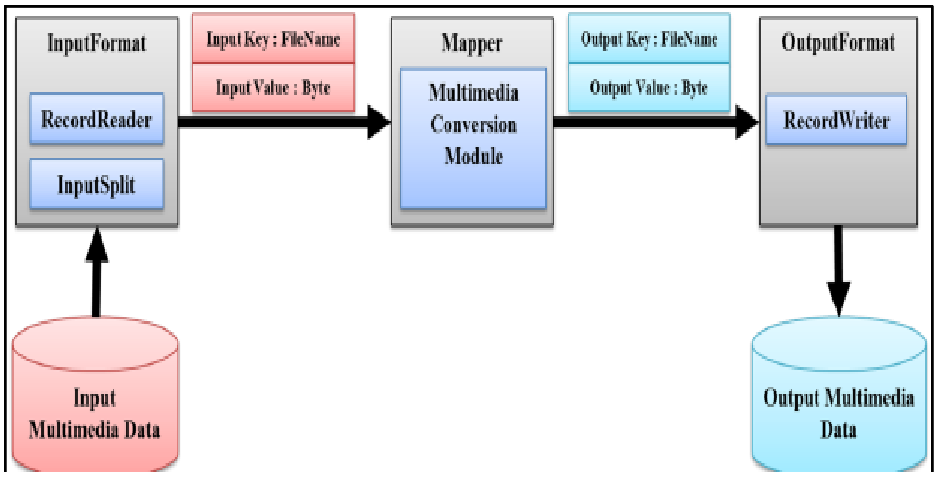


Figure 4. Programming Elements of the Image Conversion Module

The steps used for programming these processes are as follows. First, the conversion module reads image data in HDFS using the RecordReader method of the InputFormat class. InputFormat transforms the image data into sets of keys (file names) and values (bytes). Second, InputFormat passes the sets of keys and values to the Mapper class. Mapper processes the image data using the user-defined settings and methods for image conversion via the JAI library. Then, the conversion module converts the image data into specific formats suitable for a variety of devices such as smart phones and tabular and personal computers in a fully distributed manner. Mapper completes the image conversion and passes the results to the OutputFormat class as the key (file name) and value (bytes). Finally, Mapper passes the key and value set to OutputFormat. The RecordWriter method of the OutputFormat class writes the result as a file for HDFS.

5. SIMPLE USE CASE OF PESMS

An example of a simple social media service with PESMS is shown in Figure 5. The flow is as follows.

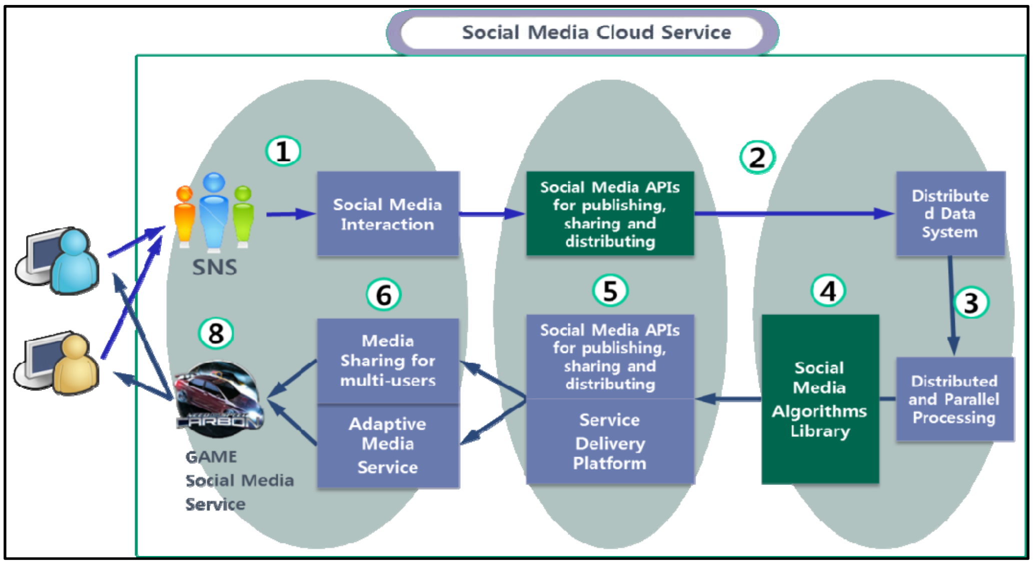

Social Media Cloud Service
1
2
SNS
Social Media Interaction
Social Media APIs for publishing, sharing and distributing
Distributed Data System
6
5
4
3
8
Media Sharing for multi-users
Adaptive Media Service
Social Media APIs for publishing, sharing and distributing
Service Delivery Platform
Social Media Algorithms Library
Distributed and Parallel Processing
GAME Social Media Service

Figure 5. Simple use case of social media service using PESMS

1. SNS users create and upload their social content.
2. This content is stored in the distributed data system using social media APIs. Content is then processed and classified in order to be found efficiently by a user search pattern and be available to recommend to users as appropriate.
3. Classified content is conducted to distributed and parallel processors by MapReduce in order to deliver to other users in the form of appreciated format.
4. Reformatted content is then delivered to the appropriate users by social media APIs and the service delivery platform.
5. As a result, large amounts of social multimedia data are efficiently shared in the SNS application because of PESMS.

6. PERFORMANCE EVALUATION

The experiments were conducted on a 28-node test bed, which is a single-enterprise scale cluster consisting of 27 data nodes (slave nodes) and 1 head node (master nodes). The only way to access the cluster is through the master node. All nodes run on Linux OS (Ubuntu 10.04 LTS). Each node is equipped with two 2.13 GHz Intel Xeon Quad-Core processors, 4 GB of registered ECC DDR memory, and a 1 TB SATA-2 HDD (7200 RPM). The machines were interconnected using a 1000 Mbps Ethernet adapter. Excluding the hardware specifications, the experimental environments used are as follows:

- ✓ To build a variety of experimental conditions, we used image datasets (Table 1) from eight groups. The average size of each image file is approximately 19.8 MB.
- ✓ To implement the image conversion function, we also used JAI 1.1.3 (Java Advanced Imaging) APIs and Java 1.6.0_23.

- ✓ To process the large data datasets on our test bed, we selected Hadoop-0.20.2. The default options selected for Hadoop are as follows: 1) the number of block replications is set to three, and 2) the block size is set to 64 MB.

We conducted seven sets of experiments for our performance evaluation of the partially implemented MapReduce-based cloud distributed and parallel data processing platform used in the PESMS. These experiments were chosen to provide an overview of the transcoding and transmoding functions. In particular, we measured the run time for the image conversion function based on MapReduce to convert large amounts of image datasets into specific formats (e.g., from JPG to PNG) suitable for a variety of mobile devices under a variety of conditions.

6.1. Performance of Transcoding

The objective of the first experiment was to measure the transcoding time and speedup for the image conversion function under varying cluster sizes. As shown in Figure 6, the run times decrease when the number of nodes increases. In particular, the elapsed times decrease dramatically for the first eight nodes. From 8 to 28 nodes, the run times are reduced gradually. In addition, we also measured the parallel speedup. The experiments were conducted with different numbers of parallel nodes to allow the parallel speedup to be calculated. The parallel speedup calculates how much faster the parallel and distributed execution is than running an image conversion function implemented using the same MapReduce programming in a single node. If the speedup is greater than 1, it indicates that there is at least some gain from carrying out the conversion in a parallel manner. If the speedup is equal to the number of machines, it indicates that our cloud server and MapReduce programming have perfect scalability and an ideal performance. The calculated speedups are shown in Figure 7. As the results show 2, 4, and 8 nodes used in parallel result in an ideal and perfect scalability. Although the performance is not optimum since 10 nodes, we can see that our cloud server has a high-performance throughput from the use of distributed processing. Moreover, for very large or very small image datasets, we can see that the throughput in the distributed processing performance is reduced in our cloud server. In fact, the calculated speedups of all datasets in the 28 nodes are approximately 11, 15, 19, 22, 23, 26, 27, and 27, respectively.

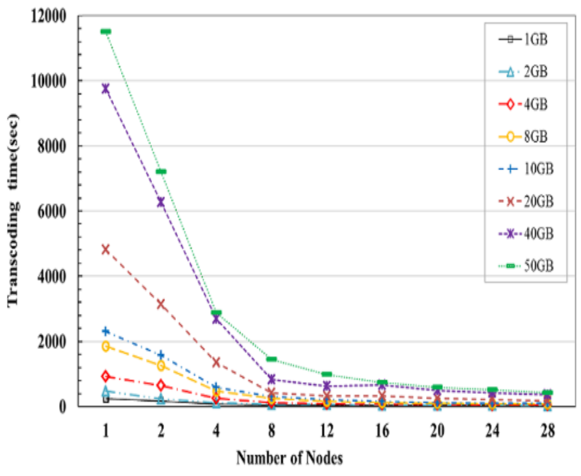


Figure 6. Transcoding Time versus Cluster Size

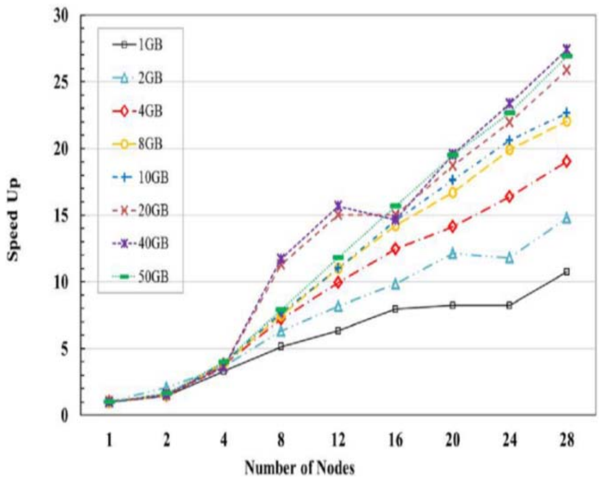


Figure 7. Speedup versus Cluster Size

6.2. Performance Based on Changes in Block Replication Options

The second experiment was conducted to measure the run times for the image conversion function according to the changes in the number of block replications. MapReduce splits large datasets into fixed-sized blocks, allowing a quick data search and processing. In fact, using the default replication value of 3, the replicated data is stored in three nodes of the HDFS to rebalance the data, move copies around, and keep the data replication high when system faults such as a disk failure, network connection problems, or other issues occur. The purpose of this experiment is to verify how block replication will materially affect the performance. The numbers of block replications used in experiment were 1, 2, 4, and 5 with a default value of 3. The experiment results are shown in Table 3. The experimental results indicate that the execution times are reduced when the block replication numbers increase. In particular, there is little difference in the execution times for small datasets of 1 to 8 GB, whereas for larger datasets, the difference in the execution times is bigger. In fact, for a 50 GB dataset, the execution times are 1,092, 437, and 428 s for replication factors of 1, 2, and 3 blocks, respectively (Table 4).

We wondered whether selecting as large a number of block replications as possible would be the best way to increase the performance and handle large amounts of datasets in our conversion module based on MapReduce. Through this experiment, we found that processing datasets in an HDFS using a large number of block replicas would bring about a significant amount of extra overhead. First, if the number of block replications is larger, an increase in storage capacity needed to store the block replicas occurs. The time required to store the block replicas also increases exponentially. For this reason, we evaluated the times required to store replicated blocks in an HDFS depending on the increase in the number of block replications. Table 5 shows the results of this experiment. As shown in the table, for a 50 GB dataset, the time taken to store three block replicas in HDFS is 1984 s, while the amount of time taken for five block replicas is 3587 s. The difference in the storage time between three and five block replicas is approximately 1,500 s. This execution time is much greater than the difference in the run time for an image conversion under the same conditions. Furthermore, overhead related with the job scheduling in the HDFS is generated. The point of this experiment is to determine whether the form and size of the datasets, programming techniques, business logic, and configuration of the cluster systems are effective for the processing when using a selected number of block replications

Table 3. Flickr Image Datasets Used for the Performance Evaluation

SECTION	CONTENT							
Name	Flicker dataset							
Format	JPG							
Source	Keyword: Sun							
Size	1 GB	2 GB	4 GB	8 GB	10 GB	20 GB	40 GB	50 GB

Table 4. Total Image Transcoding Times (s) for the Block Replication Factors

Block Replication	Image Dataset Size							
	1 GB	2 GB	4 GB	8 GB	10 GB	20 GB	40 GB	50 GB
1	28	40	63	114	135	496	975	1,092
2	25	32	50	85	103	192	358	437
3	27	32	50	85	102	186	359	428
4	30	33	50	85	103	188	358	437
5	30	33	50	85	103	192	359	437

Table 5. Execution Time (s) for Storing All Block Replications in HDFS

Block Replication	Image Dataset Size							
	1 GB	2 GB	4 GB	8 GB	10 GB	20 GB	40 GB	50 GB
1	17	34	69	130	172	377	581	672
2	33	68	129	257	332	754	1,120	1,392
3	41	84	163	344	428	837	1,790	1,984
4	69	137	237	512	667	1,490	2,234	2,878
5	110	215	301	671	842	1,890	2,829	3,587

6.3. Performance Based on Changes in the Block Size Option

The purpose of the third experiment was to measure the execution times according to the block size. Basically, Hadoop processes significant amounts of datasets after splitting them into a default block size value of 64 MB. We measured the run times for an image conversion using 16, 32, 64, 128, 256, and 512 MB block sizes, respectively. The results are shown in Table 6. As the results show, for a 64 MB block size, the run time is the best among the five cases studied. If a developer sets the block size in Hadoop to smaller than the file size (in our case, approximately 20 MB) included in the image dataset, the execution times increases as a large number of blocks are created in the HDFS.

Table 6. Total Image Transcoding Time (s) Based on the Block Size

Block Replication	Image Dataset Size							
	1 GB	2 GB	4 GB	8 GB	10 GB	20 GB	40 GB	50 GB
16 MB	30	51	70	155	138	349	732	902
32 MB	23	30	49	84	102	185	356	437
64 MB	23	32	49	84	102	186	356	428
128 MB	23	31	50	84	102	185	356	436
256 MB	24	30	49	85	103	185	356	436
512 MB	23	32	50	85	102	185	356	436

6.4. Image Transcoding Performance Based on Changes in the apred.tasktracker.map.tasks.maximum Option

In the fifth set of experiments, we focused on exploring and analyzing different values for mapred.tasktracker.map.tasks.maximum. This option represents the maximum number of map tasks performed simultaneously in a single data node.

Before performing this set of experiments, we expected that the transcoding job performance for the maximum number of map slots would depend on the number of CPUs in the physical machine, i.e., if the value is set to 4, the four map tasks used to process the MapReduce job are performed simultaneously in a single data node. It was expected that a value of 8 would exhibit a better performance than the other values. In fact, from the experimental results shown in Figure 8, the best transcoding performance is achieved when the value of this option is set to 8, because our system has eight CPUs in each node.

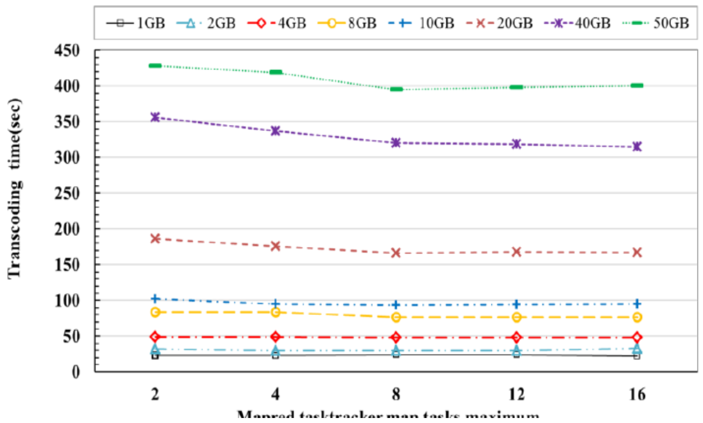


Figure 8. Total Transcoding Time in Hadoop for 5, 10, and 20 GB Datasets for Various Values of mapred.tasktracker.map.tasks.maximum

6.5. Image Transcoding Performance for Resizing an Image Dataset

For this performance evaluation, we measured the total transcoding times for image resizing. The transcoding times required to encode an original image dataset into QVGA (320 × 240), VGA (640 × 480), and WVGA (800 × 408) resolutions were measured. The performance results are shown in Figure 9. We can clearly see from the results that there is no difference in the transcoding performance for resizing an image dataset.

7. CONCLUSION

This paper introduced an efficient multimedia processing for SNS multimedia service using cloud computing system, more precisely Hadoop system. To do this, PESMS was designed; this provides an environment of the developing, building of Social service by adopting cloud computing technologies and elastic computer resources. And this study presented an image conversion module for transcoding and transmoding based on MapReduce running on an HDFS using PESMS. The aim for proposing and implementing the image conversion module using PESMS was to process lots of social multimedia with more efficiency.

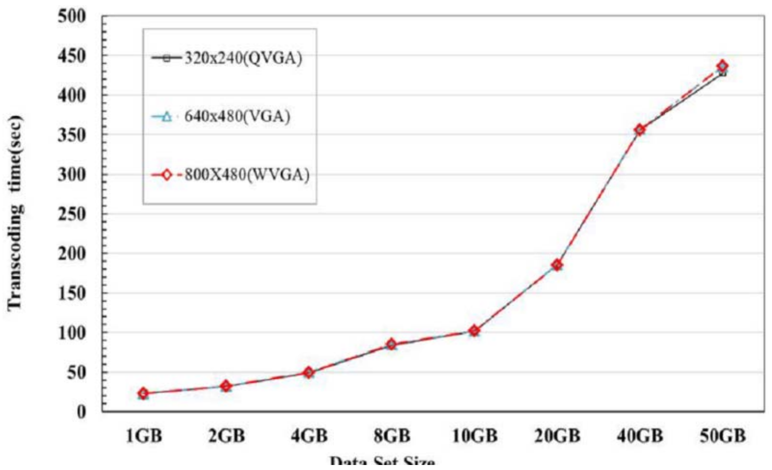


Figure 9. Total Image Transcoding Time for Resizing an Image Resolution

We verified the excellent performance of the image conversion module using varying experimental conditions conducted on a 28-node test bed. In fact, we measured the run times for an image conversion under seven sets of experiments: changes in the cluster size, block replication factor, block size, JVM reuse factor, mapred.tasktracker.map.tasks.maximum option, resizing function, and conversion formats. Our conversion module can reduce the run times for converting image datasets into specific formats suitable for various devices. In particular, based on the experimental results on the changes in the Hadoop options, we can see that MapReduce programmers should carefully consider selecting options related with the block size and block replication, JVM reuse option, and mapred.tasktracker.map.tasks.maximum option depending on the form and size of the datasets, the programming techniques, the business logic, and the configuration of the cluster systems.

Finally, due to PESMS that is PaaS platform, service provider or developer could easily develop social multimedia service without complex installation. They only have to consider application. We can conclude that PESMS is worthy for a large social multimedia data processing.

ACKNOWLEDGEMENTS

This work was supported by the IT R&D program of MSIP/IITP. [B0101-15-0559, Developing On-line Open Platform to Provide Local-business Strategy Analysis and User-targeting Visual Advertisement Materials for Micro-enterprise Managers] and MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1004) supervised by the NIPA(National IT Industry Promotion Agency.

REFERENCES

[1] Kim M., Lee H., Lee H., SMCCSE: PaaS Platform for processing large amounts of social media, The 3rd International Conference Internet, Malaysia, (2011), 631-635.
[2] Kim M., Lee H., SMCC: Social Media Cloud Computing Model for Developing SNS based on Social Media, Springer Communications in Computer and Information Science, 206(2011), 259-266.
[3] Golbeck J., Robles C., Turner K., Predicting Personality with social media, Conference on Human Factors in Computing Systems, Vancouver, (2011) , 253-262.
[4] Wikipedia,http://en.wikipedia.org/wiki/Social_media, 2011.2
[5] Kaplan A.M., Haenlein M., Users of the world, unite! The challenges and opportunities of Social Medial, Journal of business horizons, 53(2010), 59-68.
[6] Kim Hak J., Online Social Media Networking and Assessing Its Security Risks, International Journal of Security and Its Applications, 6(2012), 11 – 18.
[7] Peter Mell, Timothy Grance, The NIST Definition of Cloud Computing, Special Publication 800-145, September 2011
[8] http://www-01.ibm.com/software/data/infosphere/hadoop/
[9] Li X., Shi Y., Guo Y., Ma W., Multi-Tenancy Based Access Control in Cloud, 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, (2010),1-4.

[10] Ouirdi M.E., Ei A., Segers J., Henderickx, E, Social Media Conceptualization and Taxonomy: A Lasswellian Framework, Journal of Creative Comm., 9(2014), 107-126.
[11] Scheepers H., Scheepers R., Stockdale R., Nurdin, N., The dependent variable in social media use, Journal of Computer Information Systems, 54(2012), 25-34.
[12] Apache HBase, http://hbase.apache.org/.
[13] Wikipedia, ttp://en.wikipedia.org/wiki/Apache_ Hadoop , 2011.
[14] Diaz-Sanchez D., Almenares F., Marin A., Proserpio D., Media Cloud: Sharing Contents in the Large, 2011 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, (2011), 227-228.
[15] Wang L., Von Laszewski G., Young A., He X., Kunze M., Tao J., Fu C., Cloud Computing: A Perspective Study, New Generation Computing, 28(2010) 137-146.
[16] Kim M., Han S., Cui Y., Lee H., CloudDMSS: robust Hadoop-based multimedia streaming service architecture for a cloud computing environment, Cluster Computing, (2014),In Press.
[17] Xu Y., Mao S., A survey of mobile cloud computing for rich media applications, IEEE Wireless Communications, 20(2013), 46-53.
[18] Xen, ttp://www.xen.org/support/documentation.html.
[19] Shvachko K., Kuang H., Radia S., Chansler R., The Hadoop Distributed File System, 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, (2010), 1-10
[20] The Hadoop Distributed File System: Architecture and Design, http://hadoop.apache.org/common/docs/r0.18.0/hdfs_ design.pdf.
[21] Lee H., Kim M., Her J., Lee H., Implementation of MapReduce-based Image Conversion Module in Cloud Computing Environment, International conference on Information Networking, (2012), 234-238.

BIOGRAPHIES OF AUTHORS

Jongin Jung received an M.S. degree from Sung Kyun Kwan University, Seoul, Korea, in 2002. Currently, he is a Ph.D. student in the department of Internet and Multimedia Engineering at the KonKuk University, and he is a team leader of Smart Life Service Developtment part in Smart Media Research Center of Korea Technology Institute. He is interested in the cloud computing, big data, Hadoop.

Myoungjin Kim received an M.S. degree from Konkuk University, Seoul, Korea, in 2009. Currently, he is a candidate of Ph.D. in the department of Internet and Multimedia Engineering at the same university, and he is also an assistant researcher at the Social Media Cloud Computing Research Center. His research interests include distributed computing, distributed realtime programming, MapReduce, media transcoding and cloud computing.

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