Performance Evaluation of Image Conversion Module Based on MapReduce for Transcoding and Transmoding in SMCCSE

Myoungjin Kim
Division of Internet & Multimedia Engineering
Konkuk University
Seoul, Korea
tough105@konkuk.ac.kr

Hanku Lee* and Yun Cui
Division of Internet & Multimedia Engineering
Konkuk University
Seoul, Korea
{hlee, ilycy}@konkuk.ac.kr

Abstract—We presented the author’s SMCCSE (Social Media Cloud Computing Service Environment) that supports the development and construction of Social Networking Service (SNS) based on large amounts of social media including audio, video and image in earlier publication. The main contributions of this paper are to present how to design and implement MapReduce-based image conversion module for social media transcoding and transmoding that are the core functions in our SMCCSE as well as to verify performance evaluation results for image conversion module. In this paper, we show a partially functional image conversion module based on Hadoop in SMCCSE except for video and audio. Moreover, we discuss experimental results performed on a 28 node SMCCSE cluster under varying experimental conditions in order to verify the performance for image conversion function.

Keywords—Cloud Services; Cloud Platform; Hadoop; MapReduce; Image Conversion; Social Networking Services; Social Media; Transmoding and Transcoding

I. INTRODUCTION

Recently, the cloud computing has been gaining lots of interests from researchers and IT industry due to many its abilities to provide flexible dynamic IT infrastructures, QoS guaranteed computing environments, configurable software services and the state-of-the-art enabling cloud computing technologies [11]. In particular, lots of service providers and developers related with web-based services like portal service, web searching service and SNS are tending to apply cloud computing environments to their fields. The reason why they make an effort to take advantage of cloud computing is that they handle large amounts of media data such as audio, images and videos.

Social Networking Services (SNS) standing in the center of this era produce a huge amount of social data including media data daily. As a matter of fact, the amount of data in Twitter users produces every day reaches up 7TB and even Facebook also produces 10TB. However, this big data created by SNSs’ users brings several difficulties to service vendors. The first one is that service vendors are faced with the limitation of storage capacity the traditional storage systems have. The second one is that they are also confronted with difficulties in applying transmission techniques and data processing to transmit large amounts of data to end users’ devices such as smart phone, TV and smart pad. To solve these problems, new storage system and elastic computing resources and big data processing techniques for processing and storing large amounts of data are required to support the development environments of SNS.

For these, we provided and described a new concept of cloud computing service platform that is SMCCSE (Social Media Cloud Computing Service Environment) for SNS in the earlier publication [2,4]. The main contribution of SMCCSE is to support development environments for SNS based on social media [1,7,18] using enabling cloud computing techniques and elastic computing resources on cloud computing environment. In this paper, we are focusing on the implementation of SMCCSE PaaS platform as a key platform. Especially, we designed and implemented partially functional social media (audio, videos and images) conversion modules that are the most important functions for transcoding and transmoding in our SMCCSE. The main objectives of this paper are to explain how to convert large amounts of image datasets except for videos and audio data in MapReduce-based image conversion module of PaaS platform and to show performance results for image conversion function under a variety of experimental conditions. In fact, we evaluated 5 kinds of experiments.

The rest of the paper is organized as follows. Section 2 briefly introduces an overview of SMCCSE. In the next section, we explain SMCCSE as a service platform and image conversion module based on Hadoop [3]. Section 4 discusses performance evaluation and experimental results. Section 5 introduces the concept of cloud computing and Hadoop as related works. In the last section, we conclude our paper and present the directions of future work.

II. OVERVIEW OF SOCIAL MEDIA CLOUD COMPUTING

A. Social Media Cloud Computing Service Model

Our service model is multiple service models combining with cloud computing that supports to develop SNss such as Twitter and Facebook, social media service such as YouTube and social game service like the social network game of Facebook. Firstly, our service model offers social media APIs, social SDK based on Web and service delivery platform to easily develop SNS as the form of SaaS. Secondly, in order to provide social media data with reliable services to users, it also presents distributed and parallel data processing platform that deals with large social data (audio, video, and picture) for storing, distributing and en/decoding
them as the form of PaaS. Lastly, it provides IaaS based on virtualization to reduce the cost associated with building computing resources such as server, storage and so on. Figure 1 shows the concept of Social Media Cloud Computing Model.

B. Overall Architecture of SMCCSE

In this section, to implement our model on real cloud computing environment, we present SMCCSE architecture in detail. Fig 2 illustrates the architecture of SMCCSE. The general idea of designing SMCCSE is to establish an environment supporting the development of SNS and addressing of numerous SNSs, to provide the approaches of processing big social media data and to provide a set of mechanisms to manage Infrastructure.

SMCCSE is largely divided into 3 layers: SaaS layer, PaaS layer and IaaS layer. Furthermore, 3 layers are composed of 8 parts: Social Media Service Platform, Social Media Common Algorithms Library, Distributed Processing Platform, Cloud Security, Cloud QoS, Green IDC, Infra Management and Virtualization.

C. SMCCSE as a SaaS Platform

The main role of SMCCSE SaaS Platform is to provide cloud Soft as a Service that users are able to interact with social media created by other users. That is to say, it is possible for users to publish, share, and distribute user-created contents using our SMCCSE SaaS platform. Our SaaS platform is composed of multi-tenancy, SDK social APIs, service delivery platform and DLNA (Digital Living Network Alliance). The features of each part are follows as:

- **Multi-tenancy.** SaaS platform plays a role in providing SaaS service based on open solutions possible to guarantee a multi-tenancy service mode. The concept of multi-tenancy is a critical issue in cloud computing because it is directly related to security and QoS in the aspect of companies and individual. Although it still faces a big challenge of security and privacy problem[5], secured multi-tenancy should be applied in cloud computing environments to reduce cost correlated with building computing resources, especially storage resource and to effectively manage infrastructure. With the importance of multi-tenancy we apply secured multi-tenancy to our SaaS platform so that users can share and publish countless social media that is stored in a single cloud system.

- **Web Development Environment.** Users or developers can make new services based on social media generated by participants in social sites using UI components, Service Components and a set of development tools such as social media APIs, SDK based on Web and SDK for application that SMCCSE provides.

- **Service Delivery Platform.** It also contains service delivery platform that can deploy and develop new converged multimedia services quickly on a variety of smart devices such as Apple’s iPhone and Google smart phones running on Android platform.

- **UPnP and DLNA.** It includes UPnP (Universal Plug and Play) and DLNA (Digital Living Network Alliance) that deal with interoperability between networked consumer devices permitting user-generated contents to be shared among household devices [10]. DLNA technology has a merit of easily sharing data among heterogeneous devices such as TV, home theaters, smart phones, DVD players, game consoles (PS3, Xbox360), cameras and set-top boxes. UPnP (User Plug and Play) technology is architecture to connect especially home PCs and smart devices or phones to a peer-to-peer network, and this technology makes various different smart devices automatically connect with each other and helps different users use various kinds of services through networking. With such technologies, SNS users can share, distribute and use countless social media data created by social media participants at hundreds of digital devices recently released.
In addition to these main functions, SaaS platform also provides MapReduce development environment to easily develop programing framework to distribute and process large amounts of social media in distributed processing systems.

III. SMCCSE AS A SERVICE PLATFORM

A. SMCCSE PaaS Platform

This section explains PaaS platform likely to be the core platform of SMCCSE and IaaS functioning to provide physical computing environments. Figure 3 shows the whole architecture of PaaS platform and IaaS.

PaaS platform consists of Social Media Data Analysis Platform (Social Common Algorithms Library), Cloud Distributed and Data Processing Platform and Cloud Infra Management Platform. Firstly, the main functions of Social Media Data Analysis Platform are to analyze usage pattern, relationship between users and social media data on demand, and to provide the functions of encoding, decoding, transcoding and transmoding as the form of libraries. Transcoding means converting one media file into files suitable for numerous digital devices in terms of file forms. Transmoding means converting one media file into files in terms of file size. Secondly, Cloud Distributed and Parallel Data Processing Platform as a core platform of our model is able to store, distribute and process social media data created by users by applying HDFS (Hadoop Distributed File System), MapReduce and HBase (Hadoop Database System) to its system. Lastly, Cloud Infra Management Platform contains the concepts of cloud QoS, Green IDC and Cloud Infra Management. Cloud Infra management manage and monitor computing resources that do not depend on specific OS or Platform. It includes the functions of resource scheduling, resource information management, resource monitoring and virtual machine management. These functions are provided on web service based on Eucalyptus. In addition, our IaaS is designed to offer elastic computing resources including servers, storage and bandwidth using virtualization techniques based on Xen [12].

B. Image Conversion Module in PaaS Platform

The most important key function of PaaS platform in SMCCSE is image and video conversion for transcoding and transmoding in order to transmit large amounts of social media created by users in SNS to end users’ devices. In particular, cloud distributed and parallel data processing platform in PaaS platform practically carries out social media data conversion function. For this, we have designed and implemented partially functional image conversion module based on Hadoop except for video conversion function. Figure 4 illustrates the architecture of image conversion module.

The processes to conduct transcoding and transmoding using Hadoop are as follows. First of all, user created image data is automatically distributed and stored in each node running on HDFS [16]. Afterwards MapReduce performs the batch processing to convert stored image datasets on HDFS. Our conversion module has only Map step due to the fact that it is not necessary to conduct merging processing for results by Reduce step. Map function is implemented by SequenceFiles Method in the Map phase. With SequenceFiles, Maps inputs file names as key and file contents themselves as value pairs into a set of intermediate key /value pairs. Image conversion module sets file contents
into byte type using BytesWritable class. Lastly, image datasets is processed in parallel on each node.

C. Prototypes

Figure 5 shows the prototype of image conversion module of SMCCSE. Our converter are able to convert image dataset created and uploaded by users in SNS into proper file size and format suitable for users’ devices in real-time using elastic computing resources, HDFS and MapReduce programming. Image conversion module can selects a variety of options such as scale, resolution, size and target formats for image conversion as well as MapReduce options such as number of nodes, number of Map tasks, memory size and so on.

IV. PERFORMANCE EVALUATION

A. Configuration of Experiments

The experiments are conducted on a 28-node test bed. Our test bed is a single enterprise scale cluster that consists 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 are running on Linux OS (CentOS 5.5). Each node is equipped with two Intel Xeon 4 core 2.13GHz with 4GB registered ECC DDR memory and 1TB SATA-2 (7200RPM HDD). The machines are interconnected with 1000Mbps Ethernet adapter.

The experimental environments except for hardware specification are as follows.

- To build a variety of experimental conditions, we use image datasets (Table 1) of 9 groups. An average size of one image file is approximately 19.8MB.
- To implement an image conversion function, we also use JAI 1.1.3 (Java Advanced Imaging) APIs and Java 1.6.0_23.
- To process huge data datasets on our test bed, we select Hadoop-0.20.2. Selected default options in Hadoop are as follows: 1) the numbers of block replications are 3EA. 2) Block size is 64MB.

<table>
<thead>
<tr>
<th>Size</th>
<th>1GB</th>
<th>2GB</th>
<th>4GB</th>
<th>8GB</th>
<th>10GB</th>
<th>20GB</th>
<th>40GB</th>
<th>50GB</th>
<th>100GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of files</td>
<td>52</td>
<td>104</td>
<td>208</td>
<td>416</td>
<td>520</td>
<td>1040</td>
<td>2111</td>
<td>2594</td>
<td>5188</td>
</tr>
</tbody>
</table>

B. Experimental Results

We conduct several experiments for the performance evaluation of partially implemented MapReduce-based cloud distributed and parallel data processing platform in SMCCSE. The experiments are chosen to provide an overview of transcoding and transmoding functions. Especially, we measure run time for image conversion function based on MapReduce to convert large amounts of image datasets into specific formats (from JPG to PNG) suitable for a variety of mobile devices under a variety of conditions.

The objective of the first experiment is to measure the run times and speedup for image conversion function under varying cluster size. According to Figure 6, it is known that the run times decrease when the numbers of nodes increase. In particular, the elapsed times decrease dramatically until 8 nodes. From 8 nodes to 28 nodes, the run times are reduced gradually. In addition, we also measure parallel speedup. Experiments are run with different numbers of parallel nodes to be able to calculate parallel speedup. Parallel speedup calculates how many times the parallel and distributed execution is faster than running an image converting.
functions implemented by the same MapReduce programming on a single node. If speedup is greater than 1, it means that there is at least some gain from carrying out the work in parallel. If speedup is the same as the number of machines, which means that our cloud server and MapReduce programming has a perfect scalability and also has ideal performance. Calculated speedup is shown on Figure 8. As known from the results, in 2, 4 and 8 nodes, there is the best ideal and perfect scalability in parallel. Although, the performance is not optimum since 10 nodes, we know our cloud server has high-performance throughput via distributed processing. Moreover, in case of too large and too small image data sets, we know that throughput in distributed processing performance is reduced in our cloud server. As a matter of fact, calculated speedups of 2GB, 100GB, 10GB and 40GB in 28 nodes are 15.3, 14.2, 23.1 and 20.6.

Table 2. Run times for image conversion in MapReduce under varying cluster size (sec)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>1GB</th>
<th>2GB</th>
<th>4GB</th>
<th>8GB</th>
<th>16GB</th>
<th>28GB</th>
<th>40GB</th>
<th>50GB</th>
<th>100GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>165</td>
<td>322</td>
<td>634</td>
<td>1225</td>
<td>1572</td>
<td>3105</td>
<td>6238</td>
<td>7721</td>
<td>12833</td>
</tr>
<tr>
<td>2 nodes</td>
<td>88</td>
<td>167</td>
<td>322</td>
<td>635</td>
<td>1125</td>
<td>1567</td>
<td>3122</td>
<td>4111</td>
<td>6109</td>
</tr>
<tr>
<td>4 nodes</td>
<td>49</td>
<td>88</td>
<td>165</td>
<td>323</td>
<td>404</td>
<td>784</td>
<td>1523</td>
<td>1712</td>
<td>3226</td>
</tr>
<tr>
<td>8 nodes</td>
<td>25</td>
<td>40</td>
<td>71</td>
<td>134</td>
<td>165</td>
<td>323</td>
<td>622</td>
<td>769</td>
<td>1524</td>
</tr>
<tr>
<td>10 nodes</td>
<td>23</td>
<td>38</td>
<td>69</td>
<td>128</td>
<td>159</td>
<td>317</td>
<td>612</td>
<td>709</td>
<td>1477</td>
</tr>
<tr>
<td>12 nodes</td>
<td>24</td>
<td>36</td>
<td>63</td>
<td>114</td>
<td>141</td>
<td>248</td>
<td>587</td>
<td>654</td>
<td>1313</td>
</tr>
<tr>
<td>16 nodes</td>
<td>21</td>
<td>30</td>
<td>50</td>
<td>88</td>
<td>108</td>
<td>207</td>
<td>544</td>
<td>611</td>
<td>1289</td>
</tr>
<tr>
<td>20 nodes</td>
<td>21</td>
<td>24</td>
<td>37</td>
<td>72</td>
<td>90</td>
<td>204</td>
<td>478</td>
<td>577</td>
<td>1087</td>
</tr>
<tr>
<td>22 nodes</td>
<td>18</td>
<td>24</td>
<td>39</td>
<td>71</td>
<td>81</td>
<td>161</td>
<td>331</td>
<td>509</td>
<td>961</td>
</tr>
<tr>
<td>25 nodes</td>
<td>18</td>
<td>22</td>
<td>36</td>
<td>61</td>
<td>73</td>
<td>160</td>
<td>310</td>
<td>477</td>
<td>898</td>
</tr>
<tr>
<td>28 nodes</td>
<td>17</td>
<td>21</td>
<td>35</td>
<td>51</td>
<td>68</td>
<td>152</td>
<td>302</td>
<td>423</td>
<td>903</td>
</tr>
</tbody>
</table>

Table 3. Speedup for the image under varying cluster size (sec)

<table>
<thead>
<tr>
<th>Size</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>25</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 GB</td>
<td>1.9</td>
<td>3.6</td>
<td>8</td>
<td>8.9</td>
<td>10.7</td>
<td>13.4</td>
<td>14.6</td>
<td>15.3</td>
<td></td>
</tr>
<tr>
<td>10 GB</td>
<td>1.3</td>
<td>3.8</td>
<td>9.5</td>
<td>11.1</td>
<td>14.5</td>
<td>17.4</td>
<td>21.5</td>
<td>23.1</td>
<td></td>
</tr>
<tr>
<td>40 GB</td>
<td>1.9</td>
<td>4</td>
<td>9.4</td>
<td>10.6</td>
<td>11.4</td>
<td>13</td>
<td>20.1</td>
<td>20.6</td>
<td></td>
</tr>
<tr>
<td>100 GB</td>
<td>2.1</td>
<td>3.9</td>
<td>8.4</td>
<td>9.7</td>
<td>9.9</td>
<td>11.8</td>
<td>14.2</td>
<td>14.2</td>
<td></td>
</tr>
</tbody>
</table>

In the second experiment, to demonstrate the excellence of MapReduce programming to implement an image converting function, we compare our cloud server with two machines running on one node. Machine A is equipped with AMD Athlon II-X4 635 2.9GHz, 4GB memory and 600GB running on CentOS5.5 and machine B is the same as the master node's machine. We measure each running time taken in our cloud server using MapReduce programming and taken in machine A and B applying only sequential programing using JAI libraries without MapReduce, respectively. Figure 8 shows the result. The elapsed times in machine A, B are smaller than the running time taken in less than 2 nodes in our cloud server. The reason why the performance in machine A and B without MapReduce is better than our server in case of 1 and 2 nodes is that distributed processing on MapReduce programming causes overhead elements with related to the creation of map tasks, job scheduling and low disk performance.

Figure 6. The elapsed time to convert image under varying cluster size

Figure 7. Speedup for an image converting function with different number of nodes

Figure 8. Run times for our SMCCSE with two different machines

The third experiment is to measure the run times for image conversion function according to the change of the number of block replication. MapReduce splits large amounts of datasets into fixed-size block for a quick search and
processing of them. As a matter of fact, with the default replication value, 3, replicated data is stored on three nodes of HDFS in order to rebalance data, to move copies around and to keep the replication of data high when system faults such as disk failure, network connection problems and so on occur. The purpose of this experiment is to verify how block replication materially affects our performance. The block replication numbers used in experiment are 1, 2, 4 and 5 with default value, 3. The experiment results are shown in figure 9 and table 4. Experimental result indicates that the execution times are reduced, when the block replication numbers increase. Especially, there is no much difference of the execution times in small datasets, from 1GB to 8GB, whereas in case of the larger dataset, the difference of execution times is bigger. In fact, in 100GB dataset, the execution times are 1048sec, 903sec and 726sec in 1, 2 and 3 block replications, respectively.

Table 4. Run times for image conversion under the change of the number of block replication (sec)

<table>
<thead>
<tr>
<th>Numbers</th>
<th>1GB</th>
<th>2GB</th>
<th>4GB</th>
<th>8GB</th>
<th>10GB</th>
<th>20GB</th>
<th>40GB</th>
<th>50GB</th>
<th>100GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>29</td>
<td>48</td>
<td>77</td>
<td>103</td>
<td>442</td>
<td>426</td>
<td>529</td>
<td>1048</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>33</td>
<td>41</td>
<td>60</td>
<td>77</td>
<td>227</td>
<td>398</td>
<td>472</td>
<td>952</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>30</td>
<td>35</td>
<td>51</td>
<td>68</td>
<td>178</td>
<td>349</td>
<td>423</td>
<td>903</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>21</td>
<td>33</td>
<td>56</td>
<td>67</td>
<td>136</td>
<td>274</td>
<td>402</td>
<td>742</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>19</td>
<td>29</td>
<td>50</td>
<td>59</td>
<td>126</td>
<td>262</td>
<td>388</td>
<td>726</td>
</tr>
</tbody>
</table>

The purpose of the fourth experiment is to measure the execution times according to block size. Basically, Hadoop performs to process huge amounts of datasets after splitting datasets into default block size value, 64MB. We measure the run times for image conversion function in 16, 32, 64, 128 and 256 MB block size, respectively. The result is shown in table 6 and figure 10. According to the results, in case of 64MB block size, the performance of run time is the best among 5cases. If developer sets block size in Hadoop smaller than the file size (our case is approximately 20MB) included in an image dataset, the execution times increases because lots of blocks are created on HDFS. Besides, if block size is set bigger than the file size included in an image dataset, the run times also increase due to the overhead related with processing and searching unnecessary space.

Table 6. Run times for image conversion under the varying block size

<table>
<thead>
<tr>
<th>Block size</th>
<th>1GB</th>
<th>2GB</th>
<th>4GB</th>
<th>8GB</th>
<th>10GB</th>
<th>50GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>16MB</td>
<td>30</td>
<td>37</td>
<td>48</td>
<td>83</td>
<td>107</td>
<td>548</td>
</tr>
<tr>
<td>32MB</td>
<td>30</td>
<td>36</td>
<td>43</td>
<td>77</td>
<td>101</td>
<td>482</td>
</tr>
<tr>
<td>64MB</td>
<td>30</td>
<td>35</td>
<td>41</td>
<td>70</td>
<td>92</td>
<td>452</td>
</tr>
<tr>
<td>128MB</td>
<td>29</td>
<td>37</td>
<td>42</td>
<td>81</td>
<td>120</td>
<td>465</td>
</tr>
<tr>
<td>256MB</td>
<td>42</td>
<td>37</td>
<td>45</td>
<td>79</td>
<td>116</td>
<td>478</td>
</tr>
</tbody>
</table>
In the last experiment, the run times to convert JPG format into PNG, BMP and TIFF are measured. Datasets used in evaluation are 2,4,8,10,50,100 GB respectively. As shown from experimental results, we know that there is no much difference in the converting time according to converting original format into target formats.

Table 7. Run times for image converting according to target formats

<table>
<thead>
<tr>
<th>Format</th>
<th>JPG to PNG</th>
<th>JPG to BMP</th>
<th>JPG to TIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1GB</td>
<td>17</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>2GB</td>
<td>30</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>4GB</td>
<td>35</td>
<td>32</td>
<td>31</td>
</tr>
<tr>
<td>8GB</td>
<td>51</td>
<td>52</td>
<td>53</td>
</tr>
<tr>
<td>10GB</td>
<td>68</td>
<td>72</td>
<td>71</td>
</tr>
<tr>
<td>20GB</td>
<td>178</td>
<td>201</td>
<td>178</td>
</tr>
<tr>
<td>40GB</td>
<td>349</td>
<td>341</td>
<td>320</td>
</tr>
<tr>
<td>50GB</td>
<td>423</td>
<td>421</td>
<td>421</td>
</tr>
<tr>
<td>100GB</td>
<td>903</td>
<td>889</td>
<td>920</td>
</tr>
</tbody>
</table>

Figure 11. Run times for an image converting according to target formats

V. RELATED WORKS

A. The Concept of Cloud Computing

The cloud computing was born in complex combination from Distributed computing, Grid computing, Utility computing and so on [14]. Many people define cloud computing [13,17,19,21] as an emerging computing paradigm where data and services reside in massively scalable data centers and can be ubiquitously accessed from any connected devices over the internet. That is to say, users are able to obtain useful information they want and use a variety of services and computing resources through the internet as a cloud without the constraints of time and place.

Also, the NIST[20] (National Institute of Standards and Technology) defines ‘On-demand self-service’ , ‘Broad network access’, ‘Resource Pooling’, ‘Rapid elasticity’ and ‘Measured Service’ as the five essential characteristics of cloud computing. In addition, service models are classified into 3 service models: Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS) and Cloud Infrastructure as a Service (IaaS). Lastly, deployment models are classified into 3 models: Private Cloud, Public Cloud and Hybrid Cloud [8,11].

According to Ian Foster, there are three main factors contributing to the surge and interests in cloud computing: 1) rapid decrease in hardware cost and increase in computing power and storage capacity, and the advent of multi-core architecture and modern supercomputers consisting of hundreds of thousands of cores; 2) the exponentially growing data size in scientific instrumentation / simulation and Internet publishing and archiving; and 3) the wide-spread adoption of Services computing and Web 2.0 applications [13].

B. Hadoop

Hadoop inspired by Google’s MapReduce and Google File System [6] is a software framework which supports data-intensive distributed applications working with thousands of nodes and petabytes of data [3,15]. Hadoop has the ability to carry out scalable and timely analytical processing of big datasets to extract useful information. Hadoop consists of two important frameworks: 1) Hadoop Distributed File System (HDFS) like GFS is distributed, scalable and portable file system written in Java 2) MapReduce is the first developed by Google for huge data processing.

MapReduce frameworks provide a specific programming model and a run-time system for processing and creating large amounts of datasets which is amenable to various real-world tasks [9]. MapReduce framework also handles automatic scheduling, communication, synchronization for processing huge datasets and it has the ability related with fault tolerance. MapReduce programming model is executed in two main steps, called mapping and reducing. Mapping and reducing are defined by mapper and reducer functions that are s data processing functions. Each phase has a list of key and values pairs as input and output. In the mapping, MapReduce input datasets and then feeds each data element to the mapper as a form of key and value pairs. In the reducing, all the outputs from the mapper are processed and a final result is created by reducer with merging process.

![Figure 12. The procedure of MapReduce](image-url)
VI. CONCLUSION AND FUTURE WORKS

In this paper, we briefly review our SMCCSE (Social Media Cloud Computing Service Environment) that supports developing and building SNSs based on social media by adopting enabling cloud computing technologies and elastic computer resources. Moreover, this study presents image conversion module for transcoding and transmoding based on MapReduce running on HDFS in SMCCSE. The motivation of proposing and implementing image conversion module of our SMCCSE is the requirement to enhance throughput of transcoding and transmoding systems to cope with the physical limitation of device’s hardware and server platforms. This study first introduce a new concept of cloud computing service model combining cloud computing and SNS as well as image conversion module adopting MapReduce for providing QoS-guaranteed SNSs and processing huge image datasets.

In addition, we have verified excellence in performance of image conversion function with varying experimental conditions conducted on a 28 node test bed. In fact, we measure the run times for image conversion function under varying cluster size, block replicas, block size and target formats. Our conversion module is able to reduce the run times to convert image datasets into specific formats suitable for various devices. Especially, through the experimental results conducted on the change of block size and replica, we know that MapReduce programmers should carefully consider to select options related with block size and block replication depending on the form and size of datasets, programming techniques business logic and the configuration of cluster systems.

For the future works, we are going to focus on building our SMCCSE in real cloud environment. In addition, to improve image, video and audio conversion functions for transcoding and transmoding, we will implement and design fully functional modules running on SMCCSE.

ACKNOWLEDGMENT

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency (NIPA-2011 – (C1090-1101-0008)).

REFERENCES