# A Comparative Study for Optimization of Video File Compression in Cloud Environment

Navdeep S. Chahal
Senior Engineer
Centre for Development of Advanced Computing,
(C-DAC) Mohali, Punjab, India

Baljit S. Khehra
Associate Professor
Baba Banda Singh Bahadhur Engineering College,
Fatehgarh Sahib, Punjab, India

## ABSTRACT
Many organizations like hospitals for telemedicine, journalism for live-telecast and academias are using a service video-on-demand for delivering the lectures and research contents to the remote locations across the globe. The videos to be broadcasted are time and resource consuming due to the large amount of data and due to these constraints, for getting fast access over Internet and mobile devices, such video applications need to be compressed into another format. The usage of videos is occasional so to save huge infrastructure cost and time, the Infrastructure as a Service (IaaS) Cloud systems can be leveraged. In this paper, an attempt has been made to design, implement and optimize the performance of Digital Video to MPEG4 transcoding in the Cloud environment using Meghdoot (an Open-Source Cloud stack). The classical MapReduce approach is used to rationalize the use of resources by exploring on demand computing and performs parallel video conversion thereby reducing the video encoding times. Experimental results point out to suitability of better performance that by varying the technique of splitting the video file size of fragments that is through Mencoder and through default Hadoop Splitting. The comparison of both the systems to get the best compression times will help us to optimize the Cloud resources that further helps in trade-off between time, cost and quality.

## General Terms
Live telecast, Performance, transcoding

## Keywords
Cloud Computing, Video Compression, Meghdoot, MapReduce, Hadoop

## 1. INTRODUCTION
Large infrastructure investment should be required for delivery of video contents to the end-users where time is considered to be crucial. Moreover in some cases the criticality of the speed of publication is a major issue. When some short breaking news needs to be put across, there will be disruption if the compression and encoding time is not considered. The expense of large storage devices for the preprocessing is characterized by its great cost and little flexibility, as a user-specific need. The problem becomes more critical when the volume of information to be processed is variable, i.e., there is a seasonal variation of demand for processing.

In such situation, the ability to build adaptive systems, capable of using on demand resources provided by Cloud Computing [1] is very interesting. Cloud computing is an Internet based services, where we share some of the services like software, platform, infrastructure, storage, databases to computer or other devices on demand by the users. Services are sold on demand, for a minute/hourly basis, services are fully managed by the providers and consumer need only is a computer and Internet access.

In Cloud computing platform (in Figure 1), physical machines are virtualized, and a large variety of virtual machines (VMs) form a virtual cluster. Offering virtualized resources on demand is known as Infrastructure as a Service (IaaS). To achieve higher levels of resource utilization, techniques such as workload balancing across physical servers and storage frames can be used. Workload balancing is achieved with VM live migration, which migrate virtualized applications between physical resources within a resource pool in a way that is transparent to users and does not interrupt the service provided by the Cloud platform [2].

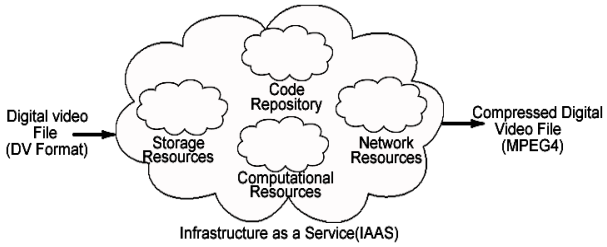


**Figure 1: Cloud Computing Platform**

Video file is splitted into several independent subparts/chunks, distributed among the available nodes, and compressed in parallel. For the same, the Hadoop Distribution File System (HDFS) represents a distributed, scalable and portable file system infrastructure and MapReduce [3] is a programming model designed for simplifying parallel data processing on clusters.

The remainder of this paper is structured as follows. In the next section, we discuss computing on demand, MapReduce paradigm and Meghdoot. In section 3, we introduced the proposed architecture and in the section 4, we discuss the results and compare between the best-optimized time and the acceptable time graphs in a Cloud implementations in section 4. In last section 5, we conclude and discuss further work.

## 2. BACKGROUD
Videos recorded by video cameras, camcoders fall in the category of digital signals recording. The output of compression of the video data would be MPEG4 in our case. Video applications require some form of data compression to facilitate storage and transmission. The process of high quality video encoding is usually very costly to the encoder, and requires a lot of production time. The Infrastructure as a Service (IaaS) paradigm relieves the burden of making huge

investments in infrastructure, and at the same time supports on-the-fly resizing of resources, and adaptation to current needs.

To achieve the video encoding, the video file is broken down into many segments and thereafter is distributed across the VM's in the Open Source Cloud environment. The issues relating to loss in the synchronization while merging the video chunks and optimal distribution need to be considered [3]. A video file is encoded in a specific format through a MapReduce framework for delivering video media content to the end user. In order to observe the compression time the video file is splitted initially using two basic methods of time and size. The effect of these splits on the compression has been experimentally done to prove which of the initial splitting method reduces the compression time more as shown below in Figure 2.

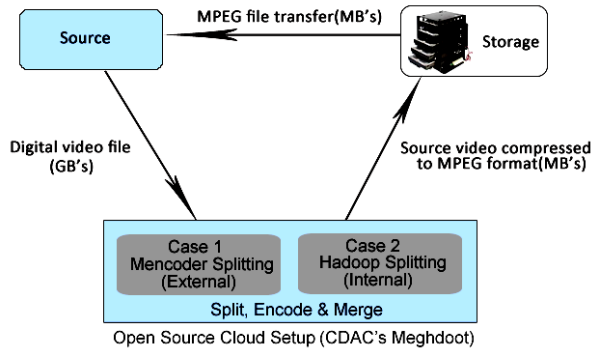


**Figure 2: Basic block Diagram of the proposed work (Experimental Setup)**

Cloud computing uses virtual hardware devices to reduce server power costs and minimize hardware costs. Virtual machine Manager (VMM) or hypervisor is a software/ firmware that creates and runs virtual machines. Cloud Middleware integrates services, applications and content available on the Cloud. In the Cloud setup, Eucalyptus is used as a middleware for the implementation of Cloud computing on computer clusters.

Xen, a VMM or Hypervisor is used which allows several guest operating systems to execute on the same computer hardware concurrently. The complete Cloud setup has been implemented on the Open Source Operating System developed by C-DAC i.e BOSS Linux Server 2.0 (Bharat operating System Solutions) by using Meghdoot Cloud stack. Meghdoot includes Eucalyptus Cloud controller, Eucalyptus Cluster controller, Monitoring Server, Load Balancer, Cloud Elasticity Manager, Web based Cloud Management and Administration Interface.

An attempt has been made to optimize the performance of digital video to MPEG4 conversion using the created Open Source Cloud Infrastructure. The Hadoop, MapReduce approach [4] is used to compare the use of resources by exploring on demand computing and performing parallel video conversion based upon time and size splitting, which reduces the video compression time. The goal is to achieve the best method for the least time, by enabling each virtual node to perform to its best processing potency by exploring the dynamic resource provisioning in the Cloud.

## 3. PROPOSED ARCHITECTURE

FFMpeg [5] and Mencoder [6] are Open Source tools for video splitting and compression. An initial splitter Mencoder or Hadoop (internally) along with the Hadoop distributed file system (HDFS) has been experimented with, where the video file is splitted with chunks based on time slices and size and transcoded on a private Cloud. The effect on the reduction on the compression time with the initial split method based on Mencoder and Hadoop have been experimented with. The experiment to evaluate the performance was conducted on a private Cloud setup based on Meghdoot (Open Source Cloud Stack) with 60 nodes divide into two clusters. The architecture is based upon the assumptions that VMs have similar storage, processing power, memory and network. These are connected through gigabit Ethernet connection. All the cloud nodes have single Intel Pentium core i3 (2.5 GHz to 3.0 GHz) processor and 4GB RAM.

HDFS is designed to run on clusters of commodity machines and Hadoop implements MapReduce, where the application's process is divided into many small fragments of work, each of which may be executed on any node in the cluster i.e. the actual parallel processing [7]. The basic underlying model used after the splitting of the frames is the MapReduce model, which has been coexistent in the HDFS system. In the existing system the Hadoop framework has been pre-installed in all the nodes. These nodes are behaving as virtual machines and in this case they all are bearing the same configuration. The task involved in the MapReduce framework is a two-step process. In the first step the initial file is broken down into fragments and later on submitted to the VMs to work upon them (see Figure 3). In the second step the processed fragments are combined together to obtain the desired output.

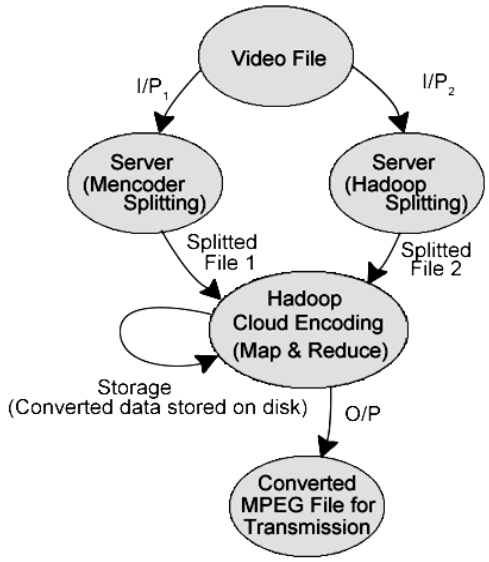


**Figure 3: Experimental State transition diagram for MPEG4 Conversion**

The number of splits based on the split variable (splitted by Mencoder or Hadoop) is compatible with various implementations of audio and video codecs. At the end of the processing step (reduce) all fragments are reconstructed by merging them together. While splitting the file at regular intervals the important point needs to be ensured that no frame coexists in more than one chunk and that no frame is lost.

## 4. RESULTS

The Hadoop is being used as a framework for doing the mapping along with the merging of the compressed video chunks with the split video file as input. An approach to have a split count that is a multiple of the number of Virtual

machines available in a private cloud is obtained by the formula. The cluster is scaled properly, wherein the exact number of VM's is blocked to perform the required task submitted to it. Hence, by using an exact number formula, the Hadoop cluster time blocking is reduced and the next task if in pipeline can be allocated instantly. To find the optimal size of Video chunks (Vc) for Hadoop map function, the following formula has been devised:

$$Vc = (Vi * Rf) / (Vm * Sf) \quad (1)$$

where Vi is the size of the Video input file, Rf is the Replication factor (default is 3) that is the property associated with Hadoop fault tolerance, Vm is the number of Virtual machines used by the Hadoop Clusters and Sf is the scaling factor. For example, if a video file (Vi) is of 2.7 GB (approx. 2760MB), number of VM are 60, replication factor is 3 and the scaling factor fixed to 2, then using eq. (1) the size of Video chunk (Vc) evaluated is 69MB. Hence, we can further calculate the number of Video chunks (Vn)

$$Vn = Vi / Vc \quad (2)$$

Now, using eq. (2), the number of Video chunks (Vn) will be 40. Similarly, if we assume replication factor as 1, then for the same 2.7GB input video file the size of chunks will be 23MB and the number of chunks will be 120.

In a small cluster, the map task creation overhead is considerable. So, dfs.block.size should be large in this case but small enough to utilize all the cluster resources. The block size should be set according to the size of the cluster, map task complexity, map task capacity of the cluster and average size of the input file. The replication factor is set to 1 instead of default 3. The optimal scaling factor (Sf) can also be calculated based upon the number of Vm's in the Cloud cluster using the formula below:

$$Sf = (Vi * Rf) / (Vm * Vc) \quad (3)$$

A private Cloud is setup, along with using the Hadoop MapReduce framework on top of the Cloud and compared the complexities of two cases in terms of time for video file compression. As a proof of concept a simple MapReduce test is implemented and tested on the Cloud to provide an analysis of the distributed computation of MapReduce. The two cases experimented upon are as follows:

Case 1: The file is split with the help of Mencoder (external splitting) and then is being used as a transcoding tool with only the mapping stage. This is then transferred to the HDFS system for the final desired output.

Case 2: The file is being split by Hadoop default splitting (internal splitting) and rests same as Case 1.

The number of chunks were allocated to the desired number of VM's in the Hadoop Map function and the time was computed which was taken for the entire job completion i.e submitting of splitted file to the VMs, transcoding, merging together and giving the output as an MPEG4 converted file.

The experiments were conducted for the above-mentioned cases using different segments i.e. from default splitting size of 64MB to 128MB and time segments of 2Min & 5Min.

**Table 1: Time evaluation for encoding different video files on Hadoop Private Cloud**

| Video File / Video Chunks | 1.2 GB | 2.7 GB | 9.0 GB |
|---|---|---|---|
| 64 MB | 4m 40s | 5m 35s | 10m 11s |
| 128 MB | 10m 46s | 9m 57s | 9m 09s |
| 2 Min | 3m 20s | 5m 10s | 9m 42s |
| 5 Min | 4m 05s | 6m 48s | 8m 30s |

It has been observed that the external segmentation of the video file and further processing by HDFS proved to be faster than the complete involvement of the HDFS system.

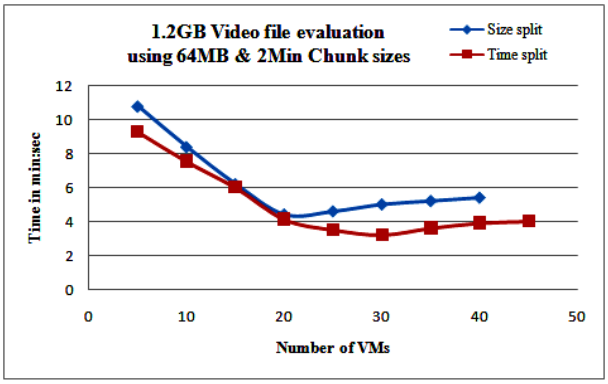


**Figure 4: Time variations for a 1.2GB video file with the selection of different no. of VMs**

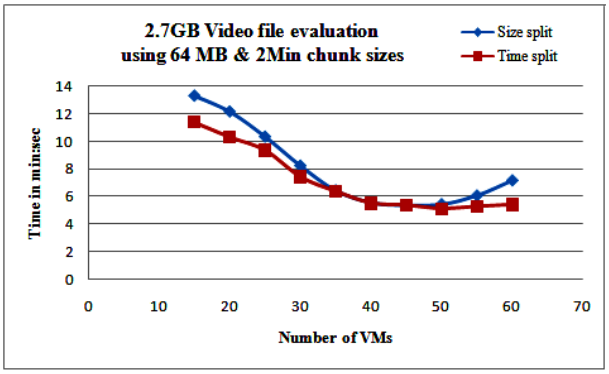


**Figure 5: Time variations for a 2.7 GB video file using different chunks with the selection of different no. of VMs**

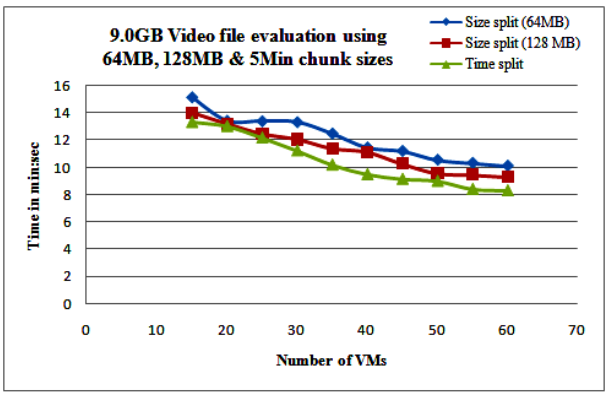


**Figure 6: Time variations for a 9.0 GB video file using size & time chunks with the selection of different no. of VMs**

It was noticed that when the video chunks were made by the Mencoder (external splitter) and the remaining processing was done by Hadoop, time taken to complete the said job was less than if the splits were made internally by Hadoop.

Even as the number of splits is the same for the 9.0GB file the complete Hadoop processing is slower than the one with the external splitting because the complexity involved in the split of the video file is more in Hadoop as depicted in Fig 6.

It is a possibility that the synchronization and separating the audio and video for the digital video file factor has not been dealt with the Hadoop splitting process efficiently is not as good as for the other data types. Mencoder splitting as per the Figures 4-6 proves more efficient than he Hadoop splitting. MapReduce works best on the less number of splits/chunks for a particular task as the complexity reduces.

As the file becomes heavier, the scaling factor also comes into picture where the number of splits is more than the number of VMs available results are better when the segments are of bigger size or segments are made of larger time segments. The scaling factor will have a considerable effect as the number of VMs is not infinite and it restricted in the private Cloud.

The devised formula helps us to find the optimal number of chunks a particular video file should have to be processed in the minimum time. Hence the cost factor can be reduced with the proper selection of number of splits and proper mapping/VMs.

## 5. CONCLUSIONS

The private Clouds typically do not have enough resources to provide the illusion of infinite capacity. Due to the limitation of VM nodes in private Cloud the elastic processing structure could not be produced. We can further optimize these splits, analyzing what is the optimum amount of chunks to be generated, which certainly vary according to the different data types (text, images, etc).

Performance of Hadoop MapReduce jobs can be improved without increasing hardware costs, by tuning several key configuration parameters for cluster specifications, input data size and processing complexity. Lot of research work is still going on to optimize the resources of Cloud computing based upon scheduling, elasticity and scalability. Future work includes the experiments with public Cloud and with different set of inputs.

## 6. ACKNOWLEDGMENTS

We sincerely thank the Cloud R&D team at Centre for Development of Advanced Computing (CDAC), Mohali, India for their valuable contribution and support.

## 7. REFERENCES

[1] Armbrust, M., Fox, M., Griffith, R., et al. (2009) "Above the Clouds: A Berkeley View of Cloud Computing", In: University of California at Berkeley Technical Report no. UCB/EECS-2009-28, pp. 6-7, February 10, 2009

[2] Huaglory Tianfield, Glasgow Caledonian University, "Cloud Computing Architectures", 978-1-4577-0653-0/11, IEEE 2011

[3] Rafael Pereira, Karin Breitman, " An Architecture for Distributed High Performance Video Processing in the Cloud", 3rd International Conference on Cloud Computing, page 482-489, IEEE 2010

[4] Dean, J., Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. In OSDI, 2004

[5] Converting video formats with FFMpeg, Linux Journal archive- Issue 146, June 2006, pp. 10

[6] Mencoder – http://www.mplayerhq.hu

[7] Apache Hadoop - http://hadoop.apache.org/mapreduce/

[8] Shivnath Babu: Towards automatic optimization of MapReduce programs, In: the 1st ACM symposium on Cloud computing, pp.137-142, ACM Press, New York (2010)

[9] Daniel Gmach & Ludmila Cherkasova, HP Labs, Palo Alto, CA (USA) and Jerry Rolia HP Labs, Bristol (UK) "Resource and Virtualization Costs up in the Cloud: Models and Design Choices", 978-1-4244-9233-6/11, IEEE 2011

[10] Rakesh Kumar Jha, Upena D Dalal, "On Demand Cloud Computing Performance Analysis With Low Cost For QoS Application", International Conference on Multimedia, IMPACT-2011, 978-1-4577, IEEE 2011

[11] Rafael Silva Pereira, Karin K. Breitman, "Video processing in the Cloud" SpringerBriefs in Computer Science, ISBN: 978-1-4471-2136-7

[12] Jiann-Liang Chen_z, Szu-Lin Wuy, Yanuarius Teofilus Larosa, Pei-Jia Yang and Yang-Fang Li, " IMS Cloud Computing Architecture for High-Quality Multimedia Applications", 978-1-4577-9538, page 1463-1468, IEEE 2011

[13] Adriana Garcia Kunzel , Hari Kalva , Borko Furht, "A Study of Transcoding on Cloud Environments for Video Content Delivery" 978-1-4503-0168/10, page 13-18, ACM 2010

[14] Hui Kang, Yao Chen, Jennifer L. Wong, "Enhancement of Xen's Scheduler for MapReduce Workloads" 978-1-4503-0552-5/11/06, June 8-11, ACM 2011

[15] Dominique A. Heger, "Optimized Resource Allocation & Task Scheduling Challenges in Cloud Computing Environments", dheger@dhtusa.com

[16] Rajkumar Buyya, James Broberg, Andrzej Goscinski, "Cloud Computing, Principles and Paradigms, Wiley 2011, ISBN: 978-0-470-88799-8, page 123-151

[17] Hai Zhong, Kun Tao, Xuejie Zhang, "An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems", 978-0-7695-4106-8/10, page 124-129, IEEE 2010

[18] Venkatesa Kumar, V., S. Palaniswami "A Dynamic Resource Allocation Method for Parallel Data Processing in Cloud Computing" ISSN 1549-3636, page 780-788, Journal of Computer Science 8 (5), 2012