A Study of Transcoding on Cloud Environments for Video Content Delivery

Adriana Garcia Kunzel
Department of Computer & Electrical Engineering & Computer Science
Florida Atlantic University
777 Glades Road
Boca Raton, FL 33431, USA
+1 (954) 401-9647
agarci53@fau.edu

Hari Kalva
Department of Computer & Electrical Engineering & Computer Science
Florida Atlantic University
777 Glades Road - Bldg 43/422
Boca Raton, FL 33431, USA
+1 (561) 297-0511
hari@cse.fau.edu

Borko Furht
Department of Computer & Electrical Engineering & Computer Science
Florida Atlantic University
777 Glades Road - Bldg 43
Boca Raton, FL 33431, USA
+1 (561) 297-2800
borko@cse.fau.edu

ABSTRACT
Media content distribution poses different challenges when it comes to supplying the variety of existing media formats requested by end-users. The need for an on demand transcoding stage may be justified under specific circumstances but is hampered by the high latency and long processing times. The great power cloud computing offers can be leveraged in a situation like this. This paper analyzes a use case where a Hadoop-based computer cloud is used as an aid in the transcoding of media content. The results point to the suitability of such a resource in a media distribution process requiring low cost streaming, such as the HTTP Live streaming scenario which supports streaming of audio or video to the iPhone, iPad, iPod touch and desktop computers as well as the transfer of video on demand with encryption and authentication.

Categories and Subject Descriptors
C.1.4 [**Parallel Architectures**]: Distributed architecture; C.4 [**Performance of Systems**]; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Video;*

General Terms
Measurement, Performance, Experimentation,Verification.

Keywords
Cloud computing, video transcoding, video distribution, Hadoop

1. INTRODUCTION
In order to deliver media content to end-users, servers need availability of the requested media format. This can be done at the expense of large storage devices for the preprocessing and saving of all the download possibilities. However, this approach is characterized by its great cost and little flexibility as user-specific needs may still not be met. The on-demand processing of video content supported by server caching options is an attractive alternative to support multiple client types that is, however, particularly challenged by the long processing time required to complete an initial request. The exploration of cloud computing approaches to leverage large processing power of less expensive or on-demand hardware for this purpose could increase the speed at which content is available to the end-user after a transcoding process.

Apple's HTTP Live Streaming Protocol is an example, where the speed of content delivery is tightly bound to the speed of an encoding processing stage. The HTTP Live streaming architecture has 3 parts, namely encoding, distribution and client. The encoder takes the input stream from a video source and transforms it into an MPEG2 transport stream, dividing it into 10 second segments during or after the encoding stage. The content is finally delivered using this transport stream segments accompanied by a playlist file (M3U8). These files are stored on a web server and accessed by client using compatible player software or web browser [19].

In this study, the usage of Hadoop streaming jobs as a way to harness the power of the computer cloud to speed-up server content availability is explored in a particular use case scenario; namely when a video playlist, in this case of a DVD content, is made available to users who have a transcoding requirement and can access only smaller sized displays, like mobile users (they have a resolution reduction requirement). Three different experiments are performed and conclusions are drawn which identify different scenarios where each approach may or may not be adequate.

The paper is organized as follows: Sections 2 and 3 review basic concepts and related work, section 4 describes the use case in detail and section 5 discusses the results obtained. Finally, the conclusions are presented and future work is proposed in section 6.

2. RELATED CONCEPTS
2.1. Cloud computing
There is no universal definition of cloud computing as of yet. Cloud computing is a buzzword, an umbrella term applied to several trends in information technology [8].

However, several characteristics are common to them, including the presence of multiple distributed computing resources that are made available to the end consumer which are greater and less

MCMC'10, October 29, 2010, Firenze, Italy.

costly than those provided by a single machine. “Computing in the “cloud” eludes to ubiquitous and inexhaustible on-demand IT resources accessible through the Internet” [8].
This availability can be achieved on a service-based type of interface where the users can make use of the available resources in an on-demand manner, reducing the implementation and maintenance cost to a single, usage based bill. Fueled by this “utility-computing” concept where computing resources are consumed the same way as electricity would, huge commercial endeavors like Amazon Web Services were born.
In addition, cloud computing has become an integral part of the operations at Google and Yahoo and Facebook, among others, processing vast amount of data (like search logs for example) in a reduced time frame.

2.2. MapReduce

Leveraging the power of distributed computing is a challenge that is taken on by Google with MapReduce.
“MapReduce is a programming model and an associated implementation for processing and generating large data sets. [In MapReduce] Users specify a map function that processes a key/value pair [of data] to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key” [1].
MapReduce’s value lies in the possibility of automatic parallelization of activities and their distributed execution thanks to the independence existing between all mapper and reducer tasks started. However, this implies that the problem to be solved through MapReduce needs to be expressible in its specific model. Common tasks that conform to it are log analysis, like web search history analysis where searches may be mapped to IPs, sorted by times and topics, among others. As a result, eventually they may be related to other user’s similar searches and smart suggestions may be given to user while browsing.
In MapReduce, the run-time system takes care of the details of partitioning the large input data, scheduling the program's execution across the machines and handling hardware failures. This allows developers to concentrate on the task at hand, which may be simple (although involving large amounts of data) and forget about underlying machine issues.

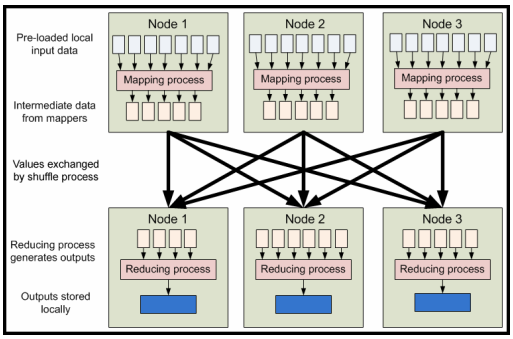


Figure 1. The MapReduce pipeline [5]

2.3. Hadoop

“Hadoop Map/Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner” [3]. It implements the MapReduce model suggested by Google in [1] in free distributions like Apache Hadoop, Yahoo Hadoop and Cloudera.

2.3.1. Hadoop Distributed Filesystem (HDFS).

“A distributed file system is designed to hold a large amount of data and provide access to this data to many clients distributed across a network. There are a number of distributed file systems that solve this problem in different ways” [4] including HDFS for Hadoop’s particular case.
HDFS is designed to store a very large amount of information (terabytes or even petabytes) which requires spreading the data across a large number of machines. This allows support for much larger file sizes than NFS and allows all the processing machines in the cluster to access a complete and consistent copy of the information at hand. However, it requires an extra step to upload the data into the distributed file system where it’s stored in the DataNodes as blocks of data. The location of these data blocks is monitored and managed by a server referred to as the NameNode.

2.3.2. Hadoop Streaming

Hadoop is based on Linux environments and suggests Java for the programming of the mapper and reducer. However, Hadoop provides an API to MapReduce that allows writing the map and reduce functions in languages other than Java: Hadoop Streaming uses Unix standard streams as the interface between Hadoop and an external program, so any language that can read standard input and write to standard output can be used to write the MapReduce program [7]. This can be particularly useful when making use of already existing tools that are to be included in the mapping or reducing stages.
Note that the concept of Hadoop streaming is not in any way related to the concept of video streaming.

3. RELATED WORK

Solutions such as x264farm approached encoding in a distributed environment as early as 2006 [18]. X264farm utilizes x264, a free h264/avc encoder [17], to accomplish encoding into the H.264 format. Not unlike Hadoop, the x264farm system consists of a controller machine that gives commands and data to a set of agent machines that run x264 to perform the encoding.
More in detail, the controller carries out several passes to complete the procedure. The first pass consists of a splitter stage where the unencoded source video is divided into ranges and runs simultaneously with a worker stage. The worker stage picks out ranges from the splitter and streams the raw video data of the range out to the agents. The workers receive the stats file produced by the agents and process it. After the first pass ends, the controller reads the agents’ stat files and breaks it into groups of pictures (GOPs) each starting with an “I” frame. A rate control equation figures out the optimal bitrate for every GOP which is then sent to the agents for encoding. This process is repeated for several more passes. How close the actual bits produced are to the optimal bits calculated ensures a more accurate GOP that requires less re-encoding passes [18].
This is the basis of the system’s controller based encoding approach, where only the controller has access to the input video. However, agent based encoding is also supported where controller’s AVS file is available to all agents and no streaming

over the network is required. This AVS file is used as input to x264, but if for some reason it cannot be found, the controller based approach is preferred [18].

The power of this solution is restricted by its limited format support. This translates into a limited problem area that requires H.264 specifically and encoding only approaches (no transcoding option). In addition, the requirement of raw video data transmission over the network is certainly a drawback that can load the network heavily.

The published implementation of media processing on cloud environments particularly has been focused specially on image processing, not video: Specifically, [11] and [10] discuss an example and an implementation respectively, of image processing tasks on cloud environments.

In [9], this processing is actually achieved through a MapReduce implementation which estimates geographic information (where is the scene?) given a specific image, by leveraging a data set of around 6 million GPS-tagged images and using scene matching to find the most similar image through a reduce-less program.

In the same manner, [12] implements on MapReduce a set of three experiments in image processing, namely a Simple Pixel Detection test, a Blob Detection test and a Sobel Edge Detection test.

One of the major examples of image processing/transformation on cloud environments is the New York Times archive conversion, where the TIFF images of the public domain scanned articles from 1851 to 1922 were converted to PDF format [13]: Around 11 million articles were converted and glued together in under 24 hours processing using around 100 nodes of Amazon Web Services' EC2 instances using the MapReduce model.

As far as video processing goes, fewer examples are available, one of them being HP Labs' VideoToon implementation [14], where a service is implemented to "cartoonize" videos with the help of Hadoop streaming.

Commercially, video transcoding on cloud environments is available through HDCloud.com and Encoding.com, which provide a flexible but proprietary cloud based video transcoding service integrated with Amazon Web Services's EC2, S3, and CloudFront CDN services.

On the other hand, an option for the distribution of video with the help of cloud environments is explored by Fouquet et.al [15]. Fouquet presents a way of making cloud-based infrastructure directly useful for end-users by integrating it into peer-to-peer systems and exemplifies it through an application that makes use of the cloud without requiring a business relationship between the software vendor and the cloud operator. In that application the first peer, the sender, has a live video stream that several clients want to view. The application that distributes the video is an open source P2P streaming system that can work with multiple cloud service providers or on a pure P2P basis. When a user is willing to pay a small amount of money to be able to view a stream and serve as a relay server, improving video quality for him/her as well as other viewers, the application spawns virtual machine images on the cloud on behalf of the user, having him assume the cloud charges that come with it. The peer-to-peer distribution tree is then updated with the new server.

An application as the one described above uses a payment model which is uncommon on today's Internet: Some users pay money and sponsor the service for the others. It is unclear whether its model will be understood and supported by the end-customers although users who sponsor a cloud server could gain certain advantages such as a privileged position in the distribution that guarantees good video quality and higher priority during information package distribution when resources are low.

4. USE CASE

For the purpose of exploring the advantages/disadvantages of implementing cloud computing concepts in the transcoding stage of the video content delivery the process the following use case was explored:

A text file representing a video playlist made from a DVD's content is processed under three different experiments conceived under the light of the cloud storage challenge of node data accessibility:

* Experiment 1 (E1): when the playlist content (the transcoding input) is available preloaded on HDFS and the transcoding output is loaded there as well, generating the necessity of an additional final transfer to the actual web server that is not taken into account in the results (HDFS to HDFS).
* Experiment 2 (E2): When the playlist content is preloaded on HDFS and the transcoding output is stored on an external storage accessible from all cloud processing nodes and potentially the web server (HDFS to Common External or CE).
* Experiment 3 (E3): When the playlist content is available on an external storage accessible from all cloud processing nodes and potentially the web server and the transcoding output is available there as well (CE to CE).

Variations of these experiments are explored as well, where the DVD content is split into chapters (context segmentation), into 10s segments (consistent with HTTP Live streaming recommendation [19]) and a number of pieces equal to the number of processing nodes available (to explore speed-up thanks to distributed processing).

These experiments were run as Hadoop streaming jobs using ffmpeg [20] as a transcoding tool with only a mapping stage. The use of ffmpeg potentially enables the use of all the formats and transcoding options supported by it.

In HTTP Live streaming, the segmented short media files are placed on the web server where the client software requests them in the order specified by a published index and displays them without any pauses or gaps between them [19]. Hence, the implementation of a Hadoop reducing stage was deemed unnecessary as the resulting output information and video segments didn't require to be re-combined in a single output and transcoding could be completely achieved in a single stage.

The hardware on which the tests were run was a 3 node cluster with 2047MiB System memory and two 3.8GHz Intel Xeon processors each; all connected to an external SCSI storage.

This setup should help in the evaluation of a Hadoop-based system that can later be migrated to a low cost unlimited-node public cloud environment such as Amazon. An Amazon based solution could potentially scale-up and support a massive amount of clients with little resource investment.

5. RESULT EVALUATION

The results for each of the three experiments proposed are presented next for analysis.

5.1. Approach comparison

From the experiments, the performance of each of the transcoding source/destination approaches is evaluated.

The total procedure time results (time for the total content to be available including transfer time) can be viewed in Figure 2 for content of a DVD's chapter 20:18 minutes long split in 10 second segments.

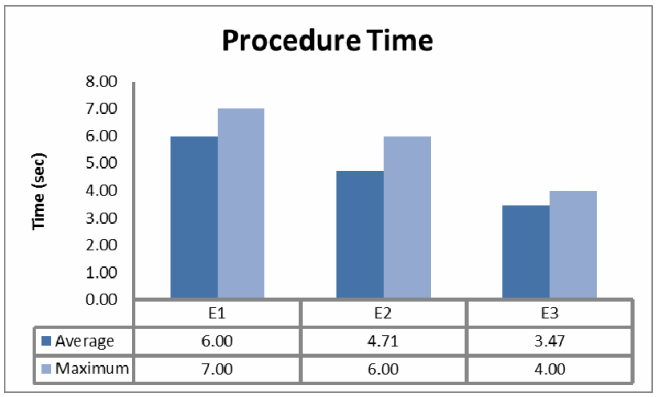


Figure 2. Source/destination combination performance comparison for transcoding

The best results are achieved by experiment number 3, where the total procedure time is minimum. Since the data and encoding paramenters are the same on all experiments, actual processing time is similar in all of them (Figure 3). The main difference lies in the transmission time for HDFS: Note that the experiments are done through the Hadoop Streaming option, hence the transmission to a location where the streaming program can access the data is required.

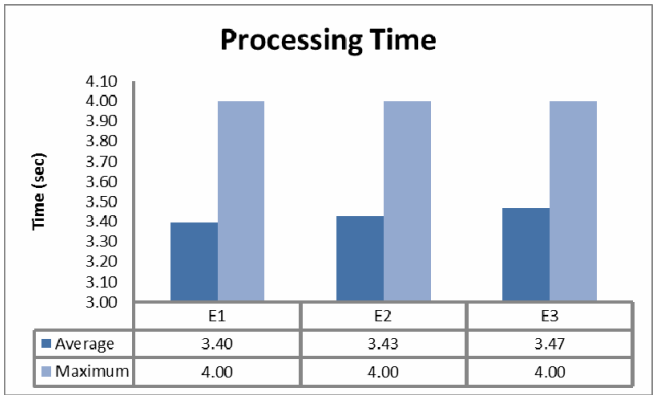


Figure 3. Source/destination combination performance comparison: Processing time

Experiment number 1 requires the transmission of the data twice, once out of HDFS and once transcoded, back into it. Experiment number 2 requires only one transmission, to a location where the file is processed and or/shared with the web server. On Experiment 3 data is only transcoded through the Hadoop job, locations are never changed.

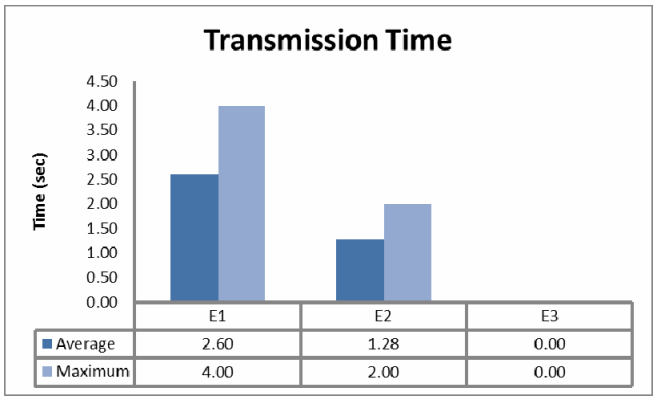


Figure 4. Source/destination combination performance comparison: Transmission time

The time drawback of approach 1 is exacerbated by an additional time required for data movement to a web server location, which is not taken into account in the experiment. Counterbalanced by this, is the less strict machine storage requirements, as the transcoding result storage load is shared among all cluster nodes. This points to an approach that's more appropriate for offline transcoding where no time constraints are present.

5.2. Availability comparison

Next, the availability of the content was studied.

Once the content is requested, a delay between the request and the availability of an output due to the encoding process is expected. Keeping this delay to a minimum is clearly desirable as it impacts the user's experience directly. This transcoding delay was analyzed under 3 different variations for all three experiments:

* Having the content split into the same number of pieces as there are nodes available for processing
* Having the content split by context (in DVD chapters)
* Having the content split into 10s segments

As already established in the previous section, experiment 3 (CE to CE), yields better results, conclusion which is consistent with the outcome for these variations. Hence, only the results for experiment 3 are presented next.

The three figures presented next show the time required in the system for the production of an encoded output (availability), as a function of the start time of the respective content split during normal playback. The minimum requirement is that the content is available at least at playback time in order to enable its display without gaps. This is represented by the Minimum Availability Requirement curve. In an ideal environment, the produced availability curve never crosses the minimum curve. In reality, the best case scenario will minimize this effect, particularly at the beginning of the encoding process where the user is impacted directly by a waiting period.

Trick modes such as rewinding and forwarding in the video are not studied here as solutions such as HTTP Live streaming don't focus on them.

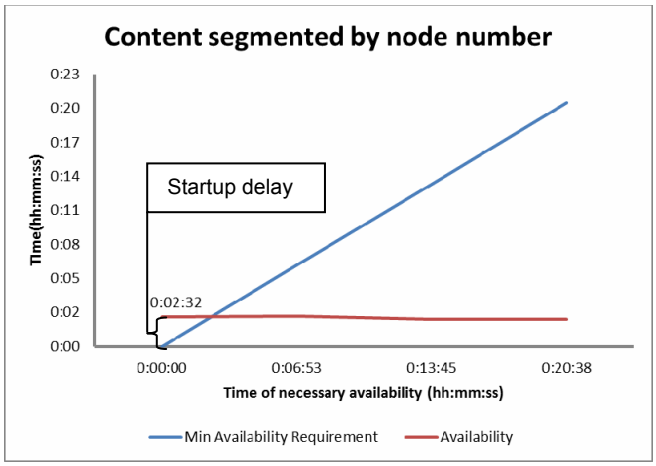


Figure 5. Availability time for a DVD chapter (20m 18s long) split in 3 equal parts. First segment delay: 2m 32s.

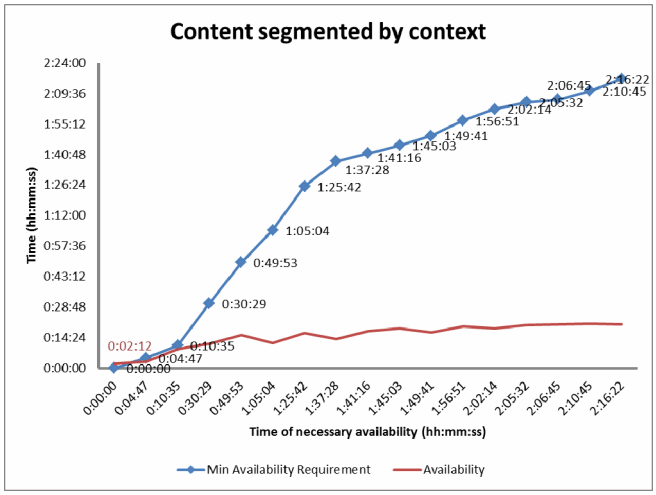


Figure 6. Availability time for all the segments in the DVD (2h 18min long). First segment delay: 2m 12s.
Note: Segments by context have different durations.

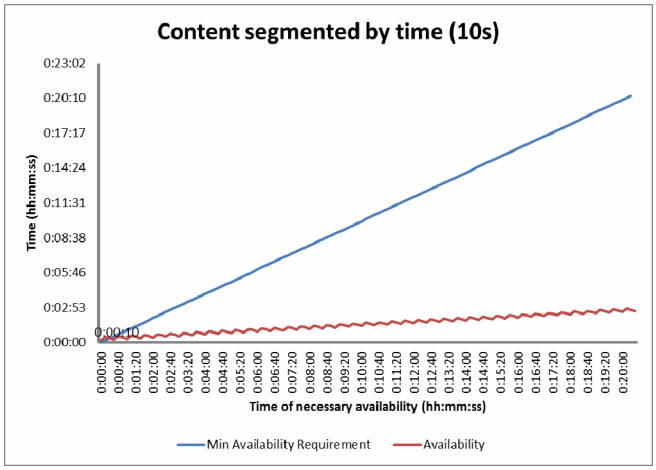
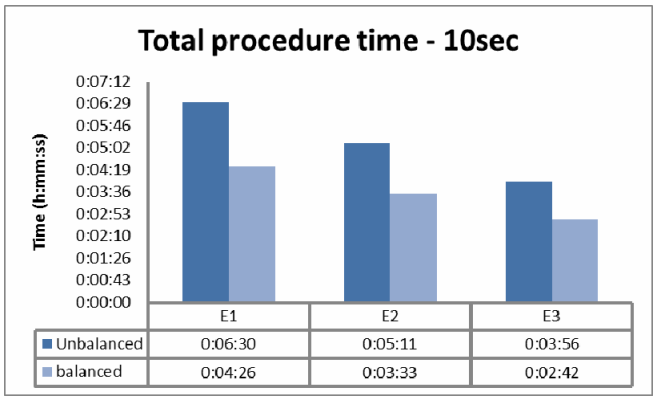


Figure 7. Availability time for a DVD chapter (20m 18s long) split in 124 10s segments. First segment delay: 10s.

From the Figure 5 to Figure 7 it can be observed that the availability time is reduced when the content is split equally among the number of nodes available for processing, but at the expense of a longer starting delay.
Splitting the content by context makes processing more unpredictable and the starting delay time will be directly related to the first split's encoding delay.
The preferred performance is achieved with the 10s segment split: The starting delay is low and the throughput uniform over time. The serrated effect on the availability curve is due to a slightly faster output for splits belonging to a later time frame consequence of the split distribution among the nodes.

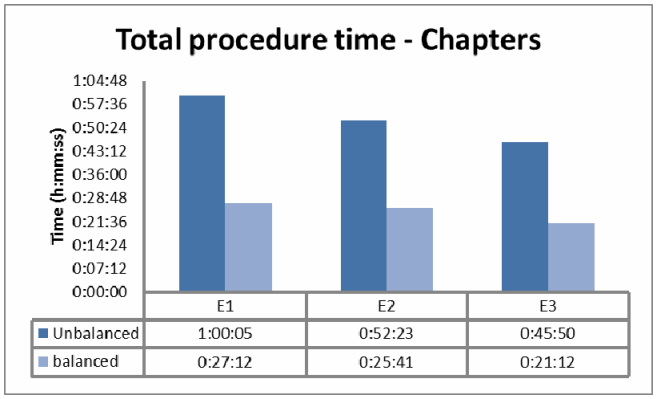
5.3. Additional observations

During experiments it was noticed that a better results can be achieved by direct manipulation of the playlist; the input list can be sorted and split among the nodes according to playback order and file size achieving lower total procedure completion time.
Notice Figure 8 where the 10s segment playlist (the list has similar sized items all over) is divided in several different Hadoop job input files. This split can assure better balancing among the nodes and ensure the participation of them all.



	E1	E2	E3
Unbalanced	0:06:30	0:05:11	0:03:56
balanced	0:04:26	0:03:33	0:02:42

Figure 8. Balanced vs. unbalanced experiments. 10s segments for a 20m 18s video.



	E1	E2	E3
Unbalanced	1:00:05	0:52:23	0:45:50
balanced	0:27:12	0:25:41	0:21:12

Figure 9. Balanced vs. unbalanced experiments for DVD chapters.

The balancing of the splits by context (by sorting, arranging by size and dividing the load equally among the nodes) reduces the total procedure time significantly as shown in Figure 9.

This reduction is particularly significant for this case, where the first chapters of the DVD pose a bigger load than the latter ones and they should be properly divided among the nodes.

6. CONCLUSIONS

MapReduce as a distributed computing technique has gained importance and has become key system in noteworthy commercial endeavors (Google, Yahoo, NetFlix, etc.) As discussed in the related work section the extension of this text-oriented technique to other areas like media processing has already begun. Its contribution in media distribution is only starting but could be significant: The processing power of a Hadoop cluster can greatly improve encoding times and free the web server's resources from unrelated tasks.

As further work is performed, the load in number of HTTP Live streaming users remains to be tested with this particular hardware configuration and approach. In the mean time, the results of this study point to interesting possibilities as far as low cost video and distribution goes which can be particularly benefit mobile device users.

7. REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in Proceedings of OSDI '04: 6th Symposium on Operating System Design and Implemention, San Francisco, CA, Dec. 2004. Online at: http://labs.google.com/papers/mapreduce.html

[2] "Apache Hadoop" Online at: http://hadoop.apache.org/#What+Is+Hadoop%3F

[3] "Map/Reduce Tutorial" Online at: http://hadoop.apache.org/common/docs/current/mapred_tutorial.html

[4] "Hadoop HDFS" Online at: http://hadoop.apache.org/common/docs/current/hdfs_user_guide.html

[5] "Yahoo Hadoop Tutorial" Online at: http://developer.yahoo.com/hadoop/tutorial/module4.html

[6] "Yahoo HDFS Tutorial" Online at: http://developer.yahoo.com/hadoop/tutorial/module2.html

[7] T. White, "Hadoop: The Definitive Guide", O'Reilly, First Edition. June (2009) ISBN: 978-0-596-52197-4

[8] D. Hilley, "Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings", Georgia Institute of Technology, April (2009) Online at: http://www.cercs.gatech.edu/tech-reports/tr2009/git-cercs-09-13.pdf

[9] S. Chen, S. W. Schlosser. Map-Reduce Meets Wider Varieties of Applications, IRP-TR-08-05, Technical Report, Intel Research Pittsburgh, May (2008) Online at: http://www.google.com/url?sa=t&source=web&cd=1&ved=0CBYQFjAA&url=http%3A%2F%2Fwww.pittsburgh.intel-research.net%2F~chensm%2Fpapers%2FIRP-TR-08-05.pdf&ei=YXASTIHvGMKclgfk593DBg&usg=AFQjCNGwpE_z9dc4qJ7Lr1faSni7ylL1ZQ

[10] D. Chantry, "Mapping applications to the Cloud" Microsoft Corporation—Platform Architecture Team, January (2009) Online at: http://msdn.microsoft.com/en-us/library/dd430340.aspx

[11] H. Trease, D. Fraser, R. Farber, S. Elbert, "Using Transaction Based Parallel Computing to Solve Image Processing and Computational Physics Problems" Online at: http://www.cca08.org/papers/Poster31-Harold-Trease.pdf

[12] J. Conner, "Customizing Input File Formats for Image Processing in Hadoop", Arizona State University. July (2009) Online at: http://hpc.asu.edu/node/97

[13] New York Times, "Self-service, Prorated Super Computing Fun!" November 1, (2007) Online at: http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/

[14] HP Labs, "VideoToon" Online at: http://www.hpl.hp.com/open_innovation/cloud_collaboration/cloud_demo_transcript.html

[15] M. Fouquet, H. Niedermayer, G. CarleCloud, "Computing for the Masses" Technische Universität München, Proceedings of the 1st ACM workshop on User-provided networking: challenges and opportunities, Rome, Italy (2009), p.31-36 Online at: http://portal.acm.org/citation.cfm?id=1659029.1659038

[16] K. Asanovic, R. Bodik, B. Catanzaro, et al., "The landscape of parallel computing research: a view from Berkeley", Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December (2006). Online at: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

[17] VideoLan, "x264 - a free h264/avc encoder" Online at: http://www.videolan.org/developers/x264.html

[18] R. Wilson, "x264farm. A distributed video encoder" (2006) Online at: http://omion.dyndns.org/x264farm/x264farm.html

[19] iOS Reference Library, "HTTP Live Streaming-Overview" Online at: http://developer.apple.com/iphone/library/documentation/networkinginternet/conceptual/streamingmediaguide/introduction/introduction.html

[20] ffmpeg, "FFMPEG".Online at: http://www.ffmpeg.org/