Bit Rate Reduction Video Transcoding with Distributed Computing

Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, Johan Lilius
Åbo Akademi University
Department of Information Technologies
Joukahainengatan 3-5, 20520 Turku, Finland
Email: {fjokhio, tdeneke, slafond, jolilius}@abo.fi

***Abstract*—This paper presents an approach to perform bit rate reduction transcoding by video segmentation. The paper shows how a high performance distributed video transcoder can be built using multiple processing units and a Message Passing Interface based parallel programming model. The computation parallelization and data distribution among computing units is discussed. For data distribution coarse grain approach is used in which significant gain in terms of execution speedup is obtained. The segmentation of video stream with (1) equal size having unequal number of intra frames and (2) unequal size having equal number of intra frames is performed to achieve high performance. The results show that the proposed distributed video transcoder provides very short startup times.**

***Keywords*-video Transcoding, Message Passing Interface(MPI), Distributed computing**

I. Introduction

Currently there is a diversity of multimedia applications and there are several video formats available with different characteristics. Users want to play a video in various formats and on different devices. Some users demand for high definition video while others demand for low resolution video. With the passage of time the number of video compression standards is growing therefore it is not practically possible to store a video in all possible formats to fulfill the end user requirements. Also the channels through which a video is distributed to the end user can have different capacities. The compressed video stream needs to be re-encoded to meet the bit rate of the communication channel. Video transcoding is a popular technique to solve these issues [1], [2].

A video transcoder takes a compressed video signal as input and produces another compressed video signal as output. The output video can have different bit rates, different frame rate, different frame size, any combination of these or even an entirely different compression standard.

To get better quality video, the video transcoder should decode the encoded video stream and then re-encode it with new characteristics. Both video decoding and encoding require a number of highly computational tasks hence this seems a time consuming process and waste of computational power. Other smart video transcoding techniques already exist which transcode the video in desired format by just partially decoding and reusing the motion vectors information [3]. Even with existing fast transcoding techniques, the process of transcoding still needs lot of computational power while dealing with high resolution videos such as 4CIF, 16CIF, HD 1080, etc. In order to get better performance, distributed video transcoding can be used to distribute the computational burden among processing units [4].

The main contributions of the paper are:

- the analysis of video segmentation for transcoding in a distributed environment
- the evaluation of the video startup time in such environment

This work also puts special attention towards scalability of the transcoder implementation. This means that we must be able to run the same transcoder in a distributed environment with any number of processing units.

In the next section, we briefly describe the background and related work. Section III gives an overview of the encoding, decoding and introduces the bit rate reduction transcoding. The method for achieving the bit rate reduction and motion vectors refinement is also discussed in this section. In section IV video stream structure is described briefly and then three different ways of video segmentation are discussed. Section V presents the system overview and describes the tasks performed by master and worker machines. In section VI experimental setup is discussed and results are provided in section VII. Finally conclusion and future work is given in section VIII.

II. Background and related work

Video transcoding has been studied and improved in the past two decades. The quality of the transcoded video and the speed of the transcoding process are main issues in video transcoding. Distributed computing is a solution to get more speedup in the transcoding process and keep the same quality of video. However, how to optimally handle multiple video streams, their startup times and how to scale the transcoding architecture is still open research problem.

Jiani Guo [5] proposes a cluster based multimedia web server. The work was related to dynamic generation of video according to the requirements of bandwidth and bit rate for many clients. The partitioning of jobs is done by a media server and the computation is done on several nodes. The proposed system was designed for seven nodes. Sambe [4] worked on distributed transcoding of MPEG-2 to produce

output video with different bit rates. His work was concerned to produce multiple formats and rates and he integrated multiple processors to fully decode and re-encode incoming video. He paid more attention on segment handling while Jiani Guo paid more attention on load balancing of multiple video streams. Neither author considered video startup time.

Among different methods of distributed computing we have chosen to use MPI (message passing interface) because of its maturity, support, open source nature, scalability and ease of use. MPI is a message passing interface for MIMD (multiple instructions multiple data) distributed memory concurrent computers and workstations [6]. In this programming model a set of tasks that use their own local memory during computation can be performed on the same physical machine and /or across an arbitrary number of machines. Tasks exchange data through communication channels by sending and receiving messages (i.e. message passing.). This means that data transfer usually requires cooperative operations to be performed by each process. That is for example, a send operation must have a matching receive operation. From programming perspective, message passing implementations commonly comprise a library of subroutines that are embedded in source code. The programmer is thus responsible and free to express all parallelism involved [6], [7].

III. Bit rate reduction video transcoding

The main goal of bit rate reduction transcoding is to reduce the bit rate while maintaining low complexity and achieving the best quality possible. The bit rate reduction video transcoding has wide range of applications such as television broadcast and streaming of video over the internet. In bit rate reduction video transcoding the compressed video is decoded and then re-encoded with new bit rates.

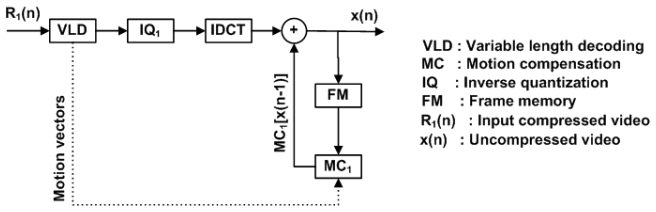

Figure 1. Video decoder

The block diagram of a video decoder is shown in figure Figure 1. Video decoding is a complex operation and it consists of several other operations such as variable length decoding, inverse quantization, inverse discrete cosine transformation, motion compensation. The computation required in decoding operation for low resolution video frames is less as compared with the high resolution video frames. The video decoder has a compressed bit stream as input and produces an uncompressed video as output.

Figure 2 shows the block diagram of a video encoder. As shown in the figure, the video encoding has even more complex operations than video decoding. A video encoder consists of a number of other operations such as discrete cosine transform, quantization, variable length coding, and motion compensation. A video encoder also performs the decoding operation after the quantization operation and then computes the difference between the original video frame and the decoded frame after compression. This difference is termed as a residual frame and is also sent with the compressed bit stream. The information entropy for a residual frame is usually less due to similarities in the nearby video frames, and it requires fewer bits.

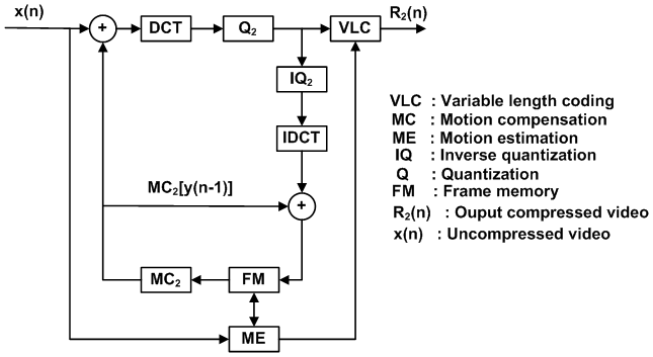

Figure 2. Video encoder

The different operations such as variable length encoding, variable length decoding, quantization, inverse quantization, discrete cosine transform, inverse discrete cosine transform, motion estimation and motion compensation are performed on the block level in a frame. Figure 3 shows the structure of a video frame. The number of blocks in a frame depends on its resolution. If the frame has high resolution, the number of blocks will also be high and more computation will be required during encoding and decoding process.

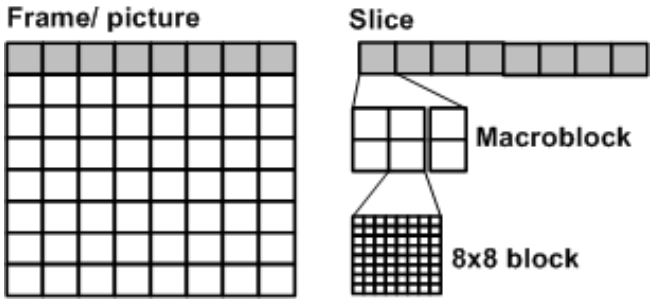

Figure 3. Video frame

In a bit rate reduction transcoder the video resolution and the frames rate are unchanged. The bit rate reduction is possible by compromising on the video quality [8], [9], [10]. It is possible to reduce the bit rate by applying the

inverse quantization and then again applying the quantization with increased quantization step at the encoder side in the transcoder [11], [12], [13]. This operation will increase the zero quantized coefficients and hence fewer bits will be required to encode the data. In order to reduce the complexity in bit rate reduction transcoding the motion vectors computed at the original bit rate are reused in the reduced rate bit stream. Using the same motion vectors will lead to degraded video quality due to the mismatch between prediction and residual components [3]. To overcome this loss of quality motion vector refinement is needed. Video frames contain objects and background. Successive video frames may have similar objects and and these objects can be displaced at another location. Motion estimation is used to examine the movement of objects. In block based motion estimation, the similar blocks are searched in the reference frame. The estimated motion of a block is represented by a motion vector. The motion estimation is performed within a fixed search window and it may have size such as [-2, +2], [-16, +16] or any other suitable size. In order to keep the low complexity, motion vector refinement is performed with small search window[1]. The size of the search window is kept very small to reduce the computational load. Increase in the search window size will give slightly better quality but it will have more computational load. The [-2, +2] search window achieves the majority of gain due to the fact that the majority of macro blocks will have a best match within this range.

IV. Video stream structure

A video stream consists of several independent units called as video sequences where every sequence has its own headers. The video sequence consists of several group of pictures (GOP). The group of pictures consists of frames. There are different types of frames; I (intra) frame, P (Predictive) frame and B (bidirectional) frame. The frame is further divided in slices, each slice consists of macro blocks and every macro block consists of blocks. Figure 4 shows the video stream structure down to the frame level.

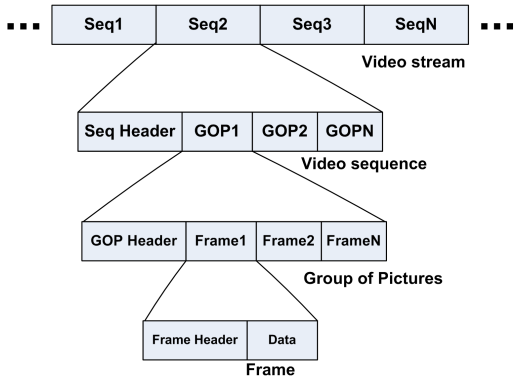

Figure 4. Video stream structure

A. Segmentation of video sequence at Group of Picture

The first issue with distributed transcoding approach is how to perform the segmentation of the source video so that parts of video can be distributed among worker machines to perform the transcoding operations. Compressed video files contain different types of frames (I, P, B) which have different compression rates and inter-dependencies among them. Therefore one cannot split a given video at any particular frame. Among the frame types a frame of type I (intra) is independent and can be decoded without any other reference frame. In a given video sequence a group of frames containing one I frame followed by a number of other B and P frames is called a group of pictures (GOP). Our video sequence partitioning algorithm utilizes this concept to divide a video file in to smaller parts. The master machine divides the incoming video file into parts which contain a number of GOPs and sends these parts to worker machines.

In any video sequence there are two kinds of group of pictures, either the entire video sequence will consist of open-GOP or it will consist of closed-GOP. In the case of closed-GOP it is possible to decode the entire GOP independently. In the case of open-GOP, the last I or P frames of the previous GOP is needed as a reference frame to decode the first B type frame. Segmentation of open-GOP is further discussed in [4]. In the case of open-GOP in every segment there will be one extra I or P frame from the other GOP. This extra frame will be discarded by the master machine while performing the merging. However it will require some extra computation in the transcoding process. In our experiments we used closed-GOPs in the source video.

We segmented the video in three difference ways:

- each segment has equal number of intra frames but the size of segment may be different due to the different sizes of GOPs.
- each segment has equal size but the number of intra frames may be different.
- each segment has unequal size and unequal number of intra frames.

Most video sequences have different number of frames in each GOP. The size of the source video sequence segments for first two cases is shown in table I. The source video used is big buck bunny, further details about the source video are provided in the experimental setup.

The third method of video segmentation is used to get the minimum video startup time. The video startup time is the time at which the end user will be able to view the video.

The MPI based transcoder implementation can handle the transcoding if the number of segments is more than the number of workers.

B. Video segmentation with unequal load

Figure 5 shows the video segmentation of source video with unequal load.

paritions	equal size partitions	Unequal size in Mega Bytes
2	39.95MB each	28, 51
3	26.63MB each	19, 21, 39
4	19.97MB each	14, 15, 19, 33
5	15.98MB each	11, 12, 13, 16, 29
6	13.32MB each	8.6, 9.0, 11, 12, 12, 27
7	11.41MB each	7.5, 7.8, 8.7, 8.8, 8.9, 11, 26

Table I
SIZE OF SEGMENTS FOR TRANSCODING

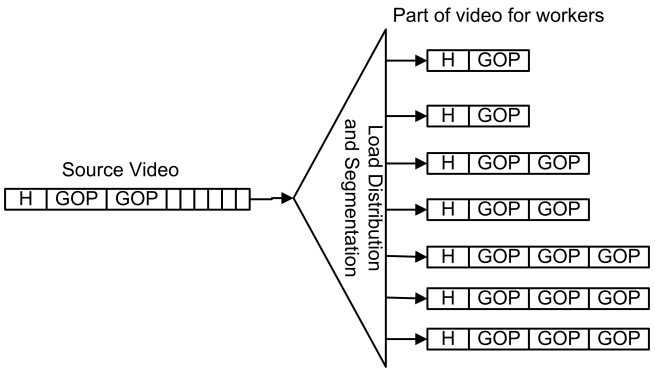

Figure 5. Video segmentation with unequal load

The MPI based transcoder with unequal size partitions is designed in such a way that the master will produce a very small size segment and will send it to the first worker. By keeping the small size of first video segment, it will require very less processing power for transcoding and hence the video start up time will be very less.

The second segment will have slightly more number of I frames and will be bigger in size than the first segment. While producing the second segment the master machine will already have the information about the first segment size and type of frames inside it. The master will make sure that the transcoding time of the second partition is less than the play time of first segment. In the same way it will keep record of the play time of the first n-1 segments while making the nth segment. Since the transcoding operation is performed in parallel the master machine will have a choice to make bigger segments after sending some parts of source video to workers. More care is required when sending segments to the first few workers.

After sending small size segments to few workers, the master will send bigger size segments to other workers. If the segments size will be bigger, there will be better efficiency in overall transcoding and there will be less traffic on the network due to fewer messages between master and workers.

V. SYSTEM OVERVIEW

We selected the ffmpeg open source video transcoder which is designed to work on a single machine as the basis for our experiments. Further details about this transcoder can be found in [14]. We modified this transcoder to execute on multiple processing units in a distributed environment using the MPI. In MPI based implementation we create multiple processes of the transcoder and each process transcode its own part of video stream.

We have two different scenarios for our transcoding system. In the first case each worker will get only one segment to perform the transcoding for one video sequence. Hence we cannot have more partitions than number of available workers. This scenario is used for the first two possible ways of video segmentation i.e. equal size segmentation with unequal number of intra frames and equal number of intra frames with unequal segment size. The one to one mapping of video segments to worker machines is helpful in getting the overall high performance in transcoding.

The second scenario is used to get the shortest possible startup time. In this case the number of video segments is higher than the number of available workers. The video segments are sent to workers in round robin fashion.

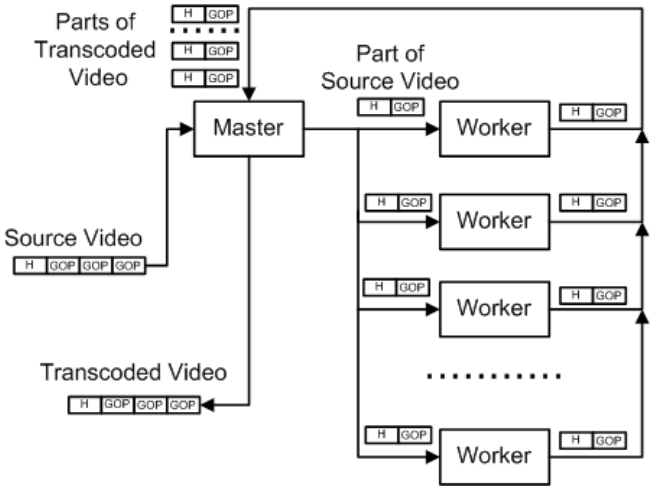

Figure 6. Distributed Video Transcoder with Message Passing Interface

Figure 6 shows the architecture of the distributed video transcoder for a single video sequence. The source video sequence header is attached to every GOP to make it a video sequence so that the transcoder can transcode it according to given parameter values.

In the MPI based implementation the number of total worker machines for transcoding a given video stream is decided by the master; video segmentation and load balancing are performed according to the number of worker machines.

In MPI based systems every machine has its own ID and the work is assigned according to their IDs. The master machine will have ID zero and it will perform the video sequence segmentation task first and then will send the data to worker machines to perform the transcoding operation. The master machine will wait until it gets back the transcoded results from all workers. After receiving the transcoded results it will perform the merging task. It is

also possible to make any other machine as a master with ID other than zero. If the number of video streams to be transcoded is high then multiple masters can also be created.

All worker machines perform the same kind of transcoding operation. The instructions for transcoding the video sequence are the same for all worker machines but they all get different parts of the video stream. The worker machines have to receive the part of video stream from the master machine then perform transcoding operation and send back the results to master machine.

VI. Experimental Setup

The experimental system consists of AMD Opteron(tm) Processor workstation and the configuration of the system is shown in table II. Each core of the Dual-Core processor is used as a processing element and takes part in the transcoding operation. The same implementation of the MPI based video transcoder can be mapped on a multi-core system.

model name	Dual-Core AMD Opteron(tm) Processor 2214 HE
cpu MHz	2194.498
cache size	1024 KB
address sizes	40 bits physical, 48 bits virtual

Table II
Configuration of the machines in cluster

The big buck bunny video sequence [15] was used as source video to perform transcoding operations. The source video has H263 4CIF (704x576) format with 24 fps and 1125 kbps. The size of the source video is 79.9 MB and its play time is 09 minutes and 56 seconds. The total number of frames in this video sequence is 14314.

VII. Results

To test our approach we transcode the video down to lower bit rates. The original size of the video sequence was 79.9 Mega Bytes. We started transcoding video sequence at 982kbps and went down to 349kbps. With lower bit rates the quality of video was degraded. Table III shows the file size after transcoding with different bit rates.

Bit rate	File size	Bit rate	File size
982 kbps	70Mb	518 kbps	44Mb
883 kbps	63Mb	428 kbps	38Mb
789 kbps	57Mb	359 kbps	31Mb
699 kbps	50Mb	349 kbps	25Mb
608 kbps	44Mb		

Table III
File size after transcoding for different bit rates

A. Transcoding time

The bit rate reduction transcoding requires the same number of operations for transcoding the video sequence at different bit rates hence the transcoding time is the same for different bit rates. Figure 7 shows the transcoding time for both equal size partitions having unequal number of intra(I) frames and unequal size partitions having equal number of intra (I) frames. The graph shows that there is gain in terms of speed up when using more workers. With equal number of intra (I) frame partitions the overall transcoding time is less and the performance is better for a single video stream.

The quantization process requires more computation in bit rate reduction transcoding; this process is performed only on intra macro blocks. The intra frames have only intra macro blocks and require more computational power as compared with P and B frames having inter macro blocks. The P and B frames may also have intra macroblocks but the number of those intra macro blocks is very less as compared to the macro blocks of intra frames.

The equal size segmentation with unequal number of intra frames partitioning also provides speed up as the number of workers is increased but is less efficient than the equal number of intra frames and unequal size segmentation.

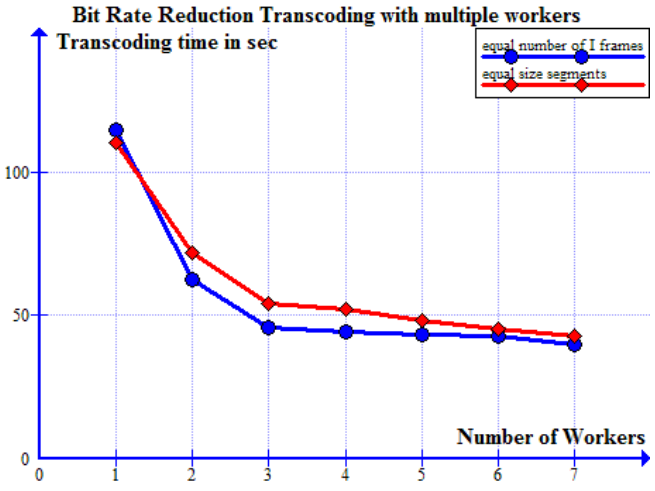

Figure 7. Bit rate reduction Transcoding Time

B. Startup time

The figure 8 shows the start up time for different sizes of video segments. With unequal size segmentation having small number of Intra frames, the startup time can be very less. The results show that for 2 megabytes video segment the transcoding time is less than 2 seconds. The number of frames in 2 mega bytes video is more than 360. With 24 frames per second the play time for this video segment will be 15 seconds. Hence the second worker will be able to transcode a bigger size video segment and it has to send back the transcoded result before the 15 seconds deadline so

that the user may able to see uninterrupted video. The third worker will be able to transcode on even bigger size video segment. The video segmentation time is small as compared with transcoding time. It takes less than 1 second to perform the video segmentation operation and send segments to workers for transcoding.

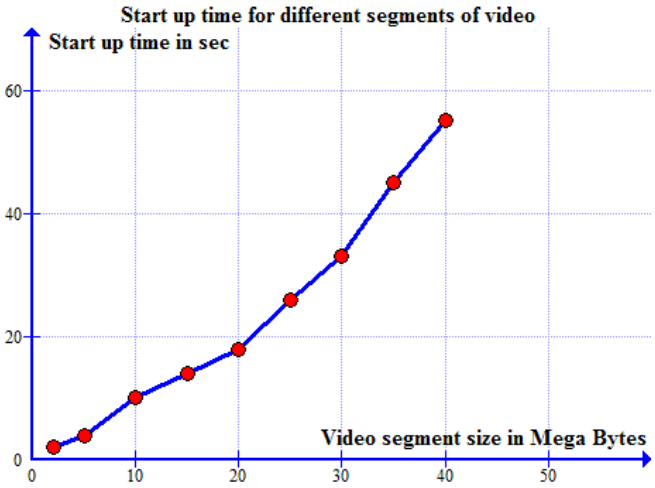

Figure 8. Start up time for different sizes of video segements

C. Analysis

The Peak Signal to Noise Ratio (PSNR) is used to measure the quality of compressed images. The Average Peak Signal to Noise Ratio is used to measure the quality of video. Here PSNR is calculated for all frames of video and then finally Average of PSNR is calculated to get APSNR.

$$PSNR = 10 \times log_{10}(\frac{MaxErr^2 \times W \times H}{\sum_{i=1,j=1}^{W,H}(x_{ij} - y_{ij})^2})$$

The x_{ij} and y_{ij} shows the pixel values of source image and compressed image. The W and H indicate the width and height of the image.

We performed the transcoding operation for different bit rates. The Peak Signal to Noise Ratio at 349kbps is shown in figure 9.

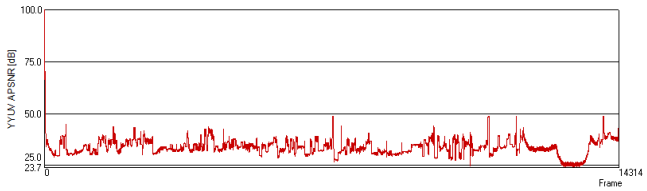

Figure 9. Peak Signal to Noise Ratio at 349kbps

Figure 10 shows the APSNR for different bit rates starting from 349kbps to 982kbps. If the value of APSNR is above 30 it means the video quality is acceptable and if the value of PSNR is higher it means the quality of the compressed image is better. Maximum value of PSNR can be 100 and in that case two images will be exactly identical.

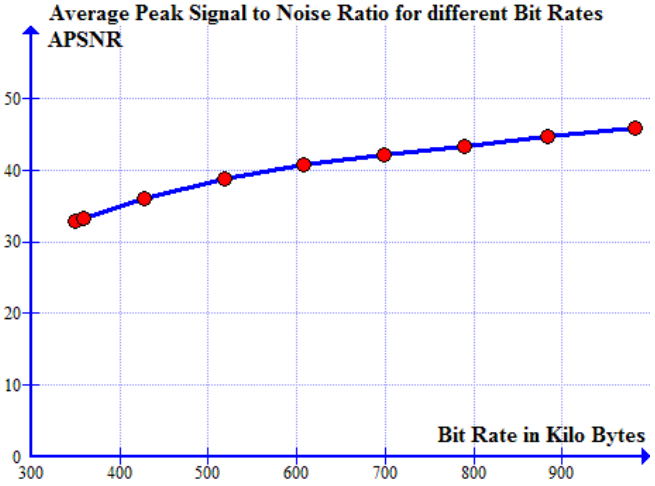

Figure 10. Average PSNR for various bit rates

The transcoding experiment was performed several number of times starting with one master and one worker to one master and seven workers. The output video quality was the same with different number of workers hence the MPI based transcoder did not degraded the video quality.

VIII. Conclusion and Future Work

In this paper, we have proposed a scalable distributed MPI based transcoder implementation. In this implementation a master node (workstation) partitions a given input video file into a number of parts and distributes among worker nodes. The actual transcoding is performed by worker machines in parallel to get more speedup of overall transcoding process. The worker machines send back the transcoded video to master for merging. We were able to see a considerable performance gain with MPI based transcoder as number of worker machines increases. It was observed that the segmentation with equal number of intra frames is more efficient than equal size segmentation. The unequal size segmentation is better for having short startup time. The video start up time can be as low as 2 seconds and then uninterrupted service is possible for the end user. In addition the MPI based transcoder needs no change in its design as the number of processing units grows up. The MPI based transcoder can handle any other type of video format but the headers information needs to be handled while performing segmentation. The MPI based transcoder can also be used for other types of transcoding just like spatial and temporal transcoding and further experiments can be performed on both these types of transcoding.

The MapReduce can also be used to perform video transcoding in a cloud computing environment. In future we

intend to perform distributed transcoding using the cloud computing with both MPI and MapReduce and then see which one of them provides better results.

REFERENCES

[1] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *Signal Processing Magazine, IEEE*, vol. 20, no. 2, pp. 18 – 29, mar 2003.

[2] S. F. Chang and A. Vetro, "Video adaptation: Concepts, technologies, and open issues," *Proceedings of IEEE*, vol. 93, no. 1, pp. 148–158, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1109/JPROC.2004.839600

[3] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," *Consumer Electronics, IEEE Transactions on*, vol. 44, no. 1, pp. 88 –98, feb 1998.

[4] Y. Sambe, S. Watanabe, D. Yu, T. Nakamura, and N. Wakamiya, "High-speed distributed video transcoding for multiple rates and formats," *IEICE Transactions*, vol. 88-D, no. 8, pp. 1923–1931, 2005. [Online]. Available: http://dx.doi.org/10.1093/ietisy/e88-d.8.1923

[5] J. Guo, F. Chen, L. Bhuyan, and R. Kumar, "A cluster-based active router architecture supporting video/audio stream transcoding service," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, april 2003, p. 8 pp.

[6] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*. Knoxville, TN: University of Tennessee, Jun. 1995.

[7] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI, Portable Parallel Programming with the Message Passing Interface*. MIT Press.

[8] P. Assuncao and M. Ghanbari, "Transcoding of single-layer mpeg video into lower rates," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 144, no. 6, pp. 377 – 383, dec 1997.

[9] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *Multimedia, IEEE Transactions on*, vol. 2, no. 2, pp. 101 –110, jun 2000.

[10] H. Sun, W. Kwok, and J. Zdepski, "Architectures for mpeg compressed bitstream scaling," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 2, pp. 191 – 199, apr 1996.

[11] Y. Nakajima, H. Hori, and T. Kanoh, "Rate conversion of mpeg coded video by re-quantization process," in *Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3 - Volume 3*, ser. ICIP '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3408–. [Online]. Available: http://portal.acm.org/citation.cfm?id=839284.841401

[12] M.-T. Sun, T.-D. Wu, and J.-N. Hwang, "Dynamic bit allocation in video combining for multipoint conferencing," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 45, no. 5, pp. 644 –648, may 1998.

[13] O. Werner, "Requantization for transcoding of mpeg-2 intraframes," *Image Processing, IEEE Transactions on*, vol. 8, no. 2, pp. 179 –191, feb 1999.

[14] "Ffmpeg project." [Online]. Available: http://www.ffmpeg.org/

[15] "Big buck bunny video sequence." [Online]. Available: http://www.bigbuckbunny.org/index.php/download/