PAPER

High-speed Distributed Video Transcoding for Multiple Rates and Formats

Yasuo SAMBE[†a)], ***Member***, **Shintaro WATANABE**[†], **Dong YU**[†], ***Nonmembers***, **Taichi NAKAMURA**[††], ***and*** **Naoki WAKAMIYA**[†††], ***Members***

SUMMARY This paper describes a distributed video transcoding system that can simultaneously transcode an MPEG-2 video file into various video coding formats with different rates. The transcoder divides the MPEG-2 file into small segments along the time axis and transcodes them in parallel. Efficient video segment handling methods are proposed that minimize the inter-processor communication overhead and eliminate temporal discontinuities from the re-encoded video. We investigate how segment transcoding should be distributed to obtain the shortest total transcoding time. Experimental results show that implementing distributed transcoding on 10 PCs can decrease the total transcoding time by a factor of about 7 for single transcoding and by a factor of 9.5 for simultaneous three kinds of transcoding rates.

key words: *MPEG, video-transcoding, distributed computing*

1. Introduction

With the launch of digital broadcasting in many countries and the proliferation of digital video disks (DVD), both of which use the MPEG-2 video coding standard [1], it is expected that MPEG-2 will become the de facto video compression format in video archives. On the contrary, as the number of different video compression algorithms in use increases in the networked video application over the Internet, there is a growing demand to convert a pre-encoded MPEG-2 digital video in archives to other compressed formats such as MPEG-1, MPEG-4, H.263 and so on.

Besides converting formats, video content will be altered in terms of bit-rate and resolution to meet the network bandwidth and terminal capability. For instance, the bandwidth of end user access networks can vary from several tens of kilo-bits per second to 20 to 30 megabits per second. Moreover, modern terminals use displays with various sizes and resolutions. Therefore, it is often necessary for service providers delivering video over the Internet to transcode the same content to yield different video formats, spatial resolution, and bit-rates simultaneously.

Several video transcoding techniques have been proposed [2]–[7], and most attempt to decrease the computational complexity by using information like DCT coefficients and the motion vectors extracted from the original coded data. None of them, however, was designed to produce multiple formats and rates. The aim of our work is to provide a video transcoding system that can convert MPEG-2 video files into other kinds of formats and bit-rates at high-speed. To realize multiple transcoding and speed up the transcoding process, we integrate multiple processors to fully decode and re-encode incoming video

Our transcoding system divides the MPEG-2 file into small segments along the time axis and transcodes them in parallel [8]. Parallel transcoding along time axis usually suffers from quality discontinuity and degradation around the segmented cut points in the re-encoded video, because of a lack of information such as the coding complexity of the previous video segment. To get the information of the previous segment from another processor requires inter-processor communication, which leads to additional overhead and a significant performance degradation.

To achieve high performance without significant quality degradation due to parallel transcoding, we propose the segment handling method; it divides in-coming MPEG-2 data with minimum duplication and the data are used to determine re-encoding parameters. We also investigate scheduling algorithms and the segment length of the distributed transcode that minimizes overall transcoding time.

The organization of this paper is as follows. Section 2 introduces the distributed video transcoder architecture. Section 3 proposes a video segment handling techniques for distributed transcoding along with experimental results. In section 4, we investigate the performance model of the transcoder and the optimum segment length. Section 5 uses experimental results to assess the performance model. Our conclusion is presented in Section 6.

2. System Overview

Our transcoding system consists of a source PC, several transcoding PCs, and a merging PC. These PC are connected by Giga-bit Ethernet LAN, as shown in Fig.1.

[†]The authors are with the R&D Headquarters, NTT DATA Corporation, Tokyo, Japan

[††]The author is with the School of Computer Science, Tokyo University of Technology, Tokyo, Japan

[†††]The author is with the Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Osaka, Japan

a) E-mail: sanbey@nttdata.co.jp

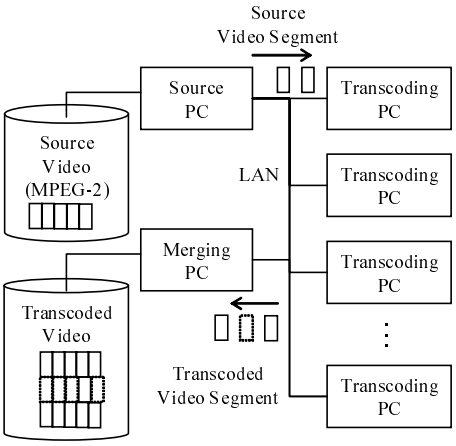


Fig. 1 Distributed video transcoder

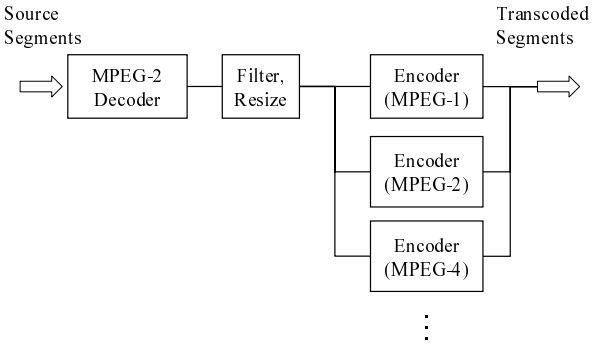


Fig. 2 Block diagram of a transcoding PC

The source PC has a source MPEG-2 Program Stream (PS) input file in which is multiplexed audio and video data. Upon user or operator request, the source PC demultiplexes the audio and video data, divides the MPEG-2 video file into video segments of appropriate length, and transmits these segments to the transcoding PCs. Segment length is determined so that the total transcoding time will be minimized with additional overlapped data, as explained in Section 3 and 4. When the source PC transmits segments to the transcoding PC, it also sends the transcoding parameters that specify the operation of the filter and the encoder of transcoding PC. These parameters include filter function specifications, spatial resolution, temporal resolution, and re-encoding formats desired. The demultiplexing, dividing and transmitting processes are implemented by multi-threaded programming so that they can be performed in parallel. A video segment consists of one or more consecutive Groups of Pictures (GOP). Audio transcoding is done on a single transcoding PC, because it is not computationally expensive to transcode audio. Audio transcoding is ignored hereafter.

Each transcoding PC decodes and re-encodes the video segments into the different video formats specified. Decoded frames are filtered, resized, and passed to the encoder frame by frame. Figure.2 shows a block diagram of the transcoding PC. In the segment transcoding process, a frame is decoded only once and the encoding modules specified by transcoding parameters re-encode the frame. This frame by frame based transcoding architecture gives greater flexibility in transcoding. For example, new encoding modules or new filter operations like digital water marking can be easily added. All of the modules shown in the figure, including transmission modules, are implemented by multi-thread programming. Therefore, both transmitting and transcoding process also can run in parallel.

Transcoded segments are sent to the merging PC and concatenated to form the desired video format files at the merging PC. After the second segment for each desired file is received, the concatenation process begins and runs in parallel to the following segment transcoding. This concatenation process includes modification of time-code. In this paper, video buffer verifier (VBV) requirements are not taken into account, because the probability that a bitstream concatenated with segments does not meet the VBV requirements can be decreased by overlapped segment transcoding. In order to strictly guarantee the requirements, efficient methods for compressed domain video editing as proposed in [12] should be applied .

3. video segment handling

If the video segments are to be transcoded in parallel with minimal communication overhead, we need efficient video segment handling techniques with regard to Open-GOP as well as rate control for re-encoding the segments.

3.1 Segmentation at Open-GOP

There are two kinds of GOP in MPEG-2: Closed-GOP and Open-GOP. In the case of Closed-GOP, all frames of the GOP can be independently decoded and no problem occurs in the decoding process. On the contrary, if the first GOP of the segment is an Open-GOP, the last reference frame of the previous segment is needed to decode the first bidirectional coded frame. To better understand this, Fig.3 shows a typical GOP structure. In this figure, GOP $(k-1)$ is a closed-GOP and GOP (k) is an Open-GOP. Decoding bidirectionally coded frames B14 and B15 requires both P13 of GOP $(k-1)$ and I16. If GOP$(k-1)$ is located on another transcoding PC, the transcoding PC processing GOP(k) should get the decoded P13 frame via inter-processor communication. Generally, Open-GOP is more efficient in coding than Closed-GOP, because the former reduces the temporal redundancy between consecutive frames around GOP boundaries. Therefore most GOPs in MPEG-2 encoded files are Open-GOPs and most segments would

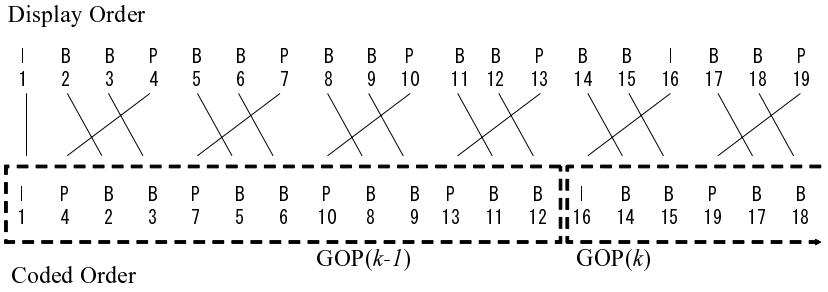


Fig. 3 Segmentation at Open-GOP

begin with Open-GOP.

This leads to the following approaches: (i) transmitting decoded reference frames among transcoding PCs, (ii) duplicating the coded frames data in segmentation so that each segment can be decoded independently. The second approach is more suitable, because the size of coded data is much smaller than decoded data and it takes smaller time to transmit. In this distributed transcoding system, source PC makes video segments with duplicating one GOP after segmentation point and transmits it to a transcoding PC. The first bi-directionally coded frames of the GOP are transcoded by the PC. The next segment except the first bi-directionally coded frames is transcoded by another PC. By this method, all frames of the Open-GOP can be transcoded and the transcoded frames has the same structure of source coded video.

3.2 Re-Encoding Rate Control around Cut Points

If each video segment is encoded independently, segment encoding quality may differ which leads to discontinuity around the segmentation points and irregular video quality. This is because re-encoding parameters of each segment are determined without regard to the coding complexity of the previous segment. The more complexity the video frame has, the more bits must be allocated to make coded video quality constant.

In the widely used MPEG-2 Test Model 5 (TM5) rate control [10], the target size of the frame is made proportional to complexity. In TM5, the frame complexity X_n is defined as the product of the coded frame size S_n (in bits) and the average quantization scale Q_n of the frame, where n denotes the coding picture type (I, P, B). The complexity of the frame to be coded is estimated as the same as that of the same type of previous coded frame. For example in Fig.3, in encoding B14, the complexity of the frame is estimated as the same as that of B12. By doing this, TM5 assures that the video quality keeps consistent. TM5 calculate the target frame size using the complexity. After deciding the target frame size, quantization scale Q of each macro-block is determined so that actual coded size will be equal to the target size using virtual buffer memory d_n. The virtual buffer d_n is the accumulation of difference between the actual size and the target size of coded frames. The quantization scale Q is calculated as $Q = rd_n/31$, where r is reaction parameter which is defined as $r = 2bitrate/framerate$.

Therefore, if the X_n and d_n of the first frames in a segment can be estimated properly, the re-encoder can calculate the target bit budget and the initial quantization scale so that re-encoded video quality is made to be constant around segment cut point. In our system, each transcoding PC determines the complexities of the first frames of the video segment from the source MPEG-2 coded data. In fact, the complexities of the transcoded frames would be different from the input video due to the frame resolution and bit-rate change. However, since it has been shown that there are strong correlations between the input and output video [6], we employ the complexities of the input coded data, multiplying each of them by the ratio of output resolution to that of input. The virtual buffer memory d_n can't be estimated without actual frame size of all re-encoded frames including those of frames on other transcoding PCs. However, d_n is expected to be stabilized within a few GOP transcoding. To shorten the stabilization period, d_n of the first frames is calculated from source MPEG-2 data as $d_n = 31Q_n/r$. The re-encoder of transcoding PC begins to transcode one GOP before segmentation point with the initial complexities and virtual buffer memory.

To verify the above proposed segment handling, we conducted segment transcoding experiments as follows:

Table 1 Experimental transcoding conditions

Source Video Format	MPEG-2 MP@ML
Source Video	*football*, *flower garden* *mobile & calendar*, *sailboat*
Source Video Rate	8 Mbit/s
Source Frame Size	Horizontal 720 pixels, Vertical 480 lines
Output Video Coding	MPEG-2, MPEG-4 ASP
Output Video Rate	MPEG-2: 2 Mbps, 1 Mbps MPEG-4: 1 Mbps, 750 Kbps
Output Frame Size	Horizontal 360 pixels, Vertical 240 lines
GOP/GOV Structure	M=3, N=15

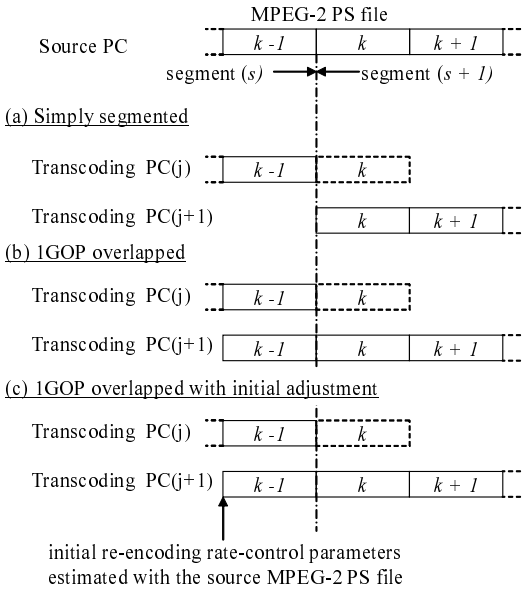


Fig. 4 Video segment handling methods

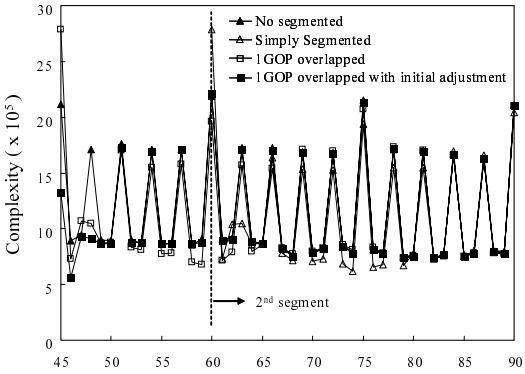


Fig. 5 Complexity around cut-point: MPEG-2 of 2Mbit/s, *mobile* & *calendar*

(a) simply segmented and transcoded independently, (b) 1GOP overlapped transcoding without initial parameter adjustment, (c) 1GOP overlapped transcoding with initial parameters, as the above proposed method. These methods are illustrated in Fig.4 and the transcoding parameters are listed in Table.1. The first segment has the first 60 frames and the second segment has the next 60 frames. Figure 5 and 6 show the the complexity X_n and virtual buffer memory d_n using the standard video *mobile* & *calendar* for MPEG-2 transcoded at 2 Mbit/sec and Fig.7- 8 show them using *sailboat* for MPEG-4 transcoded at 750 Kbit/sec. We can see that the X_n and d_n of the proposed method (c) in the second segment are the most approximate to those of transcoding without segmentation. The proposed initial adjustment shorten the period in which the X_n and d_n become close to those of non segmented transcoding, compared with the one GOP overlapped transcoding without initial adjustment. Figure

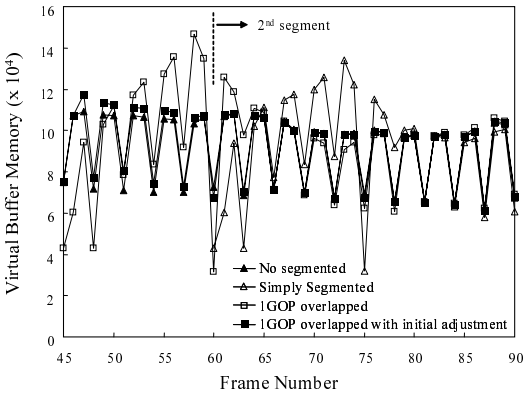


Fig. 6 Virtual buffer memory around cut-point: MPEG-2 of 2Mbit/s, *mobile* & *calendar*

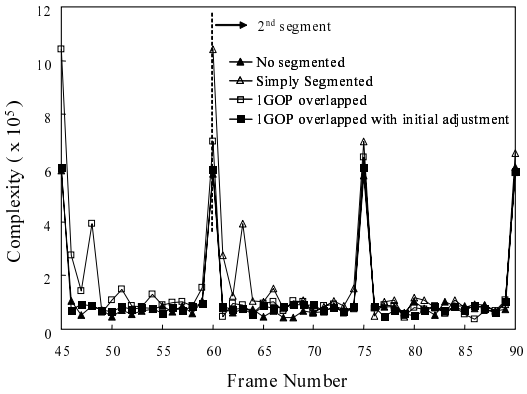


Fig. 7 Complexity around cut-point: MPEG-4 of 750Kbit/s, *sailboat*

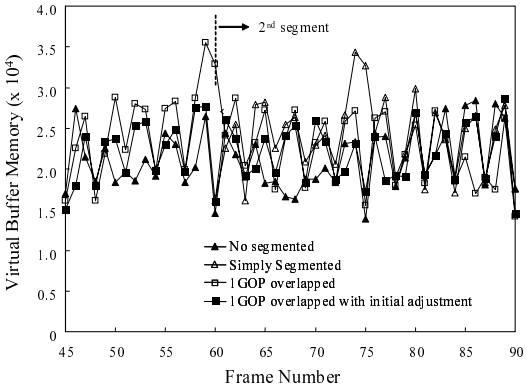


Fig. 8 Virtual buffer memory around cut-point: MPEG-4 of 750Kbit/s, *sailboat*

9 shows the Peak Signal-to-Noise Ratio (PSNR) of these methods using the standard video *mobile&garden* for MPEG-2 transcoded at 2 Mbit/sec and Fig.10 shows them using *sailboat* for MPEG-4 transcoded at 750 Kbit/sec. These results show that the proposed method (c) achieves the same quality and continuity as the non-segmented video sequence. Example transcoded pic-

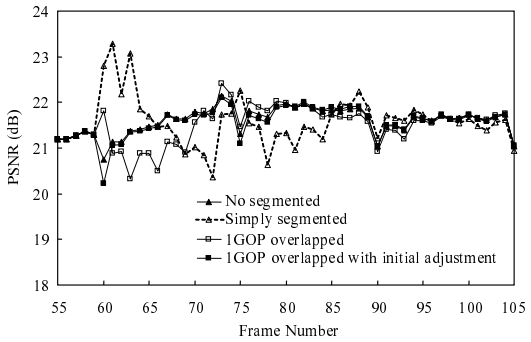


Fig. 9 Transcoded video quality around cut-point: MPEG-2 of 2Mbit/s, *mobile & calendar*

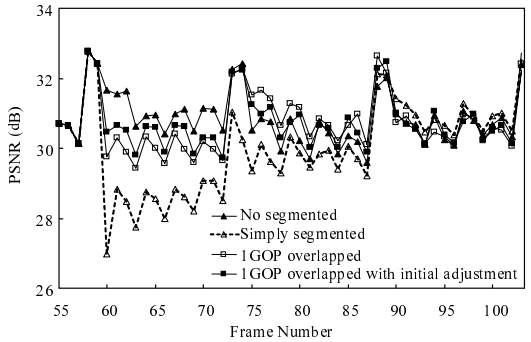


Fig. 10 Transcoded video quality around cut-point: MPEG-4 of 750Kbit/s, *sailboat*

Fig. 11 An example frame of simply segmented method (a) at cut-point

Fig. 12 An example frame of the proposed transcoding method (c) at cut-point

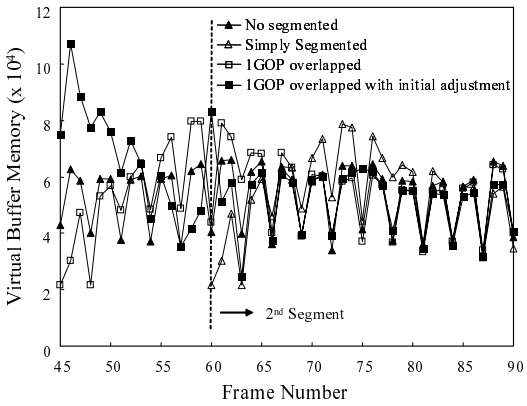


Fig. 13 Virtual buffer memory around cut-point: MPEG-4 of 1Mbit/s, *sailboat*

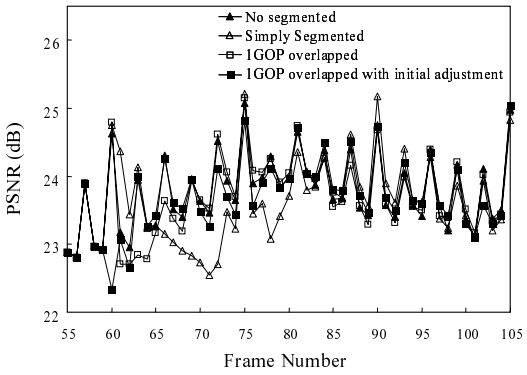


Fig. 14 Transcoded video quality around cut-point: MPEG-4 of 1Mbit/s, *sailboat*

tures are shown in Fig.11 and Fig.12. These pictures are the frame 60 of simply segmented method and that of the proposed method in Fig.10, respectively. The proposed method is especially efficient for video having little movement like this *sailboat* video, because the even short time quality degradation as Fig.11 for this kind of video is very noticeable.

Table 2 shows the comparison of quality degradation during 30 frames after cut-point (the frame 60), compared with transcoded video without segmentation.

Although the results show the proposed method achieve the least quality degradation for many transcoding conditions, there are some cases in which method (b) 's degradation are smaller than those of the proposed method. This is because the initial estimation of virtual buffer memory d_n doesn't work well due to saturation of quantization scale Q estimated for that of output video segment. When this case occurs, the estimated size of previous coded frames are smaller than that of actual size, and then the d_n is estimated much higher than that of video without segmentation.

Table 2 Comparison of image quality degradation for segment transcoding methods

format	rate (bit/s)	Simply segmented (dB)	1GOP Overlapped (dB)	1GOP Overlapped with initial adjustment (dB)
MPEG-4				
football	750K	-24.6	-16.7	-6.1
flower	750K	-13.8	-2.3	-5.3
mobile&calendar	750K	-15.0	-8.9	-6.2
sailboat	750K	-44.8	-14.4	-9.7
football	1M	-17.3	-6.4	-3.1
flower	1M	-11.9	-2.2	-5.0
mobile&calendar	1M	-11.3	-4.2	-6.5
sailboat	1M	-46.2	-12.8	-3.3
MPEG-2				
football	1M	-33.9	-13.2	-2.0
flower	1M	-15.0	-5.2	-2.0
mobile&calendar	1M	-10.8	-6.9	-1.8
sailboat	1M	-32.5	-5.6	-5.8
football	2M	-12.9	-1.4	-1.3
flower	2M	-11.7	-1.0	-0.4
mobile&calendar	2M	-9.7	-2.1	-0.2
sailboat	2M	-28.9	-0.2	-0.2

As an example of this case, d_n using *mobile & calendar* for MPEG-4 transcoded at 1 Mbit/sec is shown in Fig. 13. There are large discrepancies at the initial frame (frame 45) between the d_n of the proposed method and that of the non segmented method and some differences remains at the second segment. However, as shown in Fig 14, the period in which the proposed method has the most quality degradation is limited within a few frames.

4. Segment Allocation of Distributed Transcoding

This section describes how the video segments should be allocated to transcoding PCs in order to minimize the overall transcoding time.

4.1 Segment Transcoding Time

In a transcoding PC, since the MPEG-2 decoder, filter, and encoders are implemented as threads operating in parallel and the encoding process is the most time consuming process, the segment transcoding time primarily depends on the encoder.

Most encoding algorithms include discrete cosine transform, motion compensation (MC). In this paper, the encoders we use employ simple block matching motion estimation with fixed search range in the MC. Therefore, The time taken to transcode segments are expected to be constant irrespective of the degree of movement and the texture.

In order to ensure the assumption that segment transcoding time can be treated as constant and to investigate how the video segment transcoding can be determined,

we measured the MPEG-4 transcoding time of 40 second video segments created from a one hour football video. The video sequence included a wide variety of frames in terms of the degree of movement and texture. The other transcoding parameters were the same as those in Table 1. The transcoding PC had two 1.26 GHz Pentium processors and 2GB of memory. Fig.15 and Fig.16 show the transcoding time and the time histogram respectively. These results show that the fluctuation in segment transcoding time is about only 5% or so and the transcoding speed can be estimated as constant within the video sequence. Therefore, for a transcoding job, the video segment transcoding time can be taken as the product of c and d, where c is transcoding performance and d is segment length in terms of display time.

This c is time taken to transcode unit length of source video and is the sum of decoding time c_{dec} and re-encoding time c_{enc}. For multiple transcoding into p kinds of rates or formats, the time taken to encode only increases: $c = c_{dec} + pc_{enc}$. However, in case that transcoding PC has multiple CPUs like our experimental system, the part of encoding might be done in parallel and the re-encoding time is less than pc_{enc}. This parallelization effect depends on the implementation of encoder. In our system having dual CPU, when two encoders run, about 60% of encoding process is overlapped and c can be estimated as

$$\begin{aligned} c &= c_{dec} + c_{enc} && (p = 1) \\ &= c_{dec} + 1.4\lceil \frac{p}{2} \rceil c_{enc} && (p \geq 2) \end{aligned} \quad (1)$$

where $c_{dec} \approx 0.5, c_{enc} \approx 1.5$ for the transcoding conditions listed as Table.1. The estimation of c_{dec} and c_{enc} using other transcoding conditions remains further study.

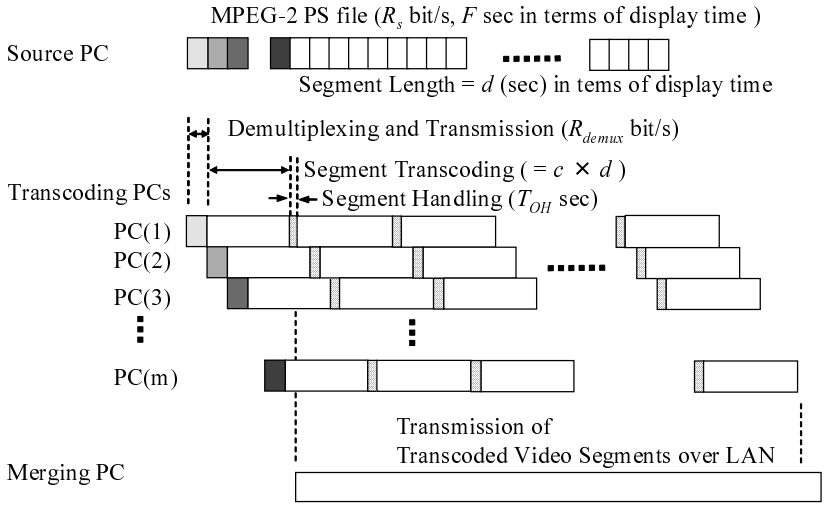


Fig. 17 Process flow of the distributed transcoding

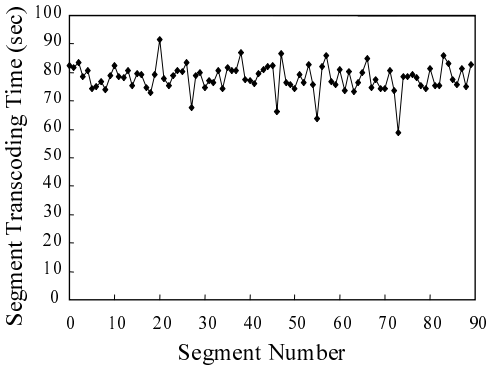


Fig. 15 Transcoding time of each 40 second video segment

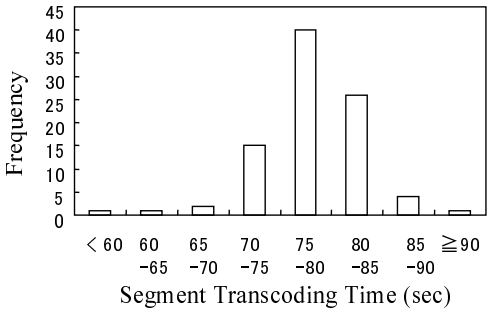


Fig. 16 Histogram of the segment transcoding time

4.2 Performance model and the optimum segment length

Since the segment transcoding time can be considered as constant, the proposed system implements a simple round-robin scheduling method (Fig.17). Segment lengths are equal, as is transcoding PC performance. For this allocation, shortening the length of the video segment decreases the waiting time of each transcoding PC, because the delay until the first video segment is transmitted to the transcoding PC becomes shorter. However, this increases overhead costs including the segment handling process described in the previous section and transmission overhead such as connection setting, making the total transcoding time longer.

If we assume that segment transmission time is relatively small and can be neglected,the total transcoding time (T_{total}) can be estimated as the sum of the time taken to transcode a source file (length F in display time) using m PCs in parallel and the time (T_c) which is not able to do in parallel. The former is cF/m. According to Amdahl's law [9], the total transcoding time T_{total} is estimated using parallelism a as

$$T_{total} = (T_c + cF)\{(1-a) + \frac{a}{m}\} \tag{2}$$

$$a = \frac{cF}{cF + T_c} \tag{3}$$

The T_c is the sum of time taken to transmit the first video segment to the last PC (which is denoted by PC(m) in Fig.17, and the time taken to transcode overlapped 2GOP data as described in 3.2 and communication setup overhead. We assume the latter time is constant for each segment and denoted as T_{OH}, and assume that segment merging time can be neglected, because the merging time is relatively small compared to the transcoding time and the only few last segments contribute to T_c. Then, the T_c can be estimated as

$$T_c = \frac{dR_s m}{R_{demux}} + \frac{T_{OH}F}{dm} \tag{4}$$

where R_s and R_{demux} are source video coding rate in bits/ sec, demultiplexing speed in bits/sec, respectively. Therfore,using above equations,

$$T_{total} = \frac{dR_s m}{R_{demux}} + \frac{cF}{m} + \frac{T_{OH}F}{dm} \tag{5}$$

Fig. 18 Overview of the distributed transcoding system

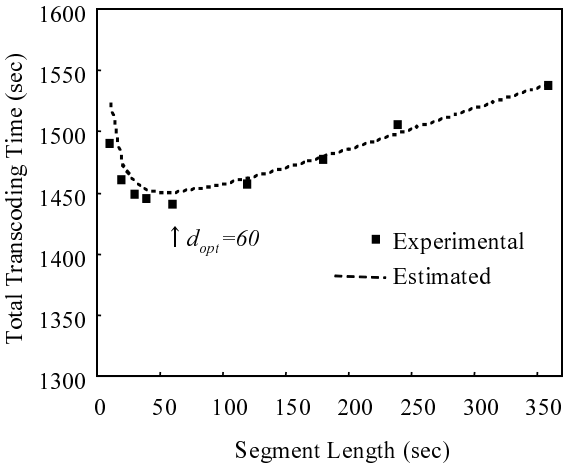


Fig. 19 Performance of the distributed transcoder using 5 transcoding PCs

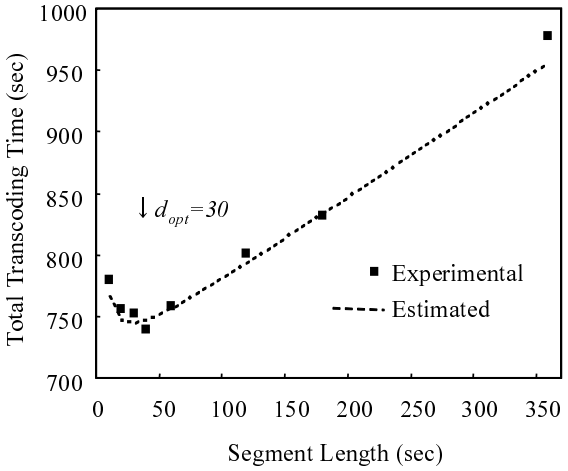


Fig. 20 Performance of the distributed transcoder using 10 transcoding PCs

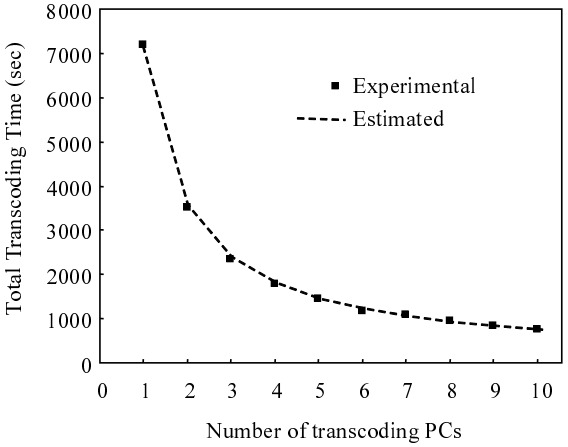


Fig. 21 Performance of the distributed transcoder

The optimum length of segment d_{opt} for minimizing the total transcoding time T_{total} can be calculated by differentiating (5) with respect to segment length d.

$$d_{opt} = \sqrt{\frac{R_{demux} T_{OH} F}{R_s M^2}} \tag{6}$$

5. Experimental Results

In order to verify the performance model and the optimum segment length, we conducted experiments using the one-hour football video introduced in the previous section and the transcoding parameters in Table 1. The overview of experimental system is shown in Fig. 18. All PCs had two 1.26GHz Intel's Pentium processors, and are connected by 1000Base-T LAN.

Figure 19 shows the total transcoding time for various segment lengths from 10 sec to 360 sec, using 5 transcoding PCs and Fig. 20 shows that of 10 transcoding PCs, respectively. Estimated performance and optimum segment length, calculated by Eq.(5) and Eq.(6) well predict the experimental results. The estimation used c, R_{demux} and T_{OH} values of 2.0, 120Mbit/sec and 1.5 sec, respectively; these values were obtained in a preliminary experiment. The performance of merging is 30 Mbit /sec to form each output video listed in Table 1. Figure 21 shows how the total transcoding time decreases with the number of transcoding PCs. In this experiment, each segment length was determined by (6).

With 10 PCs, the proposed transcoding system decreases the total transcoding time achieved with one PC by a factor of 7. While performance seems to saturate at about 10 processors due to the bottleneck of the de-multiplexing process in our current implementation, higher performance can be achieved by improving the demultiplexing algorithm and tuning operations such as like disk I/O. Also, to achieve higher performance, the optimum segment allocation taking into account of the fluctuation in the segment transcoding time needs to be investigated. Figure 22 shows

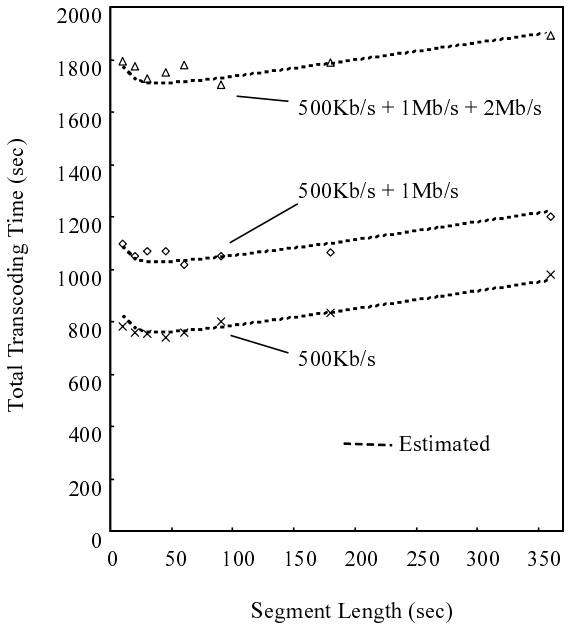


Fig. 22 Performance of the distributed transcoder for simultaneous multiple transcoding

the simultaneous multiple MPEG-4 transcoding performance using 10 transcoding PCs. The estimation used c is 2.0 for 500Kb/s, 2.6 for 500Kb/s + 1Mb/s, and 4.7 for 500Kb/s + 1Mb/s + 2Mb/s, calculated by Eq.(1), respectively. These results show that the distributed transcoding is efficient for speedup of multiple transcoding. If three kinds of transcoding are done on one transcoding PC, it will take about 16920 ($= 3600 \times 4.7$) seconds. The distributed transcoding on 10 PCs decrease the total time by a factor of about 9.5 ($=$ 16920/1780). The performance become improved for multiple transcoding compared with that of single transcoding. The performance improvement is because decoding process is done only once and eliminate the multiple decoding time. The reason that the estimated performance at short segment length doesn't well predict for multiple transcoding is because segment handling overhead time T_{OH} is assumed to be constant and merging time is neglected. These may cause underestimation of total segment overhead time.

6. Conclusion

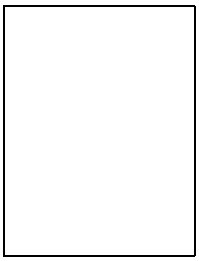
In this paper, we investigated high-speed distributed video transcoding for multiple rates and formats, using our proposed segment handling method and rate-control to ensure uniform transcoded video quality. Experimental results show the distributed transcoding system with 10 PCs decrease the total transcoding time by a factor of 7. More work needs to be done on a better video rate-control for variable bit rate encoding, and a better scheduling algorithm for unequal PC performance [13] and that can deal with PC failure during operation.

Acknowledgment

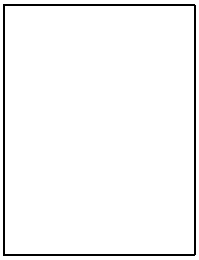
The Authors would like to thank Professor Masayuki Murata of Osaka Univ. for his valuable suggestions on distributed multimedia networks and Dr. Sakuichi Ohtsuka of NTT DATA corp. for his comments on video coding quality.

References

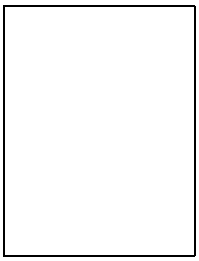
[1] ITU-T H.262/ISO-IEC 13838-2, MPEG-2, H.262, 1996.
[2] G. Morrison, "Video transcoders with low delay," IEICE Trans. Commun., vol.E80-B, no.6, pp.963–969, June, 1997.
[3] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," IEEE Trans. Multimedia, Vol. 2, No. 2, pp.101–110, June, 2000.
[4] Z. Lei and N.D. Georganas, "Rate adaptation transcoding for precoded video streams," Proc. ACM Multimedia02, pp.127–136, Juan-les-Pins, Dec., 2002.
[5] J. Youn, M.T. Sun and J. Xin, "Video transcoder architecture for bit rate scaling of H.263 bit streams," Proc. ACM Multimedia, pp.243–250, Orlando, Nov., 1999.
[6] J. Xin, M.T. Sun and B.S. Choi and K.W. Chun, "An HDTV-to-SDTV spatial Transcoder," IEEE Trans. Circuits & Systems Video Technology, vol.12, no.11, pp.998–1008, Nov., 2002
[7] Y. Nakajima and M. Sugano, "MPEG bit rate and format conversions for heterogeneous network/storage applications," IEICE Trans. Electron., vol.E85-C, no.3, pp.492–504, Mar., 2002.
[8] Y. Sambe, S. Watanabe, Dong Yu, T. Nakamura, N. Wakamiya, "A High Speed Distributed Video Transcoding for Multiple Rates and Formats," Proc. ITC-CSCC2003, pp.921–924, 2003.
[9] J.L. Hennesy and D.A. Patterson, "Computer architecture : A quantitative approach," Morgan Kaufmann Inc., 1990.
[10] Test Model 5, ISO/IEC JTC1/ SC29/ WG11/ N0400, MPEG93/457, April, 1993.
[11] P. Tiwari and E. Viscito, "A parallel MPEG-2 video encoder with look-ahead rate control," Proc. IEEE International Acoustics, Speech and Signal Processing Conf., vol.4, pp.1994–1997, 1996.
[12] R. Egawa, A. A. Alatan, and A. N. Akansu, "Compressed domain MPEG-2 video editing with VBV requirement," IEEE Proc. ICIP2000, 2000.
[13] Y. Sambe, S. Watanabe, Dong Yu, T. Nakamura, N. Wakamiya, "Distributed video transcoding and its application to grid delivery," IEEE Proc. APCC2003, vol.1, pp.98–102, 2003.



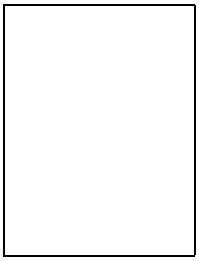
Yasuo Sambe received the B.E. and M.E. degrees from Osaka University, in 1984 and 1986, respectively. Since 1986, he has been with NTT Data Corporation and engaged in research and development of multimedia communication systems and video coding. He ia a member of IEICE, ACM and IEEE.



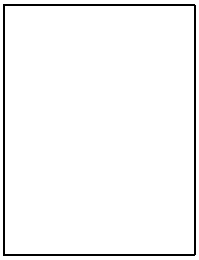
Shintaro Watanabe received his B.S. degree from Tokyo Institute of Technology in 1994 and M.E. degree from Nara Institute of Science and Technology in 1996. He has been working for Research and Development Headquarters at NTT DATA Corporation since 1996. His main interests are in image processing and video processing.



Dong Yu received the M.E. and Ph.D degrees from University of Tokyo, in 1994 and 1991, respectively. Since 1994, he has been with NTT Data Corporation and engaged in research and development of multimedia communication systems.



Taichi Nakamura received the B.E. and M.E. degrees from Chiba University, Japan, in 1972 and 1974, respectively. He received the Ph.D. degree from Hokkaido University in 1988. Since 1974, he worked with NTT Research Laboratory and engaged in research of image communications. Since 1988, he was with NTT DATA and engaged in research and development of multimedia systems. Since 2003, He is a Professor of the School of Computer Science, Tokyo University of Technology. He is a member of IEICE and IEEE.



Naoki Wakamiya received the M.E. and Ph.D. degrees from Osaka University in 1994 and 1996, respectively. He was a Research Associate of the Graduate School of Engineering Science, Osaka University, from 1996 to March 1997, and a Research Associate of the Educational Center for Information Processing, Osaka University, from 1997 to March 1999. He is an Assistant Professor of the Graduate School of Information Science and Technology, Osaka University, since April 1999. His research interests include performance evaluation of computer communication networks, and distributed multimedia systems. He is a member of IEICE, ACM and IEEE.