Analysis of Video Transcoding on Multi-core Platform

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

ABSTRACT

This paper analysis video segmentation methods for temporal and spatiotemporal video transcoding on a multi-core platform where video segmentation is performed at a group of pictures (GOP) level. Three methods of video segmentation are analysed: (1) equal number of group of pictures (GOP) in each segment (2) equal number of frames in each segment, (3) equal size segmentation. The existing single processor implementation of open source ffmpeg video transcoder is modified to execute on multiple cores. For coomunication among different processing cores the Message Passing Interface is used. We tested our implementation on a workstation which has two processors having four cores each with shared memory. The performance of the system and the relationship between the number of cores used and speed in computation is measured in terms of speed up of execution. The Master/ Slave processing model is selected in order to perform the transcoding operatons on seperate parts of the source video in parallel.

1. INTRODUCTION

In the past two decades there is significant advancement in the execution speed of programs due to faster hardware processing units. The serial code written for one hardware platform runs faster on another more powerful hardware platform. The programmer does not need to rewrite the program to get more speed up. But this trend seems to come on end due to physical limitations of processing unit.

Multi-core architectures are one of the available solutions to get more speed up in applications. The applications designed to execute on a single processing unit requires some changes in order to execute on multicore processors hence bringing a new challenge for programmers.

Video transcoding is a suitable application for multicore architectures. In video transcoding the compressed video needs to be processed to produce another compreesed video with different characteristics. The new video can have different bit rate, frame rate, frame resolution, or any combination of these. It is also possible to transcode a video in an entirely different video format. The video transcoder can be a cascaded pixel domain transcoder in which the entire video is decoded and then again re-encoded according to new requirements or it can be a pixel domain transcoder in which the video is partially decoded and where the motion vectors information are reused in order to reduce the computational load.

On a single core architecture the transcoding operation can be considered as a single task and will utilize all avaiable resources where the multicore processor will provide more processing resources for exectuing multiple transcoding tasks.

The main contribution of this work is to analyse different methods of video segmentation to perform temporal resolution reduction video transcoding and spatiotemporal resolution reduction video transcoding on multicore architectures. Three different methods of video segmentation are analysed (1) equal size segmentation (2) equal number of frames in each segment (3) equal number of group of pictures (GOP) in each segment. The performance of the system and the relationship between the number of cores used and speed in computation is measured in terms of speed up of execution.

2. BACKGROUND AND RELATED WORK

There exists a number of video transcoding applications mainly designed to execute on single processor. While transcoding the quality of the output video and the speed of the transcoding process are very important. The main focus of the work in [7] was on motion vectors downscaling for spatial resolution reduction transcoding. In [8], the drift error and source of drift error for spatial resolution reduction video transcoding was analyzed. The motion vector re-estimation in temporal resolution reduction video transcoding is discussed in [10].

In [6] a distributed video transcoder was designed to transcode MPEG-2 video into MPEG-4 video. The source MPEG-2 video was fully decoded and a MPEG-4 encoder was used to get the desired transcoded video. To get more performance in terms of speed up a cluster based distributed video transcoding approach was proposed in [3]. The main contribution of the work was the design of a multimedia web server for distributed video transcoding. The video contents were generated in accordance with the bit rate requirement.

The above transcoding architectures are either single processor or cluster based distributed architectures but our transcoding architecture is applicable for multicore architectures as well as distributed and cluster based architectures.

3. MASTER/SLAVE MODEL FOR PARALLEL PROCESSING

In the Master/Slave model the master is used to provide the centralized control and slaves performs the computation in the distributed environment. The scheduling of different

tasks and the resource allocation is performed by the master core. The master core also sends the source video to the slaves and recieves back the transcoded video. Different video frames types have different transcoding time, hence the load balancing is the main challenge in this model.

The processing core can have one or more tasks at a time. The MPI is used for sending messages between cores. The master core has the tasks of partitioning the source video, sending the video data, receiving the video data and combining the transcoded video. The slaves have the tasks of receiving the video data, transcoding the video and sending the transcoded video data back to the master. The message passing interface is suitable for multiple instructions multiple data systems [5].

4. VIDEO TRANSCODING

In this paper we focus on block based video transcoding in which the discrete cosine transform (DCT) and motion compensation (MC) are used. The video frames are divided into macroblocks (MBs) of size 16x16. The macroblocks can be of luminance or chrominance type. Each macroblock is further divided into 8x8 size blocks. Figure 1 shows frame rate reduction and frame resolution reduction transcoding.

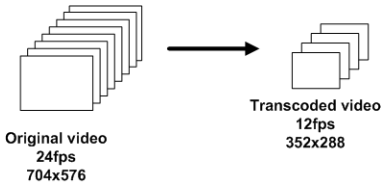

Figure 1: Frame rate reduction and frame resolution reduction transcoding

The source video needs to be downscaled to get reduced resolution frames. Downscaling is possible by decoding the entire video into raw format and then re-encoding it with the desired resolution and frame rate. Motion estimation requires 60% of the execution time [9] in the encoding process. It is also possible to reuse the existing motion vectors information and generate new motion vectors from the existing one. To generate new motion vectors both motion vectors downscaling and motion vectors re-estimation will be required [10]. Figure 2 shows the spatial resolution down conversion of the macroblocks.

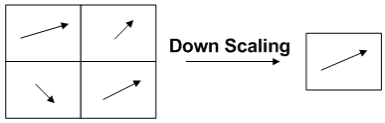

Figure 2: Spatial resolution down conversion

While downscaling a frame several 8x8 DCT blocks are downscaled into one 8x8 DCT block. In this paper every four macroblocks of the source video are downscaled into one macro block in the spatiotemporal video transcoding. Macroblock and motion vectors downscaling is further discussed in [4].

In temporal video transcoding some frames from the source video are dropped. Therefore the motion vectors pointing to the macroblocks in the dropped frames becomes invalid and motion vector re-estimation is needed. The re-estimation of a new motion vector can be done by tracing back all motion vectors of the dropped frames to the desired reference frame. Bilinear interpolation [10] of the motion vectors of the dropped frames can be used for this.

Figure 3 shows a case of motion vector re-estimation in which Frame (n-1) is dropped. the macroblock in Frame (n) needs a new motion vector from the non skipped frame (n-2). A new motion vector is predicted from existing motion vectors. The motion vector produced by re-estimation [10] will require vector refinement with a smaller search window to get the appropriate motion vector.

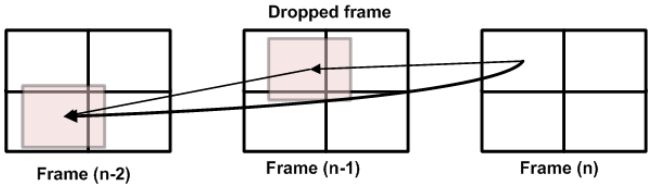

Figure 3: Motion vector reestimation

5. EXPERIMENTAL SETUP

The existing single processor implementation of the open source ffmpeg [2] video transcoder is modified to execute on multiple cores. For coomunication among different processing cores the Message Passing Interface is used. We tested our implementation on a workstation which has two quad-core Intel(R) Xeon(R) processors (E5430) at 2.66 GHz with 16 GB shared memory. Each core has 6144 KB cache.

The big buck bunny video sequence [1] was used as source video to perform transcoding operations. The table 1 shows further details about the video sequence.

Resolution	Rate	Size	Play time	Frames	GOPs
704 x 576	24fps	77.95 MB	09:56	14315	1279

Table 1: Big buck bunny video sequence

6. VIDEO SEGMENTATION

The source video consists of different types of frames (Intra, Predicted, Bi-directional predicted). The Intra frames are independent and do not require any other reference frame for decoding. To partition the source video into smaller parts, there must be an intra frame at the beginning of every segment. An intra frame followed by a sequence of P and B frames is termed a group of pictures (GOP). We performed the video segmentation at GOPs level with three different methods. The methods of video segmentation used are: (1) equal number of group of pictures (GOP) in each segment (2) equal number of frames in each segment, (3) equal size segmentation.

The minimum unit of video segment is a complete group of pictures (GOP). Tables 2, 3, and 4 shows the size of each video segment in Mega Bytes, total number of group of pictures in each segment, total number of frames in each segment for the three methods of video segmentation.

	1	2	3	4	5	6	7
size	77.95						
GOP	1279						
Frame	14315						
size	28.41	49.54					
GOP	656	623					
Frame	7167	7148					
size	19.35	20.36	38.24				
GOP	440	426	413				
Frame	4774	4781	4760				
size	15.01	13.4	18.62	30.92			
GOP	335	321	320	303			
Frame	3590	3577	3578	3570			
size	12.46	10.14	13.19	14.31	27.85		
GOP	273	253	257	255	241		
Frame	2867	2868	2863	2859	2858		
size	10.70	8.65	9.06	11.30	11.53	26.71	
GOP	231	209	216	210	213	200	
Frame	2386	2388	2393	2388	2380	2380	
size	9.95	6.75	7.88	8.64	10.13	9.59	25.01
GOP	202	177	184	186	179	180	171
Frame	2051	2042	2047	2050	2042	2051	2032

Table 2: Equal number of frames in each segment

	1	2	3	4	5	6	7
size	77.95						
GOP	1279						
Frame	14315						
size	38.98	38.97					
GOP	841	438					
Frame	9280	5035					
size	26.03	25.94	25.98				
GOP	600	495	184				
Frame	6536	5591	2188				
size	19.49	19.48	19.7	19.28			
GOP	445	396	321	117			
Frame	4832	4448	3651	1384			
size	15.61	15.6	15.62	15.6	15.52		
GOP	347	356	268	201	107		
Frame	3734	3951	3000	2366	1264		
size	13.0	13.03	12.95	12.99	13.05	12.93	
GOP	284	316	241	254	84	100	
Frame	2987	3549	2744	2847	1008	1180	
size	11.15	11.14	11.13	11.14	11.16	11.16	11.07
GOP	244	270	239	189	205	37	95
Frame	2530	3061	2647	2137	2376	444	1120

Table 3: Equal size segmentation

7. RESULTS

We performed the video transcoding operation using multiple cores starting with a single master and a single slave core to a single master and seven slave cores. Each slave core recieves part of source video for transcoding. The transcoding time for temporal resoluation reduction with different number of core is shown in Figure 4. The source video has 4CIF (704x576) resoluatoin, 24 fps and it is transcoded at 16fps and 8fps.

The transcoding time for spatiotemporal resoluation reduction with different number of core is shown in Figure 5. The

	1	2	3	4	5	6	7
size	77.95						
GOP	1279						
Frame	14315						
size	27.65	50.3					
GOP	639	640					
Frame	6977	7338					
size	18.76	20.55	38.64				
GOP	426	427	426				
Frame	4624	4791	4900				
size	14.42	13.27	18.14	32.12			
GOP	319	320	320	320			
Frame	3405	3572	3587	3751			
size	11.51	10.68	12.4	15.04	28.32		
GOP	255	256	256	256	256		
Frame	2662	2904	2840	2892	3017		
size	10.24	8.59	8.92	11.62	11.61	26.97	
GOP	213	214	214	214	214	210	
Frame	2178	2458	2365	2431	2383	2500	
size	8.78	7.48	7.78	8.68	8.88	10.5	25.85
GOP	182	183	183	183	183	183	182
Frame	1813	2130	2045	2021	2072	2070	2164

Table 4: Equal number of GOPs in each segment

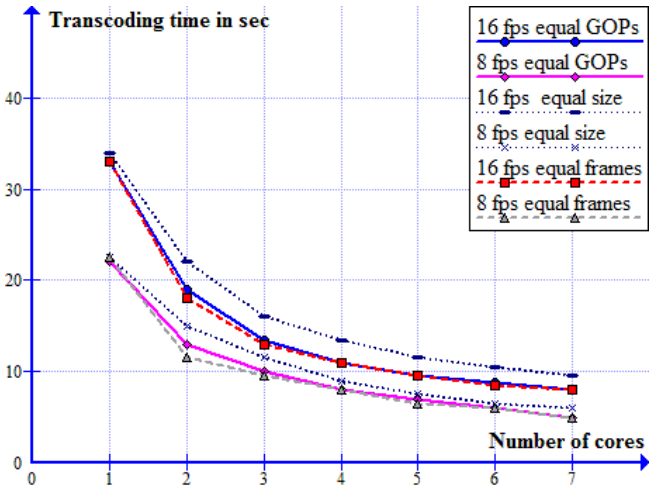

Figure 4: TemporalTranscodingTime

source video has 4CIF (704x576) resoluatoin, 24 fps and it is transcoded down to CIF (352x288) resolution at 24fps and 16fps.

The increase in number of cores gives speedup in the transcoding processes. To analyse different methods of video segmentation, the speedup obtained with multiple cores is considered. The speed up obtained with different segmentation methods for temporal resolution reduction transcoding is shown in Tables 5, 6 and for spatiotemporal resolution reduction in Tables 7, 8.

The results in tables 5, 6, 7, and 8 indicates that the video segmentation performed at equal number of frames provides better speed up. The video segmentation with equal number of group of pictures (GOP) in each segment gives almost similar speed up as we get with equal number of frames in each

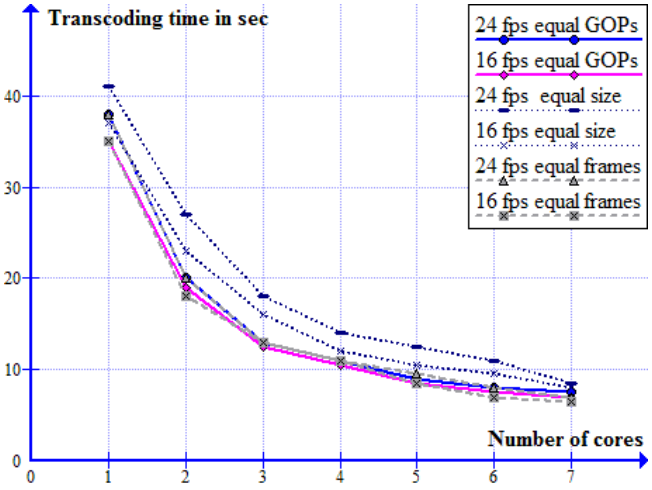

Figure 5: Video Transcoding time with different number of cores and frame rates

No of slave cores	2	3	4	5	6	7
equal size	1.54	2.12	2.52	2.96	3.24	3.58
equal No of GOP	1.73	2.44	3	3.47	3.75	4.13
equal No of frames	1.83	2.54	3	3.47	3.88	4.13

Table 5: speedup: temporal transcoding 16fps

No of slave cores	2	3	4	5	6	7
equal size	1.5	1.96	2.5	3	3.46	3.75
equal No of GOP	1.69	2.2	2.75	3.14	3.67	4.4
equal No of frames	1.93	2.37	2.81	3.46	3.75	4.5

Table 6: speedup: temporal transcoding 8fps

No of slave cores	2	3	4	5	6	7
equal size	1.52	2.28	2.93	3.28	3.72	4.82
equal No of GOP	1.9	2.92	3.45	4.22	4.75	5.06
equal No of frames	1.9	2.92	3.45	4	4.75	5.43

Table 7: speedup: spatiotemporal transcoding 24fps

No of slave cores	2	3	4	5	6	7
equal size	1.61	2.31	3.08	3.52	3.89	4.62
equal No of GOP	1.84	2.8	3.33	4.11	4.66	5
equal No of frames	1.94	2.69	3.18	4.11	5	5.38

Table 8: speedup: spatiotemporal transcoding 16fps

segment, hence both methods are better than the equal size segmentation. While transcoding a video, different frames require different time, the intra (I) frames require more computational power than predicted (P) and bi-directional predicted (B) frames. For different video sequences the equal size segmentation will not partition the video in eqaul computational load hence it is prefered to use either equal number of frames or equal number of GOPs methods for spatiotemporal video transcoding. The equal number of frames will be better than equal number of GOPs for temporal video transcoding.

8. CONCLUSION AND FUTURE WORK

In this paper the implementation of video transcoding system on multi-core platform using the Master/ Slave model for parallel processing is discussed. For video segmenation three methods are used: (1) equal number of frames in each segment (2) equal size segmentation (3) equal number of group of pictures (GOP) in each segment. The communication among different processing units is perofrmed with standard Message Passing interface and data is sent by send and recieve messages. The masters core segments the source video and sends it's parts to slaves (workers) cores for transcoding and after recieveing back the transcoded video parts it performs the merging task. It is observed that with seven worker cores the transcoding operation is five times faster as compared with single worker node.

The segmentation method with equal number of frames in each segment provides better speed up as compared with the other two methods. More analysis can be performed with different types of video transcoding (bit rate resolution and spatial resolution reduction) on multicore architectures using these segmentation methods. It is also possible to have segmentation with equal number of slices or equal number macroblocks in each segment by keeping the GOP as the smallest segmenation unit for multicore architectures.

9. REFERENCES

[1] Big buck bunny video sequence, http://www.bigbuckbunny.org/index.php/download/.

[2] Ffmpeg project, http://www.ffmpeg.org/.

[3] 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*, page 8 pp., april 2003.

[4] Z. Lei and N. D. Georganas. H.263 video transcoding for spatial resolution downscaling. In *ITCC*, pages 425–430. IEEE Computer Society, 2002.

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

[6] Y. Sambe, S. Watanabe, D. Yu, T. Nakamura, and N. Wakamiya. High-speed distributed video transcoding for multiple rates and formats. *IEICE Transactions*, 88-D(8):1923–1931, 2005.

[7] D. Shen, I. K. Sethi, and B. Vasudev. Adaptive motion-vector resampling for compressed video downscaling. *IEEE Trans. Circuits and Systems for Video Technology*, 9(6):929, Sept. 1999.

[8] P. Yin, A. Vetro, B. Liu, and H. Sun. Drift compensation for reduced spatial resolution transcoding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(11):1009 – 1020, nov 2002.

[9] P. Yin, M. Wu, and B. Liu. Video transcoding by reducing spatial resolution. In *ICIP*, 2000.

[10] J. Youn, M.-T. Sun, and C.-W. Lin. Motion vector refinement for high-performance transcoding. *IEEE Transactions on Multimedia*, 1(1):30–40, 1999.