ARCHITECTURE OVERVIEW OF MEDIANET MULTIPROCESSOR TRANSCODER

Javed I. Khan and S. S. Yang

Technical Report 2000-08-01

Internetworking and Media Communications Research Laboratories
Department of Math & Computer Science
Kent State University, 233 MSB, Kent, OH 44242
(Revised October 11, 2000)
javed@kent.edu

1 Active Processing Speed and Video Transcoding

Application level transcoding of video at network junction points can be breakthrough application for active networks—if not the killer. However, video transcoding is a computationally daunting task by its own virtue [4,7]. It becomes more so when we plan to perform it in-stream in real time inside on a network.

Current state-of-the art in video compression technology requires custom chipset to obtain real-time performance in MPEG-2 encoding . Roughly speaking, something close to CIF video (CCIR 4:2:0 CIF=352x240 at 30 frames/second or 352 x288 at 25 frames/second) can be decoded in software satisfactorily. It requires processing of 69,300 blocks per second. In comparison, a broadcast quality video (CIR-601 4:2:2: 720x480 at 30 frames/second or 352x576 at 25 frames/second) requires processing of 405,000 blocks per second, while a production or medical quality video (MPEG-2 HIGH@HIGH-1440 profile=1920x1152x 60 frames/second) will require processing of 5,184,000 blocks per second.

Active network [1, 5, 6] based transcoding adds further computational challenge to the above scenario. First of all, a Transcoder is composed of a decoder and an encoder. Thus, it is computationally a more complex task. Secondly, on an active network the available processors are expected to be general purpose with general programmability. Which almost excludes the possibility of using any custom chip set (although field programmable logic may still be an option). Finally, an active network processor, by definition and location, is going to be a highly shared resource. The competing processes will also be real time processes. Thus, not only its processing performance, but also the complexity of the Transcoder itself has to be highly optimized.

As a first approach, in this goal, we are investigating course parallelization of the decoding/ coding architecture. It seems, that it is not unreasonable to expect commodity low cost parallel system with 2-4 processors. In this system we demonstrate how MPEG-2 stream [2,3] can be transcoded in parallel in a multiprocessor system.

2 Overview of Task Segmentation

In our model, we first identified the major computation clusters based on computational complexity and internal data dependency of full logic MPEG-2 transcoding. Accordingly we have separated the sub-processes involved. We then redesigned our transcoder architecture on these cluster boundaries, and re-implemented the transcoder with self-contained and co-operating sub-processes. The design goal we wanted to meet are (a) ability to process stream, (b) minimum dependency between the processes and (c) natural scalability.

Since, GOP is a natural self-content data unit in a MPEG-2 stream therefore, we segmented the task along GOP boundaries. Also, we segmented the tasks along the line of decoding and encoding operations. Two additional units were added- GOP demultiplexer (GOP-MAX) and GOP multiplexer (GOP-DEMUX). MPEG-2 decoding is about +4 times faster than encoding operation. Therefore, on a 2-4 processor target architecture, we decided to concatenate the task of stream decoding into a single process. GOP MUX/DEMUX operations are O(p), there fore, we further aggregated GOP-DEMUX/MUX operation with the Decoder. Fig-1 below shows an overview of a parallel transcoder architecture.

3 Notes on Implementation

Below we describe the frame-wise GOP reorganization. As shown in the figure 2, MPEG video stream consists of GOPs with headers. It starts with a sequential header followed by GOPs. In each GOPs, there are frames which represent image frame data in three different forms (I, P, and B.) A GOP has a separate information group from others, so it is independent.

3.1 Process Modules

Consequently, we built two sub-processes: (a) X-Stream-Processor, (b) X-GOP-Coder. X-Stream Processor is responsible for decoding, demultiplexing the incoming stream, parameter extraction, and multiplexing. While GOP-Coder works as a depdent process which accepts the frames and encodes them per GOP basis.

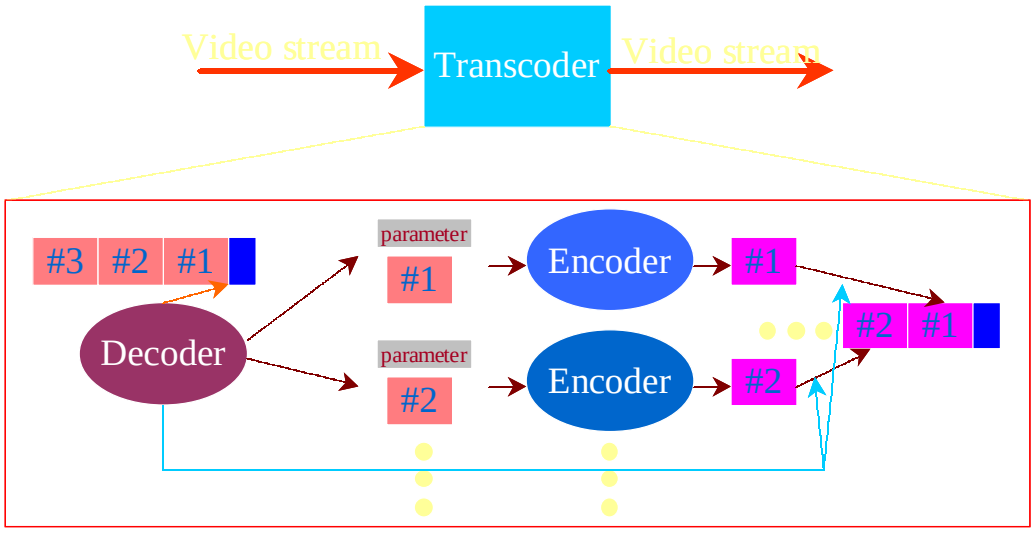

Figure 1 Parallel Transcoder Architecture

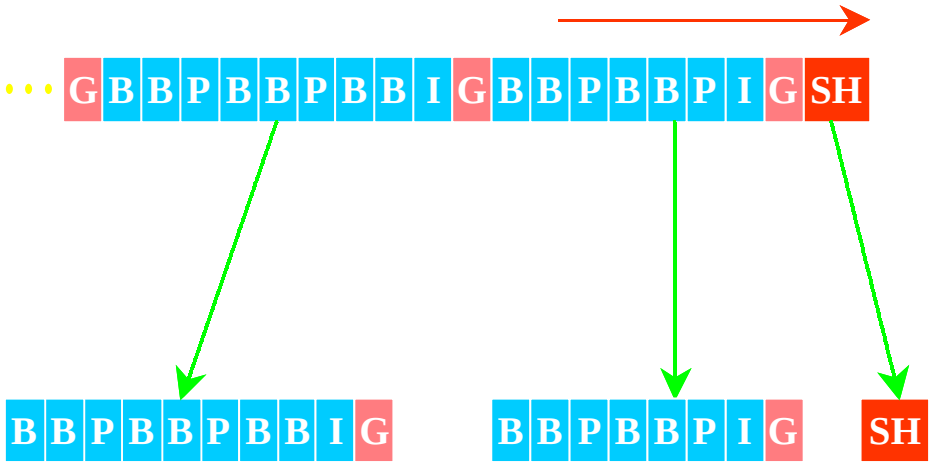

Figure 2 Separation of each GOP

3.2 Operation

X-Stream-Processor first decodes the incoming frames from the input video stream. It also extracts required encoding parameters into a parameter from the incoming stream for parameter passing. One a full GOP is decoded the Stream-Processor invokes a GOP-Coder. GOP-Coder is slow, therefore with each GOP decoding a new GOP-coder is invoked until the maximum running GOP-coder is p-1. GOP-Coder reads the parameter bus to obtain all the coding parameters. It also checks for the information in the parameter file if it is the first GOP data. If it is the first GOP, then the coder generates a MPEG-2 Sequence Header. Every GOP-coder also generates the GOP header. Once, the Coder completes processing the frames it returns the frames to the Stream-Processor. Finally, the transcoded data from concurrent GOP-Coders is gathered by the Stream Processor. The MUX unit then combines the GOPs and completes the outgoing video stream.

3.3 Synchronization and Processor Scheduling:

We implemented the system as separate processes. It can be run with default OS kernel scheduling. However, if needed the IRIX PSET command set can be used to schedule the processes to specific processors. We have tested the system with LINUX RedHat 6.1 PSET implementation. For default scheduling, the decoder invokes a GOP-Coder and wait until the GOP-Coder is done. Sure, the wait does not block decoding of other GOPs and invoking other GOP-Coders. After a GOP-Coder ends, the decoder sends its output video stream to a final output video output stream.

[1] Bhattacharjee, S., Kenneth L. Calvert and Ellen W. Zegura. An Architecture for Active Networking. High Performance Networking' 97 ,White Plains, NY, April 1997. [also available at http://www.cc.gatech.edu/projects/canes/papers/anarch.ps.gz, October 98]

[2] Haskell B. G., Atul Puri and Arun Netravali, Digital Video: An Introduction to MPEG-2, Chapman and Hall, NY, 1997.

[3] Information Technology- Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC International Standard 13818-2, June 1996.

[4] Keesman, Gertjan; Hellinghuizen, Robert; Hoeksema, Fokke; Heideman, Geert, Transcoding of MPEG bitstreams Signal Processing: Image Communication, Volume: 8, Issue: 6, pp. 481-500, September 1996,

[5] Tennenhouse, D. L., J. Smith, D. Sincoskie, D. Wetherall & G. Minden., "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, Jan 97, pp 80-86

[6] Wetherall, Guttag, Tennenhouse, "ANTS: A Tool kit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, April 1998. Available at: http://www.tns.lcs.mit.edu/publications/openarch98.html

[7] Youn, J, M.T. Sun, and J. Xin, "Video Transcoder Architectures for Bit Rate Scaling of H.263 Bit Streams," will beappeared to 'ACM Multimedia 1999', Nov., 1999. pp243-250.