H.26L Video Server Modeling Using Computational Process Networks

Serene Banerjee

Dept. of ECE, The University of Texas, Austin, TX 78712-1084 USA

{serene}@ece.utexas.edu

Abstract

Recent applications like video conferencing, Digital Versatile Disc (DVD) systems, digital cable television, and video streaming, have promoted an impressive growth of the digital video research. State-of-the-art video coding solutions such as H.263, MPEG-2, and MPEG-4 have a common goal of achieving higher compressed video quality for lower bit-rates. The International Video Coding Experts' Group (VCEG) is standardizing the next-generation video-coding standard, H.26L. This will reduce the bit-rate by about 50%, at the expense of increased (3.8X) complexity. Computational Process Networks (CPN) is a scalable framework for real-time data-intensive systems' implementation on commodity multiprocessor workstations. This is an extension of the process networks model, which captures parallelism, guarantees determinate execution in bounded memory. In this project we propose to implement a version of the H.26L video encoder on the CPN framework.

1 Introduction

The International Telecommunication Union (ITU) is proposing the new video coding standard, H.26L, which is aimed at providing enhanced compression performance at a very low bit rate, real-time, low end-to-end delay mobile applications [1–8]. The model provides bit rate savings of about 50% or more compared to a performance optimized H.263+ codec, for most sequences at low bitrates. Also, improved network adaptation layers (NAL) is being developed so that the coded video data can be transported in existing and future network, such as circuit-switched wired networks, IP networks with RTP packetization, and 3G wireless systems. But this increase in performance is at the cost of several times increase in complexity, compared to the H.263+ codec. It takes up to 10 seconds to code a single 176×144 QCIF picture using the available unoptimized software [8]. We propose to exploit inherent data level parallelism of the H.26L encoder and implement a system level modeling for faster performance, using Computational Process Networks (CPN). Allen and Evans [9] implement a real-time beamformer on multiple workstations, and He and Zhong [10] implement MPEG-4 encoder on a similar model. He, Ahmad and Liou implement a MPEG-4 [11, 12] codec on a group of workstations using a hierarchical Petri-nets-based model.

2 H.26L Standard

H.26L video-coding standard, separates the video server design into two separate layers: a *video coding layer* that is responsible for efficiently representing the video content, and a *network adaptation layer* that packs the coded data in an appropriate manner based on the network it is being transmitted. Our focus will be on the video coding layer. Figure 1 shows a generic block diagram of a video compression codec, for example an H.263, MPEG-2, or MPEG-4 simple

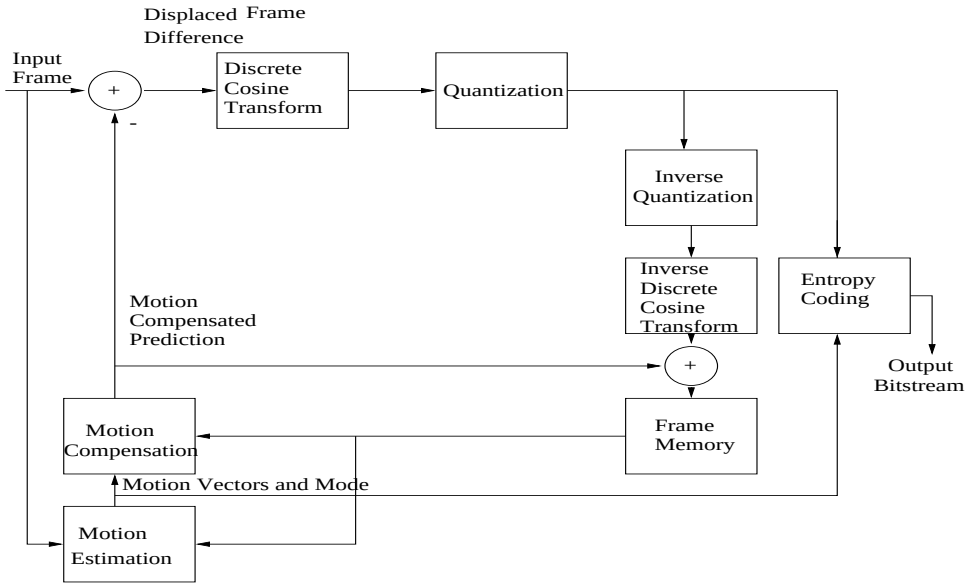

Figure 1. Generalized block diagram of a hybrid block-based motion compensation and transform coding video encoder.

profile codec, where the encoded bitstream reduces the spatial and temporal statistical redundancy existing in video sequences. The following innovative features in the H.26L encoding standard improve performance over the state-of-the-art models at the cost of complexity increase:

- **Motion representation:** Depending on the context, the block sizes for Intra (I), Inter (P), and Bi-directional (B) motion prediction can be 16×16, 8×16, 16×8, 8×8, 4×8, 8×4, and 4×4 pixels. Thus multihypothesis pictures can be generated for H.26L [4] producing bit-rate savings up to 13% for test video sequences. The standard supports $\frac{1}{4}$ and $\frac{1}{8}$ sample motion vector accuracy. Multi-frame motion-compensated prediction is also supported. Also, for motion representation, a predictive switching between different video streams or between different parts of a single video stream is also supported with the inclusion of SP-pictures.

- **Transform processing:** After finding the best matching block, the prediction block is subtracted from the original block to produce a residual image signal. A two-dimensional Discrete Cosine Transform (DCT) given by Equation 1 is applied to the 8×8 residual image block so that the energy is compacted in small number of transform domain coefficients.

$$y_{kl} = \frac{c(k)c(l)}{4} \Sigma_{i=0}^{7} \Sigma_{j=0}^{7} x_{ij} \cos\left(\frac{(2i+1)k\pi}{16}\right) \cos\left(\frac{(2j+1)l\pi}{16}\right) \quad (1)$$

$$\text{where } k, l = 0, 1, ..., 7 \text{ and } c(k) = \begin{cases} \frac{1}{\sqrt{2}} \text{ if } k = 0 \\ 1 \text{ otherwise} \end{cases}$$

However, as the DCT is defined in terms of floating point values, H.26L approximates the DCT using a reversible integer transform. To extend the length of the basis functions for smooth regions, a 4×4 and 2×2 integer transform is applied to the DC values of the luminance and chrominance data, respectively. Also, for improved rate control capability a variable quantization step sizes is supported, instead of fixed step sizes.

- **Entropy coding and coefficient scanning:** Universal Variable-Length Coder (UVLC) and Context-Adaptive Binary Arithmetic Coder (CABAC) are supported. The code table generated for UVLC can be represented in the following compressed form:

$$1$$

$$0 \quad x_0 \quad 1$$

$$0 \quad x_1 \quad 0 \quad x_0 \quad 1$$

$$0 \quad x_2 \quad 0 \quad x_1 \quad 0 \quad x_0 \quad 1$$

$$0 \quad x_3 \quad 0 \quad x_2 \quad 0 \quad x_1 \quad 0 \quad x_0 \quad 1$$

where x_n take values 0 or 1. Thus, a codeword can be uniquely represented by its length in bits (L) and INFO = x_n ... x_1 x_0. UVLC code table is customized to the probabilistic behavior of the data. CABAC adapts to local source statistics and exhibits extremely efficient entropy coding, especially for very fine or very coarse grain quantization. For CABAC, a different model is maintained for each of the syntax elements (e.g., motion vectors and transform coefficients have different models).

- **Deblocking filter:** Block-edge artifacts are reduced using in-loop filter applied to the horizontal and vertical edges.

- **Intra prediction:** Directional prediction in the spatial domain is used providing greater efficiency than coefficient value prediction in the transform domain.

For the *Foreman* input sequence at 24kbps, the unoptimized encoder takes 84% time for motion estimation, 4% for image interpolation and rest 12% for the other functions. For a highly optimized version, these figures are 57%, 9%, and 34% respectively [5].

3 Process Networks

For formal system-level modeling statically schedulable Synchronous Dataflow (SDF) model, or dynamically schedulable Boolean Dataflow (BDF), or Dynamic Dataflow (DDF) may be used to provide portability and scalability over heterogeneous software environments to guarantee determinacy and correctness [13]. Kahn's process network is a superset of dataflow models and is a computational model where concurrent processes are connected by unidirectional first-in-first-out (FIFO) queues. The results obtained from a process network are determinate, irrespective of the

execution order in the model. However, determining that the process network can be scheduled in bounded memory cannot be predicted on a finite amount of time. Parks [13] developed a scheduling policy that will yield bounded execution if one exists.

However, when a queue is full, *artificial deadlocks* are introduced, which can be avoided by increasing the queue size in case one occurs. Allen and Evans [9] added queue-firing thresholds to form computational process networks (CPN). These queues are equivalents of modulo addressing units on digital signal processors (DSP). Their implementation of CPN using C++ and portable operating system interface (POSIX) threads is intended for computationally intensive algorithms on large symmetric multiple workstations. However, with context switching between the nodes and amount of load on each node, tradeoffs between overhead, latency and parallelism will exist. The H.26L video encoder will be ported on the C++ implementation of their CPN framework available at *http://www.ece.utexas.edu/~allen/PNSourceCode/*. This implementation of CPN is scaled by the operating system according to the available number of processors.

4 Parallel Image/Video Processing

He and Zhong [12] model a MPEG-4 simple visual profile video encoder, with block diagram similar to Figure 1, using CPN. Each functional block is modeled as a CPN node and implemented as a Pthread class. Each arc is represented by a FIFO queue, and fork nodes are implemented so that inputs are copied to multiple outputs. Deadlock is avoided by having an arbitrary initial token on a delay arc, before executing the motion estimation and motion compensation blocks. They show that their concurrent implementation of MPEG-4 on the average is about 30% faster than a sequential implementation of MPEG-4 encoder.

He, Ahmad and Liou [11, 12] implement a parallel processing version of an MPEG-4 video codec using hierarchical Petri-nets-based model. This is a graphical and mathematical tool for describing concurrent, asynchronous, distributed, nondeterministic and parallel characteristics of systems. They have experimented with three types of scheduling algorithms to assign video objects to workstations in parallel, so that synchronization requirements are enforced and presentation deadlines are met. They are:

- **Round-robin schedule:** Video objects are assigned to workstations sequentially.
- **Group schedule:** Divides the workstations in groups and each group encodes a video object plane concurrently.
- **Group of video adjusting schedule:** For video objects whose size change rapidly, tasks are merged to obtain load balancing.

Each algorithm while showing real-time encoding rates for MPEG-4, exhibits load balancing, scheduling overhead cost and global performance tradeoffs. Their results show that with 20 workstations, the MPEG-4 encoder can have an encoding rate higher than real time for CIF resolution (352×288) video sequences, allowing the encoding of multiple sequences simultaneously.

In the Petri-nets-based modeling [11, 12] real-time performance is dependent on the scheduling and partitioning algorithms, but not on the modeling part. However, in the CPN based modeling [9, 12] the memory, latency, and scheduling issues are automatically resolved by the operating system, thus providing a unified approach and better scalability.

On a single workstation, the concurrent execution of CPN nodes in He and Zhong's [12] model outweighs overheads created by C++/Pthread programming. However, they have not experi-

mented on a multiple workstation platform, by refining each CPN node and balancing the computational load among them. For a single processor, varying the FIFO queue size did not produce any significant reduction in the execution time. However, as their model is scalable to a multiprocessor platform, real-time performance will be achieved by varying the FIFO queue sizes on a CPN model with load-balanced nodes.

Feil, Kutil, Meerwald, and Uhl [14] show data parallelism involved in wavelet based image and video codecs. Patrick, Sanders, DeBrunner, DeBrunner, and Radhakrishnan [15] implement a JPEG codec via parallel processing, where each operation is performed on a different DSP.

5 Proposed Work

Like the present video-coding standards, H.26L also has inherent data level parallelism, and can be hence modeled using formal system-level computational models. Allen and Evans' [9] Computational Process Networks model is a scalable framework suitable for implementing computationally intensive algorithms on a cluster of workstations. The goals of this project are:

- Use computational process networks to model the data flow of the H.26L video encoder, to provide real-time encoding, determinacy, correctness, completed and bounded execution.
- Implement a version of the H.26L video based on the Pthread library and Allen and Evans' [9] C++ implementation of Computational Process Networks
- Compare complexity, cost and performance tradeoffs with varying the FIFO queue sizes and the number of processors.

References

[1] G. Bjontegaard, "H.26L Test Model Long Term No. 1 (TML-1) Draft 2," Tech. Rep. 1, ITU-T Video Coding Experts Group, Aug. 1999. *ftp://standard.pictel.com/video-site/9908_Ber/q15h36d2.doc.*

[2] A. P. Joch, "H.26L: Analysis and Real-Time Encoding Algorithms," Master's thesis, The University of British Columbia, Jan. 2002.

[3] K. Dovstam, "Video Coding in H.26L," Master's thesis, Royal Institute of Technology, Apr. 2000. *http://www.it.kth.se/docs/Reports/DEGREE-PROJECT-REPORTS/000601-Kristofer-Dovstam.pdf.*

[4] M. Flierl, T. Wiegand, and B. Girod, "Multihypothesis Pictures for H.26L," in *Proc. IEEE Int. Conf. on Image Proc.*, vol. 3, pp. 526–529, Oct. 2001.

[5] A. Hallapuro, V. Lappalainen, and T. D. Hamalainen, "Performance Analysis of Low Bit Rate H.26L Video Encoder," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol. 2, pp. 1129–1132, May 2001.

[6] V. Lappalainen, A. Hallapuro, and T. D. Hamalainen, "Optimization of Emerging H.26L Video Encoder," in *Proc. IEEE Workshop on Signal Proc. Systems*, vol. 1, pp. 406–415, Sept. 2001.

[7] G. J. Sullivan, T. Wiegand, and T. Stockhammer, "Using the Draft H.26L Video Coding Standard for Mobile Applications," in *Proc. IEEE Int. Conf. on Image Proc.*, vol. 3, pp. 573–576, Oct. 2001.

[8] S. Wenger, "A High Level Syntax for H.26L: First Results," in *Proc. IEEE/SPIE Int. Conf. on Visual Comm. and Image Proc.*, vol. 4067, pp. 1307–1316, June 2000.

[9] G. E. Allen and B. L. Evans, "Real-Time Sonar Beamforming on Workstations Using Process Networks and POSIX Threads," *IEEE Trans. on Signal Proc.*, vol. 48, pp. 921–926, Mar. 2000.

[10] Y. He, I. Ahmad, and M. L. Liou, "Real-time Interactive MPEG-4 System Encoder Using a Cluster of Workstations," *IEEE Trans. on Multimedia*, vol. 1, pp. 217–233, June 1999.

[11] Y. He, I. Ahmad, and M. L. Liou, "A Software-Based MPEG-4 Video Encoder Using Parallel Processing," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 8, pp. 909–920, Nov. 1998.

[12] C. He and S. Zhong, "System Modeling and Software-based Implementation of MPEG-2 Video Encoder," in *Proc. IEEE Asilomar Conf. on Signals, Systems and Comp.*, vol. 2, pp. 1058–1062, Oct. 2000.

[13] T. M. Parks, *Bounded Scheduling of Process Networks.* PhD thesis, University of California at Berkeley, Dec. 1995. *http://ptolemy.eecs.berkeley.edu/papers/95/parksThesis/.*

[14] M. Feil, R. Kutil, P. Meerwald, and A. Uhl, "Wavelet Image and Video Coding on Parallel Architecture," in *Proc. IEEE Int. Symp. on Image and Signal Proc. and Analysis*, vol. 1, pp. 24–35, June 2001.

[15] J. S. Patrick, J. L. Sanders, L. S. DeBrunner, V. E. DeBrunner, and S. Radhakrishnan, "JPEG Compression/Decompression via Parallel Processing," in *Proc. IEEE Asilomar Conf. on Signals, Systems and Comp.*, vol. 1, pp. 596–600, Nov. 1997.