REAL-TIME DISTRIBUTED AND PARALLEL PROCESSING FOR MPEG-4

Yong He[†], *Ishfaq Ahmad*[‡], *Ming L. Liou*[†]

[†]Department of EEE, [‡]Department of Computer Science,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong

ABSTRACT

MPEG-4 is currently being developed by MPEG to specify the technologies for supporting current and emerging multimedia applications. Because of its object-based features and flexible toolbox approach, it is much more complex than previous video coding standards. We believe that software-based implementation on parallel and distributed computing systems is a natural and viable option. In this paper, we describe such an approach on the MPEG-4 video encoder using a cluster of workstations. We propose to use hierarchical Petri Nets as a modeling tool to describe the temporal relations and time constrains among various video objects at different levels. This would allow us to perform scheduling with a guarantee of synchronization among multiple objects. A dynamic shape-adaptive data parallel approach is used in the spatial domain for further speed-up gain. Our preliminary results indicate that real-time MPEG-4 encoding using distributed and parallel computing is achievable.

1. INTRODUCTION

MPEG-4 is currently being developed by MPEG and is scheduled to become an international standard in Nov. 1998. During the development, MPEG-4 is gaining immense popularity since the technological elements specified by MPEG-4 will enable a broad range of current and emerging multimedia applications. The objective of MPEG-4 is to integrate the production, distribution and content access of digital objects which can be natural or synthetic [1]. With a flexible toolbox approach, MPEG-4 is capable of supporting diverse new functionalities and satisfying various application requirements on different aspects. In addition, due to its extensible system configuration architecture, MPEG-4 is aimed to be more compatible with advanced new technologies.

MPEG-4 video is one of the major part of MPEG-4. In contrast to the previous video coding standards such as H.261, MPEG1/2 and H.263, MPEG-4 video is an object-based hybrid natural and synthetic video coding standard focusing on the content-based interactivity, coding efficiency, and universal access [2]. Figure 1 is the overall structure of MPEG-4 video codec. The video encoder is composed of identical VOP (video object plane) encoders. Each object is segmented from the input video signal and performs the same encoding scheme separately, the bitstream of different VOPs are then multiplexed and transmitted. At the decoder, the received bitstream are demultiplexed and decoded by each VOP decoder, the

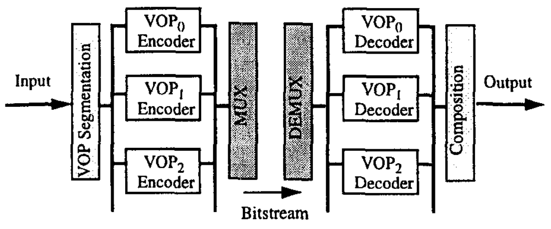


Figure 1. MPEG-4 Video Codec Structure

reconstructed video objects are composited and presented to the user. The user interaction with the objects such as scaling, dragging, replacement and linking can be handled either on the encoder or on the decoder.

Two video object profiles, *Simple* and *Core,* have been defined with a collective useful tools respectively addressing for different applications [3]. For most real-time applications such as real-time communications, surveillance and live broadcast, video compression requires a great deal of processing power and therefore the encoder should be highly efficient.

A number of implementation of H.261, MPEG1/2 and H.263 have been reported based on software ([4], [5] and [6]) as well as hardware ([7], [8]). While MPEG-4, due to its content-based representation nature and flexible configuration structure, is much more complex than previous standards. ASICs or specially developed video signal processors maybe cost effective, however, not flexible enough for MPEG-4 applications. No MPEG-4 hardware is available at present and any such implementation is likely to be very much application specific. We believe that software-based implementation using parallel and distributed computing systems is a natural and viable option.

In this paper, we will show the implementation of MPEG-4 encoder using a software-based approach on a cluster of workstations. The rest of the paper is arranged as follows: Section 2 describes the proposed methodology to model and represent MPEG-4 audio-visual information using the Petri nets. Section 3 presents a real-time scheduling algorithm for scheduling various sub-tasks of the encoder. Section 4 shows a dynamic data partitioning scheme for further speed-up gain. Section 5 provides the preliminary experimental results of our approach. The last section concludes the paper by providing an overview of our ongoing research.

0-7803-4455-3/98/$10.00 © 1998 IEEE

2. PETRI NETS MODELING

MPEG-4 video is object-based standard which may deal with various video objects during the video session. In the parallel implementation of MPEG-4 video encoder, one of the most important issues to consider is the synchronization of various video objects. Each object may have certain presentation timing constrains and may be dependent on the other objects. In order to identify the time constraints among multiple objects, a synchronization reference model should be established to describe the various presentation requirement and temporal relationship for determining an appropriate scheduling scheme. There are several modeling tools proposed in [9] for specifying the temporal behaviour of various multimedia systems. We choose Petri nets as the modeling tool since it is a promising tool for describing and studying the systems with both concurrent and sequential activities.

A Petri net is a graphical and mathematical modeling tool for describing concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic systems [10]. It has been widely used in manufacturing and automation applications, computer networks and multimedia communication applications due to its intuitive graphical representation and the simplicity of the modeling concepts.

Petri nets are defined as 4-tuple (P,T,A,M) where:

$P = (p_1, p_2, p_3, \dots, p_m)$ is a set of places,

$T = (t_1, t_2, t_3, \dots, t_m)$ is a set of transitions,

$A \subseteq \{T \times P\} \cup \{P \times T\}$ is a set of arcs,

$M: P \rightarrow I, I = (0, 1, 2, 3, \dots)$ associates a marking to each place in the net.

Since the modeling structure may become much complex for a complicated video session. We propose a hierarchical Petri net which can be used to refine the system behavior in a step-by-step fashion. The structure definition is similar to the data structure of MPEG-4 video which consists of video session (VS) level, video object (VO) level, video object layer (VOL) level, and video object plane (VOP) level (for the sake of simplicity in our implementation, we consider only VS, VO, and VOP levels).

For example, Figure 2 is the time chart representation for a general case of MPEG-4 video session. The session has 4 video objects (VOs); VO_0, VO_1, VO_3 start at time 0, and VO_2 starts at time unit 8. VO_0 and VO_1 end at time 8, while VO_2 and VO_3 end at time 16. The frame rates of VOs are different. The duration of a frame for VO_1 is 1 time unit, while the duration of a frame is 2 for VO_2 and VO_0 and 4 for VO_3. To describe the time constraints of this video session, a three level hierarchical Petri net model is established (see Figure 3). The first VS level represent the whole video session, the second VO level indicates the timing relation among multiple video objects, and third VOP level describes the intra and inter synchronization for each VOP. With such model we can achieve coarse or fine synchronization by applying a scheduling scheme on different levels to satisfy various levels of service quality.

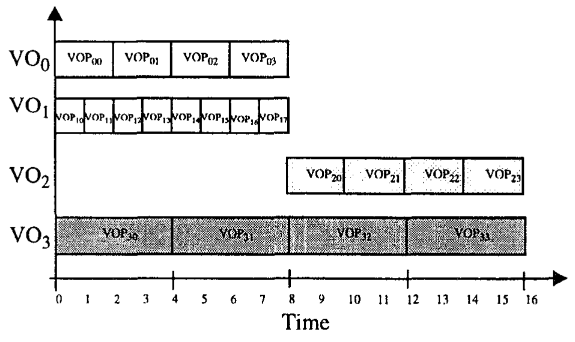
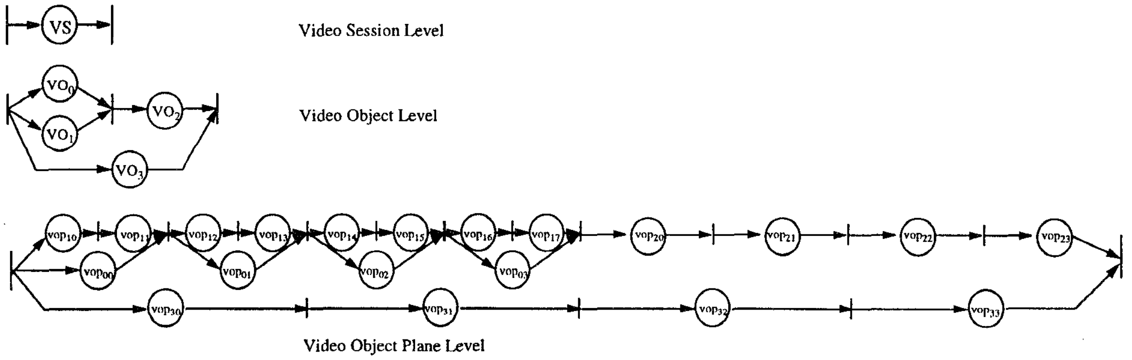


Figure 2. Time chart for video objects presentation

3. REAL-TIME SCHEDULING SCHEME

The objective of real-time scheduling is to assign the tasks to the available processors and determine the execution order of each task so that tasks are completed before their deadlines [11]. In our implementation, we have designed a hybrid static and dynamic scheduling scheme which can adapt the VOPs variation with minimum scheduling time cost. We perform the scheme on each period at VO level and length of the period depends on the availability of objects. The knowledge of video objects can only be known after observing a time period and the characteristics of these objects within the period are relatively stable. Therefore, we can schedule each period statically with the knowledge beforehand and adapt the video object changes dynamically.

A number of scheduling algorithms have been developed for both distributed and centralized systems [12]. In our



VS
Video Session Level
VO_0
VO_1
VO_2
VO_3
Video Object Level
vop_{10} vop_{11} vop_{12} vop_{13} vop_{14} vop_{15} vop_{16} vop_{17}
vop_{00} vop_{01} vop_{02} vop_{03}
vop_{20} vop_{21} vop_{22} vop_{23}
vop_{30} vop_{31} vop_{32} vop_{33}
Video Object Plane Level

Figure 3. Hierarchical Petri net model

implementation, we use a variant scheduling of the *earliest-deadline-first* (EDF) algorithm which has been widely employed by many applications. The principle of such algorithm is that the tasks with an earlier deadline value are assigned a higher priority and run before tasks with lower priority. Therefore, in our implementation, the VOPs with the earlier playout deadline or synchronization point get to be encoded and delivered first (Figure 4 shows the Petri net model of scheduled VOP execution order with the fine synchronization at VOP level). For these tasks with the same deadline, we assign a portion of existing processors to each object and the allocation quantity depends on the size ratio among these objects because a video object with a larger size generally requires more computing and vice versa.

4. SHAPE-ADAPTIVE DATA PARTITION

According to the Petri net model of EDF schedule scheme which determines the encoding sequence of various VOPs, a variety of approaches can be applied for data-parallelism within a VOP. We propose a shape-adaptive data parallel paradigm for further speed-up gain.

Data partitioning is to decompose the whole frame data into a number of data blocks and map these blocks onto the corresponding processors. Because the processors of the parallel program perform the computation on their local memory and run the program on the data blocks simultaneously, a high speedup can be achieved. Since most encoding algorithms are performed based on macroblocks, the size of partitioned data blocks should be the integer number of macroblocks to reduce the interprocessor communication overhead.

In MPEG-4, a VOP is represented by a bounding rectangle containing the whole object with the minimum numbers of the macroblocks which means a VOP window contains both the pixels inside or outside the object. Most algorithms employed by MPEG-4 video encoder are performed mainly on the pixels inside the object and consequently the computational load does not distribute uniformly within the VOP windows. Therefore, an appropriate tasks distribution is necessary for balancing the workload among processors.

Most data partitioning methods restrict the subregion to be rectangular blocks to avoid a messy problem of the data structure. For MPEG-4, some subregions of the VOP window may be full of the pixels inside the object while others may be full of the pixels outside. Therefore, no partitioning method can equidistribute rectangular subregion in a straightforward way. In addition, the object size may become too tricky to do the stripwise or boxwise partition.

Here, we developed a dynamic shape-adaptive data partitioning method to guarantee the workload balancing during the whole video session with low runtime overhead and fine granularity.

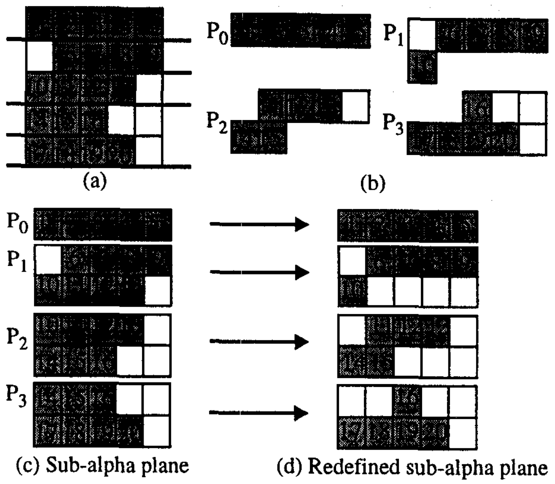


Figure 5. Shape-adaptive partitioning example

As implied in Figure 5, the gray macroblocks represent the object macroblocks which contains the object pixels while white ones represent the transparent macroblocks which are totally outside the object. The object macroblocks are equally assigned to a given number of processors. As Figure 5 (a) illustrates, there are 20 object macroblocks within the window and each processor is assigned 5 object macroblocks. Because each processor (P_0 to P_3) may get arbitrarily shaped subregions as Figure 5 (b) shows, it may cause messy data structure problem and become more complex to specify for parallel programming. Here, we extend these subregions to rectangulars called sub-alpha plane as Figure 5 (c) shows. Since some of the sub-alpha planes contain macroblocks which are redundant, we redefine the sub-alpha planes by labelling those macroblocks as transparent macroblocks in order to avoid unnecessary computation (Figure 5 (d)). For example, Processor 3 (P_3) should only encode the subregion which includes the object macroblock from 16 to 20 as shown in Figure 5 (b). In order to get a rectangular subregion which contains those blocks, we extend this subregion and the whole sub-alpha plane contains the object macroblocks from 14 to 20. Then we define the 14th and 15th macroblock as the transparent macroblock to form a redefined sub-alpha plane as shown in Figure 5(d). Therefore, processor 3 still processes 5 object macroblocks while keeps the subregion rectangular.

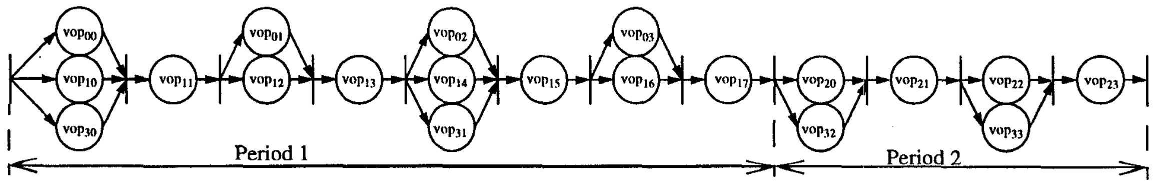


Figure 4. Petri net model for EDF schedule

Such partition adjustment should be performed dynamically whenever the VOP size changes. Thus it is possible to guarantee the load balancing during the whole session and a high processor efficiency can be achieved.

5. EXPERIMENTAL RESULTS

The proposed software-based parallel approach on MPEG-4 video VM encoder has been implemented and tested on a cluster of Sun UltraSparc-I workstations connected by a Fore Systems ATM switch. We performed the experiments using 1, 2, 4, 8, 12, 16 and 20 workstations on the MPEG-4 test sequences with QCIF resolutions. In our implementation, a fast motion estimation algorithm [13] was used to speedup the computation and the resultant visual quality is still very close to the case with full search. All of the preprocessing, including format conversion and bounding, is done off line.

Figure 6 shows the encoding frame rates using different numbers of workstations. The performance is close to real-time by using 20 workstations for most sequences. Figure 7 shows the plots of overall speedup ratios.

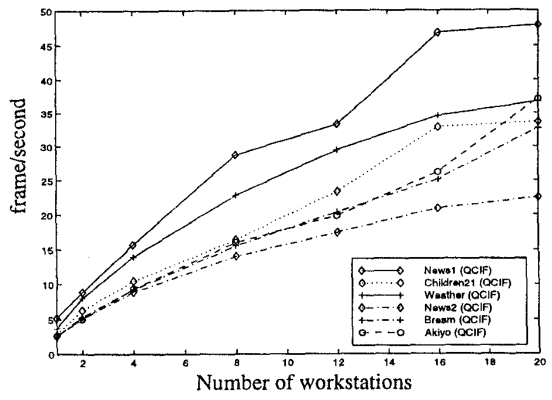


Figure 6. Encoding frame rate

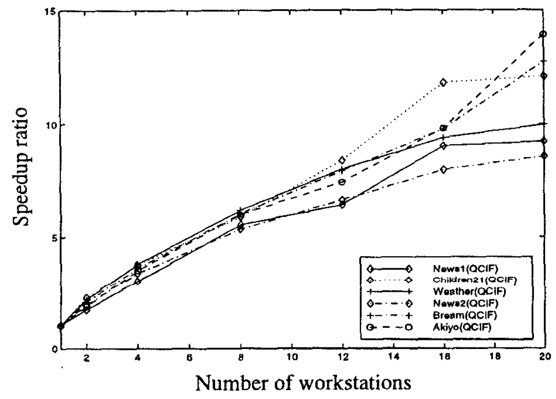


Figure 7. Overall speedup

6. CONCLUSIONS

In this paper, we have presented an implementation of MPEG-4 video verification model encoder using a cluster of workstations. The preliminary results indicate that real-time MPEG-4 encoding using distributed and parallel computing is achievable. In our implementation we have used a hierarchical Petri net model to capture the spatio-temporal relations between multiple objects of MPEG-4 video, an dynamic shape-adaptive data parallel scheme, and an hybrid real-time scheduling scheme to implement MPEG-4 video encoder. In our future work, there are a lot of things we can do to speed up the computation and make it practically useful, e.g. the granularity of scheduling, the scheduling efficiency and intelligent data partition scheme. Furthermore, a major issue to be explored is to incorporate the interactivity with the encoding processing which in turn should affect the scheduling decisions.

7. ACKNOWLEDGMENTS

This work was supported by the Hongkong Telecom Institute of Information Technology.

8. REFERENCES

[1] "Overview of MPEG-4 Version 1 Standard," *ISO/IEC/ JTC1/SC29/WG11 N1909,* Oct. 1997.

[2] T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Trans. on CSVT, vol.7, no.1, pp.19-31,* Feb. 1997.

[3] "Decisions on MPEG-4 Profiling," *ISO/IEC/JTC1/SC29/ WG11 N1908,* Oct. 1997.

[4] S. M. Akramullah, I. Ahmad, M. L. Liou, "Performance of a Software-Based MPEG-2 Video Encoder on Parallel and Distributed Systems," *IEEE Transactions on CSVT,* vol. 7, no. 4, pp. 687-695, Aug. 1997.

[5] S. M. Akramullah, I. Ahmad, M. L. Liou, "A Software Based H.263 Video Encoder using Network of workstations," *Proceedings of SPIE,* vol. 3166, Aug. 1997.

[6] Jongho Nang, Junwha Kim, "An Effective Parallelizing Scheme of MPEG-1 Video Encoding on Ethernet-Connected Workstations," *Proceedings. Advances in Parallel and Distributed Computing,* pp. 4-11, March 1997.

[7] D. Kim, *et al.,* "A Real-time MPEG Encoder using a Programmable Processor", *IEEE Transactions on Consumer Electronics* vol.40, no.2, pp. 161-70. May, 1994.

[8] H.A. Chow, H. Alnuweiri, "An FPGA-based Transformable Coprocessor for MPEG Video Processing," *Proceedings of the SPIE,* vol. 2914, pp. 308-320, 1996.

[9] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE Journal on Selected Areas in Communications,* vol. 8, pp. 413-427, Apr. 1990.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE,* vol. 77, no. 4, pp. 541-580, Apr. 1989.

[11] P. Michael, "Scheduling: Theory, Algorithms and System", Prentice Hall, 1995

[12] S. Cheng et al., "Scheduling Algorithms for Hard-Real Time Systems - a Brief Survey", *Hard Real-time Systems,* IEEE Computer Society press, 1988.

[13] Z. L. He and M. L. Liou, "A High Performance Fast Search Algorithm for Block Matching Motion Estimation," *IEEE Trans. on CSVT, vol.7, no.5, pp 826-828,* Oct. 1997.