A Shape-Adaptive Partitioning Method for MPEG-4 Video Encoding

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
Fax: [†](852) 2358-1485, [‡](852) 2358-1477
Email: [†]*eehey@ee.ust.hk*, [‡]*iahmad@cs.ust.hk*, [†]*eeliou@ee.ust.hk*

Abstract

MPEG-4 is a new standard for multimedia applications. Due to the flexible and extensible features of MPEG-4, the software-based implementation seems to be a natural and viable option. While such approaches usually require huge computing power, we can overcome such problem by using parallel and distributed processing. Because the behaviour of MPEG-4 objects may vary with time and such variation can not be predicted in advance, the issues of data partition and load balancing of the multiprocessor systems need to be addressed carefully in order to achieve real-time operation performance. In this paper, we propose a shape-adaptive data partitioning method to guarantee the load balancing among the multiprocessor systems. The effectiveness of this method has been demonstrated by the experimental results.

1. Introduction

MPEG-4 is an ISO/IEC standard being developed by MPEG to enable the integration of the production, distribution and content access paradigms in multimedia environment [1]. With a flexible toolbox approach, MPEG-4 can support a broad range of existing and emerging interactive multimedia applications, such as real-time communications, surveillance and live broadcast.

Due to the object-based nature and flexible toolbox approaches, MPEG-4 is much more complex than existing video standards. 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 is a natural and viable option because of its flexibility, portability, scalability, and its ability to allow the inclusion of new tools [2]. While for most real-time applications, video encoding usually requires huge amount of computing power which is the major problem with the software approaches. We can solve such problem by using parallel and distributed processing.

During the entire MPEG-4 video session, the size and shape of each object may vary from time to time. Moreover, some computationally intensive algorithms of the encoder are data-dependent, which means the execution time can not be predicted in advance. In order to achieve high efficient encoding performance, the issues of data partitioning and load balancing within a parallel computing system need to be addressed carefully.

We have proposed some software implementation of MPEG-4 video encoder [3]. In order to further speed-up the computing, we developed a shape-adaptive data partitioning method to guarantee the load balancing as well as to improve the global performance within a parallel computing system.

The rest of the paper is arranged as follows: Section 2 gives a brief overview of MPEG-4 video encoder. Section 3 introduces several load balancing partitioning methods. Section 4 proposes a shape-adaptive data partitioning scheme used in our testbed. Section 5 provides the experimental results of our approach. The last section concludes the paper by providing an overview of our ongoing research.

2. Overview of MPEG-4 Video

MPEG-4 video is one of the major part of MPEG-4. It is an object based hybrid natural and synthetic video coding standard enabling the functionalities such as content-based interactivity, efficient compression, error resilience and random access [4].

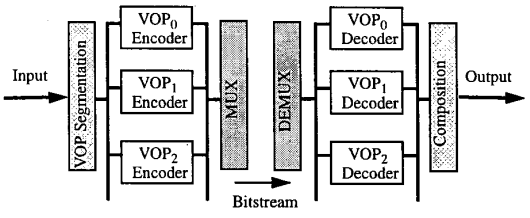

Figure 1. MPEG-4 Video Codec Structure

Figure 1 is the overall structure of MPEG-4 video codec. The video encoder is composed of identical VOP (video object plane) encoders and so is the decoder. The same coding scheme is applied to each video object separately

0-7803-5008-1/98/$10.00©1998 IEEE.

and the reconstructed video objects are composited together and presented to the user.

In order to encode the arbitrarily shaped VOPs, MPEG-4 defines the "VOP window" as the tightest rectangular of the VOP with the minimum number of macroblocks to represent the VOP. There are three kinds of macroblock (MB) within the VOP window, as depicted in Figure 2, the transparent MB, the contour MB and the standard MB. The contour and standard macroblocks include the pixels belonging to the VOP image, and transparent MB lies completely outside the object.

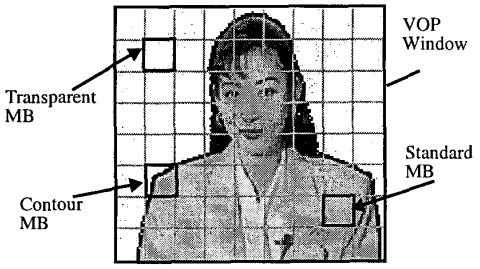

Figure 2. VOP window example for AKIYO

Each VOP encoder consists of three main parts: the shape coder, motion estimation/compensation, and texture coder. Most algorithms adopted by the encoder are macroblock-based.

Shape information which is referred to as the *alpha plane* is used in MPEG-4 to indicate the time-varying shape and location information of the VOPs. Shape coder performs the compression to the alpha plane. Since the transparent MB does not include any object pixels, it will not be processed for the motion and/or texture coding.

Motion estimation and compensation techniques are used to efficiently reduce the temporal redundancies. Due to the arbitrary shapes of VOP, repetitive padding technique is applied to the contour MBs of the reference VOP, then a block-based motion estimation is performed to both the contour and standard MBs of the current VOP. SAD (Sum of Absolute Difference) is used as the error measure due to its lower computational complexity. SAD is calculated only on the pixels inside the object, and is given by:

$$SAD_N(x, y) = \sum_{i=1, j=1}^{N, N} |orignal - previous| * (!(Alpha \equiv 0))$$

The intra and residual data after motion compensation of VOPs is coded by texture coding algorithms including DCT or shape adaptive DCT (SA-DCT), MPEG or H.263 quantization, intra DC and AC prediction, and VLC to achieve further compression. For those contour MBs, the padding technique is employed again.

The details of the encoder can be found in [5].

3. Load balancing techniques

As mentioned earlier, software-based approach is feasible for MPEG-4 video applications. But the computational requirement of a software-based encoder is simply too enormous to be handled by a single processor. It is, therefore, natural to harness the accumulated computational power offered by a high-performance parallel or distributed system. In addition, the MPEG-4 video codec structure happens to be very suitable for parallel and distributed processing. As the encoder is much more complex than the decoder, it is more challenging to speed-up the computation for the encoder.

Due to the object-based nature of MPEG-4 video, the size and location of each object may vary with time, and such behaviours can not be predicted beforehand. Therefore no matter how initial tasks are assigned, the workloads of the processors will become unbalanced later on, which will cause some processors to be highly loaded while others are idle or lightly loaded. Furthermore, some computationally intensive algorithms of the encoder are data dependent and their execution time are different to different data region. For example, some algorithms are performed on all macroblocks while others just acted on contour and standard MBs. Thus the problem of load balancing should be addressed carefully in the parallel processing in order to achieve real-time video encoding.

Since the video and image usually consist of a large amount of data, data parallel paradigm is often used in such a large data sets [6]. The main idea of data parallel is to decompose the whole frame data into a number of data blocks and mapping each block to certain processor, thus the processors can run the program simultaneously and a high speed-up can be achieved.

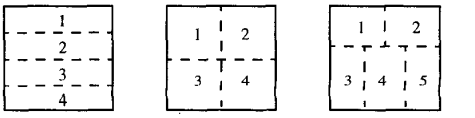

Figure 3. Common partitioning methods

Figure 3 shows several commonly used partitioning methods. Stripwise partition divides the whole VOP window horizontally or vertically into n subregions for n processors. It is easy to determine the area of subregions for corresponding processors. Boxwise partition divides the VOP window evenly along both the horizontal and vertical dimensions. The number of boundary pixels of the subregion is minimum, but the number of processors to be used is restricted and not suitable for heterogeneous systems. Recursive bisection method divides the whole VOP window recursivly in binary fashion [7]. It is capable to optimally equidistribute the computational load, while it is relatively expensive to execute the recursive

operations during the decomposition.

4. A periodical shape-adaptive partition

Due to the unpredictable variation of MPEG-4 objects, any simple and static partitioning scheme will cause workload imbalances which result in lower overall performance. A dynamic partition scheme can handle the indeterministic behaviour of the system, but it depends on the trade-off between the balancing quality and overhead runtime cost [8]. In our implementation, since the partition must be done in real-time, the cost of partition and redistribution must be kept low to ensure that the benefits gained from an efficient parallelization are not negated by a long time taken by the partitioning method.

In order to adapt object variations and minimize partitioning cost, we developed a shape-adaptive data partition method to guarantee the workload balancing during the whole video session with low runtime overhead and fine granularity.

First, the entire MPEG-4 video session is defined as a number of time intervals. The time interval boundary depends on the variation of the VOP window size. A new time interval begins whenever a VOP window changes above a certain threshold. Since the knowledge of the video objects can be obtained at the beginning of the interval, we then perform the shape-adaptive partition within each time interval. During that interval, we can assume that the spatial computation distribution is relatively stable and no need to change partitions. Therefore, the proposed load balancing can handle the object variation with minimum overhead runtime. Since most of the algorithms are macroblock-based, we employ macroblock-based data partition to map an integer number of macroblocks to each processor and enable the compression algorithm to be done locally.

Most data partitioning methods restrict the subregion to be rectangular blocks to avoid a messy problem of the data structure [7]. For MPEG-4, when the object is large enough and almost fill the VOP window, these methods may achieve good load balancing because the contour and standard MBs are likely to be distributed uniformly among multiprocessors. While in general cases, some subregions of the window may be full of transparent MBs while others may be full of contour and/or standard MBs. 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 present a shape-adaptive partitioning method whose subregions may have arbitrary shape, and the rectangular sub-alpha plane is further redefined to avoid the unnecessary computation for each processor.

As implied in Figure 4, the gray blocks represent the contour and standard MBs while white ones represent transparent MBs. By using the alpha plane information, we can get the statistical distribution of the contour and standard MBs. Then they are equally assigned to a given number of processors. As Figure 4 (a) illustrates, there are 20 contour and standard MBs within the window and each processor is assigned 5 contour and standard MBs. Because each processor (P_0 to P_3) may get arbitrarily shaped subregions as Figure 4 (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-AlphaPlane as Figure 4 (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 MBs in order to avoid unnecessary computation (Figure 4 (d)). For example, Processor 3 (P_3) should only encode the subregion which includes the contour and standard macroblock from 16 to 20 as shown in Figure 4 (b). In order to get a rectangular subregion which contains those blocks, we extend this subregion and the whole sub-alpha plane contains the contour and standard MBs from 14 to 20. Then we define the 14th and 15th MB as the transparent MB to form a redefined sub-alpha plane as shown in Figure 4 (d). Therefore, processor 3 still processes 5 contour and standard MBs while keeps the subregion rectangular.

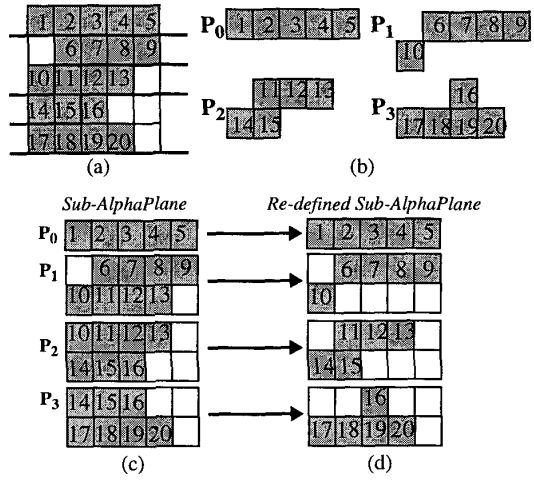

Figure 4. Arbitrary partitioning example

Because such a partition is based on macroblock decomposition, the granularity is low and the method can yield the finest workload balancing among the multiprocessors. In addition, by keeping the data block for each processor rectangular, it is easy for parallel decoders to compose the reconstructed subVOPs together since the syntax definition of the bitstream assume that the input VOP window is rectangular.

5. Experimental results

The proposed partitioning scheme has been implemented and tested on a cluster of Sun UltraSparc-I workstations connected by a Fore Systems ATM switch. A fast motion estimation algorithm [9] was used to speed-up the computing and still keeps the high visual quality.

Figure 5 shows the encoding rates for different MPEG-4 video test sequences by using various numbers of workstations.

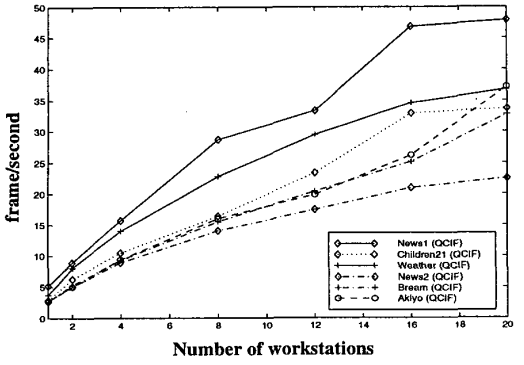

Figure 5. Encoding frame rate

Figure 6 is the comparison between the static stripwise/ boxwise partition, object-based partition [3] and our proposed method for the test sequence '*Children2*' with QCIF format. It is clear that a high real-time performance has been achieved by our method.

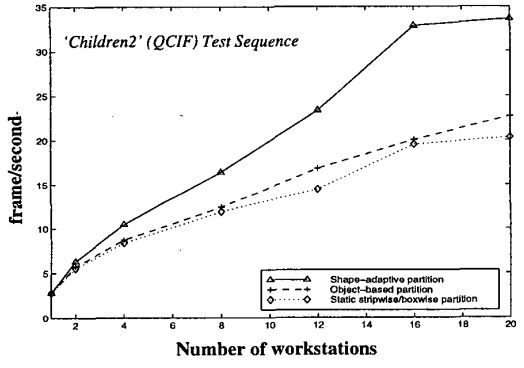

Figure 6. Partitioning performance comparison

Figure 7 is the comparison of the average partitioning time cost among three partition methods. It indicates that our proposed method can improve the coding efficient with low overhead partitioning runtime cost.

6. Conclusions

In this paper, we propose an arbitrary shape-adaptive partitioning method for MPEG-4 encoding using a cluster of workstations. The experimental results indicate that real-time MPEG-4 encoding using distributed and parallel

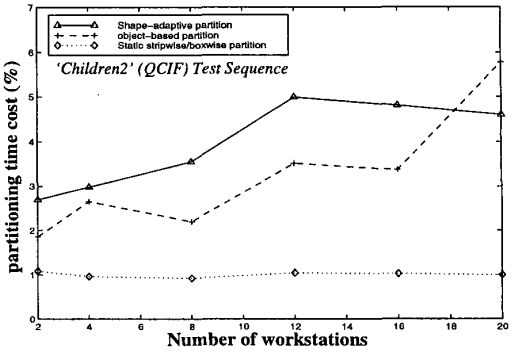

Figure 7. Partitioning time cost comparison

computing can be achieved with a careful load balancing strategy. In our future work, we are exploiting dynamic load balancing algorithms for heterogeneous computing environment for MPEG-4 applications.

7. Acknowledgments

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

References

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

[2] 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.

[3] Yong HE, Ishfaq Ahmad and Ming L. Liou, "An Implementation of MPEG-4 Video Verification Model Encoder using Parallel Processing," *Proceedings of the third Asia-Pacific Conference on Communications, pp.56-59,* Dec. 1997

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

[5] ISO/IEC, "MPEG-4 Video Verification Model Version 8.0," ISO/IEC JTC1/SC29/WG11 N1796, July 1997.

[6] Shen Ke, G.W Cook, L.H Jamieson and E.J Delp, "An Overview of Parallel Processing approaches to image and video compression," *Proceedings of SPIE, vol.2186 pp197-208,* 1994.

[7] Marsha J. Berger and Shahid H. Bokhari, "A partitioning Strategy for Nonuniform Problems on Multiprocessors," *IEEE Trans. on Computers, vol. C-36, no.5, pp.570-580,* May 1987.

[8] Y. Zhang, H. Kameda and S.L. Hung, "Comparison of dynamic and static load-balancing strategies in heterogeneous distributed systems," *IEE Proc.-Comput. Digit. Tech., vol.144, no.2, pp100-106,* March 1997.

[9] 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.