MPEG-4 Based Interactive Video using Parallel Processing

YONG HE[1], ISHFAQ AHMAD[2] AND MING L. LIOU[1]

[1]Department of Electrical and Electronic Engineering

[2]Department of Computer Science

The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

Abstract[†]

MPEG-4 which is currently being developed by MPEG (Moving Pictures Experts Group), is poised to become a standard for supporting current and emerging interactive multimedia applications. The objective of MPEG-4 is to support content-based compression, communication, access and manipulation of digital objects which can be natural or synthetic. Since MPEG-4 based video consists of objects and provides full interactivity between the client and the server, a software-based implementation seems to be the only viable approach for building an MPEG-4 encoder. Parallel processing solves the problem of large computational requirements for building a real-time encoder.

In this paper, we describe a parallel implementation of MPEG-4 video encoder using a cluster of workstations collectively working as a virtual machine. Parallelization of the MPEG-4 encoder poses an interesting problem since not only can objects be added or deleted from a video scene but their sizes and shapes may vary with time. Moreover, some of the computationally intensive parts of the encoder are non-uniform algorithms, which means their execution times are data dependent and cannot be predicted in advance. In order to guarantee the spatio-temporal relationship between various objects in a video, we propose a real-time scheduling algorithm for exploiting parallelism in the temporal domain. The algorithm divides the workstations into a number of groups and assigns one video object to one group of workstations for encoding. A dynamic shape-adaptive data partitioning strategy is proposed to exploit parallelism in the spatial domain. The partitioning strategy divides the data of an object among the workstations within a group. The scheduling scheme ensures the synchronization requirements among multiple objects while the dynamic data parallel approach adapts to the object shape variations to balance the load for all the workstations. The performance of the encoder can scale according to the number of workstations used. With 20 workstations, the encoder yields an encoding rate higher than real-time, allowing to encode multiple sequences simultaneously.

Keywords: MPEG-4, video compression, distributed and parallel processing, data partitioning, scheduling, MPI.

1 Introduction

With the development of workstations and networking technologies, the aggregated computing power of a cluster of workstations can match that of an expensive parallel computing system [6]. Because of the advantages such as scalable file storage, large memory, high performance-cost ratio, and efficient communication hardware/software support, many current parallel applications can use the cluster of workstations as the platform instead of parallel machines.

On the other hand, recently, there has been a technological revolution in the area of multimedia-based information technology. Progress in information technology is now recognized to be essential for the success of industrial and commercial businesses as well as for improving the quality of life for the masses in general. A majority of present and future multimedia-based applications require huge computing power. For instance, video is a fundamental component of multimedia systems, and the storage and transmission of large amount of required data inevitably calls for the compression and decompression of digital video. Video compression, if done through software, requires considerably extensive computing power than that offered by a single PC or workstation. Parallel processing then becomes a natural approach. The latest developments in cluster computing offer a higher degree of performance at an affordable cost (such as a network of workstations), provided the parallelism from the application at hand is effectively extracted and scheduling and load balancing schemes are properly designed.

MPEG-4, currently being developed by MPEG (Moving Picture Experts Group) [9], is expected to be finalized towards the end of this year. It will become a standard for compression, transmission, and presentation of current and emerging interactive multimedia applications. The objective of MPEG-4 is to support content-based communication, access and manipulation of digital objects which can be natural or synthetic [15]. With a flexible toolbox approach, MPEG-4 is capable of supporting diverse new functionalities and satisfy various application requirements on different aspects and hence will cover a broad range of present and future multimedia applications. In addition, due to its extensible system configuration architecture, MPEG-4 is aimed to be more compatible with advanced new technologies.

Because of its object-based features and flexible toolbox approach, MPEG-4 is considerably more complex and demands more computing power than previous video coding standards. Thus, a software-based implementation using parallel processing seems to be only viable approach for building MPEG-4 based systems.

Previous coding standards such has H.261, MPEG1/2, and H.263 have been implemented using either software (see [3], [4], [5] and [18]) or hardware-based (see [23], [2], [17] and [10]) approaches with each having its pros and cons. Interactive multimedia systems, such as digital television, that require real-time multimedia communication, both the encoder and decoder are desired to be highly efficient and must provide close to real-

†. This work was supported by the Hong kong Telecom Institute of Information Technology. The authors would also like to thank Dr. Robert Yung of Sun Microsystems and Dr. Ya-Qin Zhang of Sarnoff Corporation for technical support.

time operations. For example, mobile communication and database access require very low bitrate video coding and error resilience across various networks; virtual reality requires integration of natural and synthetic hybrid object coding; interactive video games require a high degree of object based interactivity. Instead of traditional frame based interaction such as fast-forward, fast-backward, etc., new ways of interactively are needed to efficiently realize such applications.

MPEG-4, due to its content-based representation nature and flexible configuration structure, is considerably more complex than previous standards. Any MPEG-4 hardware implementation is likely to be very much application specific. Therefore, software-based implementation is a natural and viable option. The main problem with such an approach is the requirement of a huge amount of computing power to support real-time encoding and decoding operations. As elaborated in the subsequent section, although MPEG-4 encoding is highly suitable for implementing using parallel and distributed systems, it is nevertheless a non-trivial task because of the unpredictable nature of MPEG-4 workload.

We are building an MPEG-4 based interactive multimedia environment for supporting applications in the areas of CAD, teaching, and animation. And as a part of this system, we have implemented an MPEG-4 encoder with a software-based approach using parallel processing. As illustrated in Figure 1, the system is conceptually based on a client-server model, with a number of clients making interactive requests to a server. The server encodes and delivers the requested information using MPEG-4 format. The problem addressed here is the implementation of MPEG-4 encoder which is done using a distributed cluster of workstations.

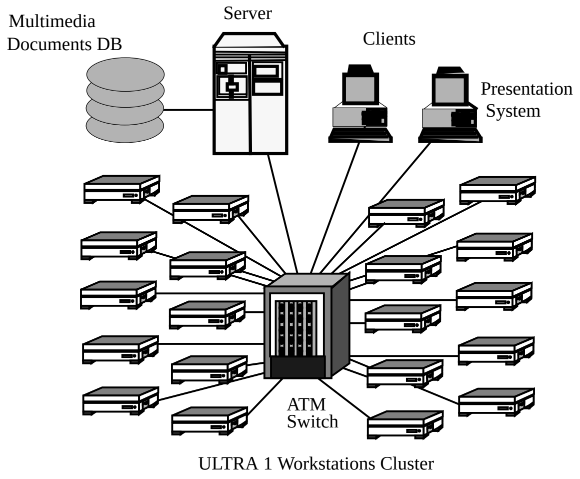

Figure 1: The MPEG-4 based interactive multimedia system.

The rest of this paper is arranged in the following manner: Section 2 gives a brief overview of MPEG-4 video verification model. Section 3 describes the proposed implementation approach in detail. A real-time scheduling algorithm is proposed to schedule various sub-tasks of the encoder. A dynamic shape-adaptive data partitioning scheme is also proposed to further divide the data of a sub-task. Section 4 provides the experimental results. The last section concludes the paper by providing an overview of our ongoing research in this area and future avenues of extending this work.

2 Overview of MPEG-4 Video

MPEG-4 is scheduled to become an international standard in November 1998. During the development, the so called "Verification Model" (VM) methodology is adopted to specify the candidate technologies which may be included in the final standard [14]. The VM is supposed to evolve through a core experimental process [19]. MPEG-4 video VM is one of the main parts of MPEG-4 with the objective to support three major functionalities: content-based interactivity, coding efficiency, and universal access [21]. Its bitrate can range from 10kbits/s up to several Mbits/s. The spatial resolutions include SQSIF/SQCIF, QSIF/QCIF, SIF/CIF, 4*SIF/CIF, and CCIR601. In contrast to the existing 'frame-based' or 'pixel-based' standards, such as MPEG-1/MPEG-2 and H.261/H.263, MPEG-4 video is object-based hybrid coding standard which specifies the technologies for representing and processing video object efficiently to support various content-based functionalities within the compression domain.

Figure 2 shows the conceptual architecture of MPEG-4 based multimedia systems. A user using a decoder can access arbitrarily shaped objects in the scene or send a request to the encoder which can manipulate the objects and deliver the requested objects. The encoder compresses the data and includes the additional necessary information such as the scene description and synchronization requirements.

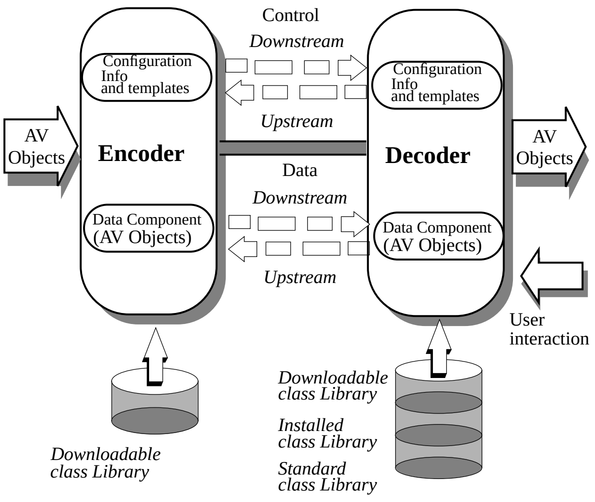
Control
Downstream
Upstream
Configuration Info and templates
Configuration Info and templates
AV Objects
Encoder
Decoder
AV Objects
Data
Downstream
Upstream
Data Component (AV Objects)
Data Component (AV Objects)
User interaction
Downloadable class Library
Installed class Library
Standard class Library
Downloadable class Library

Figure 2: Overall MPEG-4 system architecture.

Figure 3 is the overall structure of the MPEG-4 video codec (encoder and decoder) which is based on the concept of video object planes (VOPs) defined as the instances of video objects at a given time. As illustrated in Figure 4, a video may have many sessions and each session may involve many objects or layers of objects which in turn may have multiple instances in time. A VOP lasts over a number of video frames.

The video encoder is composed of a number of identical VOP encoders. Each object is segmented from the input video signal and goes through the same encoding scheme separately. The bitstreams of different VOPs are then multiplexed and transmitted. At the decoder, the received bitstream are demultiplexed and decoded by each VOP decoder. The

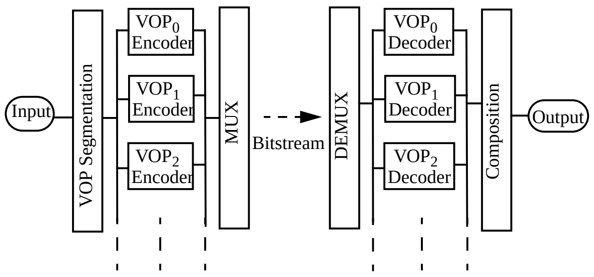

Figure 3: MPEG-4 video codec (encoder and decoder) structure.

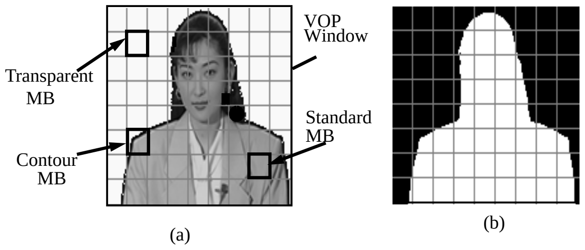

Figure 5: Representation of the VOP (person Akiyo). (a) Image of original 'Akiyo' VOP; (b) Binary alpha plane of the VOP

reconstructed video objects are then composited by the composition information (which is sent along with the bitstream) and presented to the user. The user interaction with the objects such as scaling, dragging, replacement and linking can be handled either in the encoder or in the decoder.

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 5, 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.

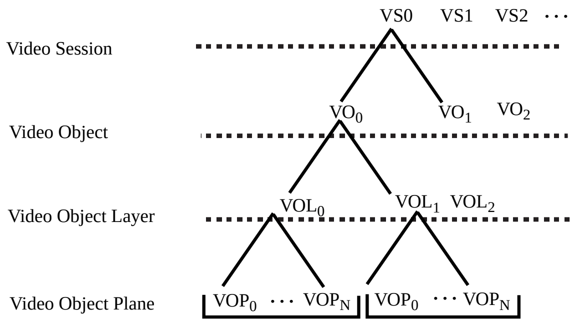

Figure 4: The concept of video object planes.

Each VOP encoder consists of three main functions: shape coding, motion estimation/compensation, and texture coding (see Figure 6). When the shape of a VOP is the standard rectangular size, MPEG-4 encoder structure is similar to that of MPEG1/2 encoder, shape coding can be skipped.

Shape coding is used to compress the alpha plane information which indicates the object region and contour within the scene. There are two types of alpha planes: binary alpha plane and grey scale plane, with both having the same format as the luminance file. The binary alpha plane is encoded by the algorithm called *content-based arithmetic encoding* (CAE). And the grey scale alpha plane is encoded by a block based DCT (Discrete Cosine Transform) with motion compensation which is similar to texture coding.

Motion estimation and compensation (ME/MC) are used to reduce temporal redundancies. Motion prediction is performed on the current block to find the best matched block within the search window in the previous frame; the block size can be either 16×16 or 8×8. The motion vector which is the displacement between the current and the best matched block from the previous frame are then coded with respect to the neighboring three motion vectors already transmitted. In addition to the basic motion technique, unrestricted ME/MC, advanced prediction mode and bidirectional ME/MC (especially for B-frame) are supported by the MPEG-4 video VM to obtain a significant quality improvement. Unrestricted motion estimation extends the predicted VOP to a large enough size and performs ME/MC over the VOP boundaries. Thus, the motion vectors may be outside the predicted VOP area. Such a mode can achieve improved video quality when the object is moved by the camera or the object is located on the picture edge. In the advanced mode, each macroblock may have one, two, or four vectors. The advanced mode also uses *overlapped block motion compensation* for luminance, which can provide a significant quality improvement with a little increase in complexity.

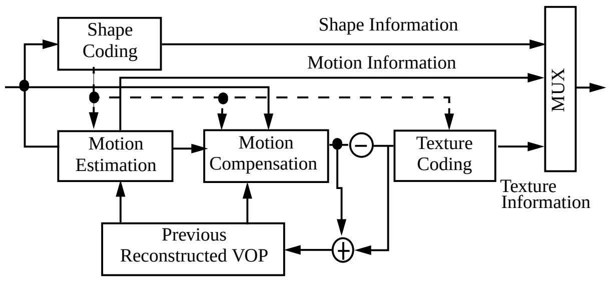

Figure 6: VOP encoder structure.

Since the shape of a VOP may be arbitrary and could vary over time, a padding technique is applied on the blocks on the previously reconstructed VOP borders to fill the values of the pixels outside the object. This allows polygon matching instead of block matching for rectangular image. 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.

The texture coding which deals with the intra and residual data after motion compensation of VOPs includes algorithms that are similar or identical to the ones used in H.263. A 2D DCT is performed on each macroblock. The DC and AC coefficients are then quantized by either MPEG or H.263 quantization method. For I-VOP and P-VOP, the intra DC and

AC coefficients can be predicted from the corresponding coefficients in the previous neighboring blocks to get the differential DC/AC values. After using a scanning method (such as zigzag scan, alternate-horizontal scan and alternate-vertical scan), the quantized transform coefficients are further coded by variable length coding (VLC). The data of the blocks on the boundary of the object can be coded by low pass extrapolation (LPE) padding and shape adaptive DCT (SA-DCT).

MPEG-4 also supports scalable coding of video objects in both spatial and temporal domains, and provides error resilience across various media. In addition to the above basic technologies used in the encoder structure, the toolbox approach of MPEG-4 video makes it possible to achieve more improvement for some special cases by dedicated tools. Further details on the coding and syntax of MPEG-4 video can be found in [16].

3 Parallelizing the MPEG-4 Encoder

Since MPEG-4 supports many new functionalities that are not available in the existing standards, it will cover a broad range of multimedia applications, such as interactive video games, intra/internet multimedia mailing, and content-based database access. Most of these applications have real-time requirements which demand the codec to be highly efficient. In order to deal with arbitrarily shaped objects, more sophisticated techniques are needed to achieve an efficient compression. But this can introduce extra complexity in the encoder which in turn requires additional computational power. Since the encoder of MPEG-4 video is much more complex and time consuming in computing than the decoder, it is more challenging to speedup the computation in the encoder.

As mentioned earlier, no hardware-based MPEG-4 encoder can fully support the flexible and extensible features of MPEG-4 standard. The object-oriented nature of MPEG-4 requires a highly flexible and somewhat programmable encoder which is more feasible using a software-based approach. But the computational requirement of a software-based encoder is simply too enormous to be handled by a single processor PC or even a very fast workstation. It is, therefore, natural to exploit the high computational power offered by a high-performance parallel or distributed system. In our MPEG-4 based multimedia project, we have implemented an encoder on a cluster of dedicated workstations that collectively work as a virtual parallel machine. The architecture of MPEG-4 encoder as shown Figure 3 also happens to be very suitable for distributed computing. Each input VOP is encoded separately and efficient performance can be achieved by decomposing the whole encoder into separate tasks with individual VOP encoders and running them simultaneously. However, the task of parallelizing the MPEG-4 encoder VM on a cluster of workstations is a non-trivial as it requires a careful data distribution and scheduling of various parts of the encoder to ensure that spatio-temporal relationships between various VOPs are preserved.

In a simpler approach, one could use a single workstation to encode one VOP. But this scheme does not fully exploit the computational power of the system because it is not scalable and the degree of parallelism offered by this approach is rather limited. A more effective approach is to form groups of workstations, with each group working on a single VOP while parallelism is exploited by further partitioning the VOP among the workstations within the group. This scheme, however, requires a careful partitioning of both control and data. Furthermore, the sizes of VOPs change with time implying that distribution and partitioning of VOPs will need to be adjusted accordingly. Since this must be done in real-time, the cost of scheduling and distribution must be kept low to ensure that the benefits gained from an efficient parallelization are not outweighed by a long time taken by the scheduler.

In our scheme, the control parallelism is achieved by making groups of workstations, and assigning the task of one VOP encoding to one group. However, the distribution of VOPs to different groups of workstations must consider the relationships between the VOPs. This is done by using a scheduling algorithm that distributes VOPs to various groups of workstations in accordance with their priorities so that their encoding is complete before their presentation deadlines.

Data parallelism is exploited by dividing the data of a VOP among the workstations within a group, allowing further gain in computing speed. For distributing the data of a VOP various partitioning schemes are possible. The details of the scheduling algorithm and data partitioning schemes are described below.

3.1 Real-time Scheduling

In MPEG-4 video VM encoder, one of the most important issues to consider is the synchronization of various video objects. Each object may have certain presentation timing constrains which, in turn, may be dependent on the other objects. The playout time requirement and associated synchronization constrains among multiple video objects must be satisfied in real-time to guarantee a smooth flow of video sequence presented to the user.

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 [20]. A real-time scheduling can be characterized as being either static and dynamic. In static scheduling, the algorithm determines the schedule with the complete knowledge of all the tasks in advance. In contrast, a dynamic scheduling algorithm deals with task assignment at run-time because the information about the tasks is not available in advance. Static scheduling incurs little run-time cost but cannot adapt to the indeterministic behavior of the system. On the other hand, dynamic scheduling is more flexible as it can be adjusted to system changes but incurs a high run-time cost.

According to MPEG-4 video hierarchical syntax structure shown in Figure 4, we can employ a completely static scheduling at the VS level which requires the knowledge of all the objects within the session beforehand. Alternatively, we can perform dynamic scheduling on a frame by frame basis at the VOP level so as to adapt to the VOPs variations. Since in an MPEG-4 video session, the number of objects may change from time to time, their characteristics such as frame rate, playout deadlines, and spatial resolutions may also be different. Thus, while a static scheduling scheme at a VS level is feasible for some non-real-time applications, its is not suitable for most real-time applications because of the unpredictable characteristics of VOPs.

In our implementation, we have designed a hybrid static and dynamic scheduling scheme applied at the VO level. The knowledge of video objects can only be known after observing a time period. During that period, either the objects variation is arisen by the operations such as user interaction or content database retrieval, or the characteristics of these objects are relatively stable. The length of the period depends on the availability of objects. When a new object is added or dropped, we have to reschedule the tasks for the next period. The main advantage of such scheduling is its ability to adapt to the variation of both deterministic and indeterministic video objects on line with a little overhead. Figure 7 shows the playout time chart of a general MPEG-4 video example. The session has 4 video objects (VOs); VO_0, VO_1, VO_3 start at time 0, and VO_2 starts at time unit 4. VO_0 and VO_1 end at time unit 4, while VO_2 and VO_3 end at time unit 12. 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_1 and 4 for VO_2 and VO_3.

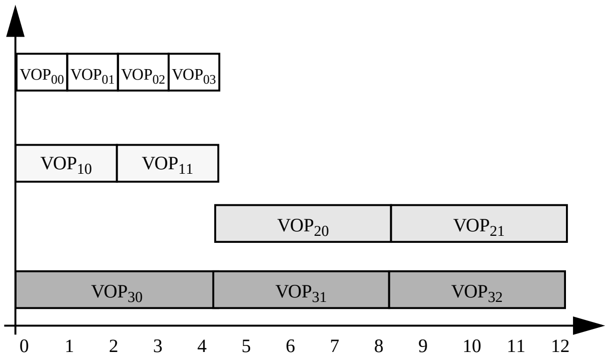

Figure 7: Playout time chart for video session.

Figure 8 indicates the scheduling period at VO level for the case of Figure 7. The scheduling period is bounded by the successive object scheduling instants (OSIs) and the complexity of the scheduling depends on the number of OSIs during the whole video session.

A number of scheduling algorithms have been developed for both distributed and parallel systems [8]. In our implementation, we use a variant of the *earliest-deadline-first* (EDF) algorithm which has been widely employed in many applications [22]. The principle of this algorithm is that the tasks with earlier deadlines are assigned higher priorities and run before tasks with lower priorities. In our implementation, VOPs with the earlier playout deadlines or synchronization points get to be encoded and delivered first. For the tasks with the same deadline, we assign a portion of available processors to each object, with the number of allocated processors depending upon the size ratio among these objects because a video object with a larger size generally requires more computing and vice versa.

3.2 Dynamic Shape-Adaptive Data Partition

Parallel programming paradigms can be classified into various models such as object-oriented model, control-parallel model, and data-parallel model. Data parallel paradigm emphasizes exploiting parallelism in a large data sets such as a video session which usually consists of a large amount of data. The main idea of data partitioning in video encoding 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 memories and run the program on

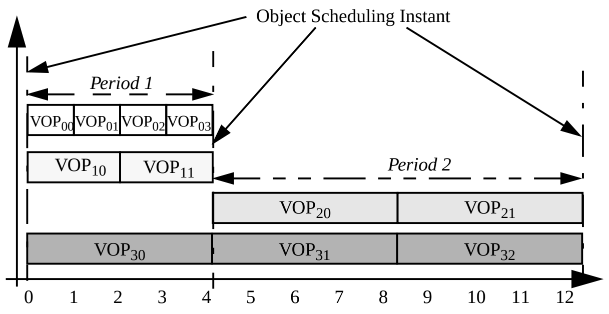

Figure 8: Scheduling time chart for video session.

the data blocks simultaneously, a high speedup can be achieved.

The exchange of the data and synchronization in a distributed system can only be done through message passing among processors. However, the communication overhead can penalize the gain achieved due to parallelism. Most tools specified by the MPEG-4 video standard, such as padding, DCT, quantization, and VLC, are block-based algorithms and perform the computing restricted within a macroblock. Therefore, we can employ macroblock-based data partition to map the integer number of macroblocks to each processor and enable the compression algorithm to be done locally. This is done by setting the workstations in a virtual two dimensional topology and then mapping the data onto the topology. As for motion estimation which finds the motion vector of current macroblock from the previous frame search window, both the current block data and search window data are involved in the computation.

To reduce the interprocessor communication overhead, we use an overlapped partition approach which minimizes the data exchange during the motion estimation procedure, but requires more memory in each processor to store the entire search window data from the previously processed frame (as shown in Figure 9). This approach allows to perform motion estimation on all the processors independently since the required data are available in the local memories.

The second problem to be addressed is the issue of load balancing. Due to the object-based nature of MPEG-4 video, the size and location of each object may vary with time, and such situations cannot 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.

Figure 10 shows several commonly used partitioning methods. Strip-wise 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. Block-wise partition divides the VOP window

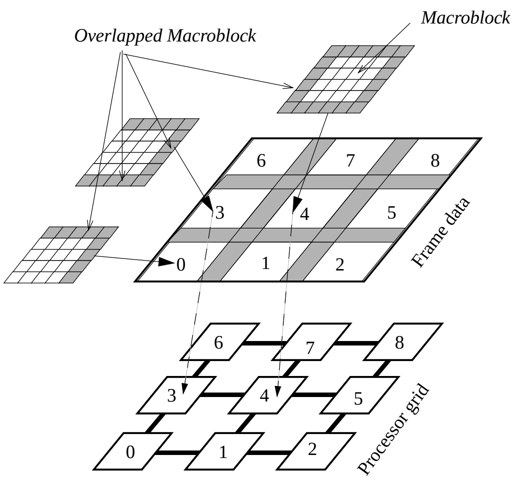

Figure 9: Overlapped data partitioning example.

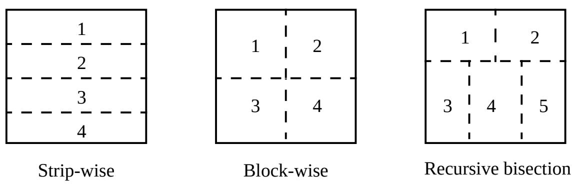

Figure 10: Common partitioning methods.

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 recursively in binary fashion [7]. It is capable of optimally equally distributing the computational load, while it is relatively expensive to execute the recursive operations during the decomposition.

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 behavior of the system, but it depends on the trade-off between the balancing quality and overhead run-time cost [1]. 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 run-time overhead and fine granularity.

First, the entire MPEG-4 video session is defined as a the 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 run-time. 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. 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 equally distribute the rectangular subregion in a straightforward way. In addition, the object size may become too tricky to do the strip-wise or block-wise partition.

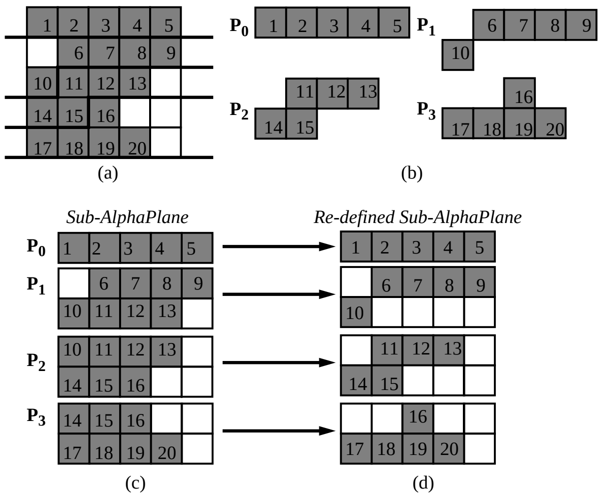

Figure 11: Arbitrary partitioning example.

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 depicted in Figure 11, the gray blocks represent the contour and standard MBs while the white blocks 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 illustrated in Figure 11 (a), there are 20 contour and standard MBs within the window. Each processor is assigned 5 contour and standard MBs. Since each processor (P_0 to P_3) may get arbitrarily shaped subregions (see Figure 11 (b)), it may require complex data structures for processing. To overcome this problem, we extend these subregions to rectangular regions called sub-AlphaPlane (see Figure 11 (c)). 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 (see Figure 11 (d)). For example, Processor 3 (P_3) encodes only the subregion that includes the contour and standard macroblock from 16 to 20 as shown in Figure 11 (b). In order to get a rectangular subregion which contains those blocks, we extend this subregion such that 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 11 (d)). Therefore, processor 3 still processes 5 contour and standard MBs while keeps the subregion rectangular.

Because such a partition is based on macroblock decomposition, the granularity is small allowing a finer load balancing of the workload among the multiprocessors. In addition, by keeping the data block for each processor rectangular, the decoder can recover the entire object easily. Because the syntax definition of each bitstream contains the position and size information of the rectangular block, reconstruction of the object is just equal to the composition of the data blocks together and no bitstream combination required.

4 Experimental Results

The proposed parallel approach has been tested on a cluster of 20 UltraSparc-I workstations connected by a ForeSystems ATM switch (ASX-1000). The cluster is virtually configured as virtual 2D processor grid which is independent of the hardware topology.

For inter-processor communication and synchronization, we use *Message Passing Interface* (MPI) [24], ensuring the portability of our MPEG-4 video encoder across various machines. MPI is an industrial standard designed by MPI Forum for supporting a portable message-passing parallel program on massively parallel computers as well as networks of workstations. MPI includes the syntax and semantics of point-to-point and collective communication routines which are useful to most parallel programmers.

Several experiments have been performed on a sets of MPEG-4 video test sequences by using a number of workstations ranging from 1 to 20. A fast block-based motion estimation algorithm [12] is adopted to speedup the computation of motion estimation while maintaining the visual quality close to that of the full search.

Our experiments included video sequences of QCIF resolutions which are chosen from different classes of MPEG-4 library and represent various characteristic in terms of spatial detail and movement.

Figure 12 shows the encoding rates for different MPEG-4 video test sequences by using various numbers of workstations, and Figure 13 is the overall speedup ratio. Figure 14 shows the comparison between the static strip-wise/block-wise partition, object-based partition [11] and our proposed method for the test sequence '*Children21*' with QCIF format. We can observe that a high real-time performance has been achieved by our method.

Figure 15 is one of the alpha planes of VOP 'Children21', we can observe that most transparent macroblocks located between two boys. Neither the block-wise or strip-wise partition can make the workload balancing when the number of processors increased. It is unavoidable that some processor might deal with the data block full of the transparent

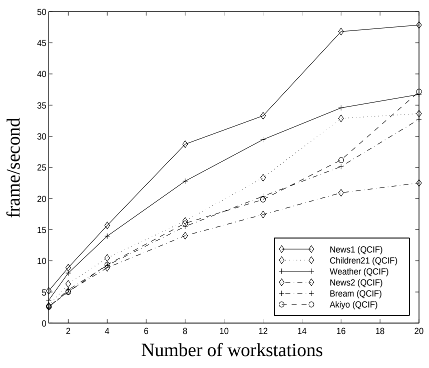

Figure 12: Encoding frame rate.

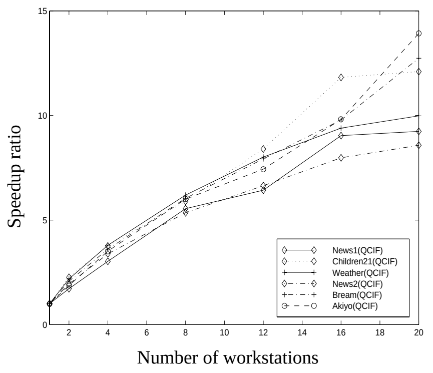

Figure 13: Overall speedup.

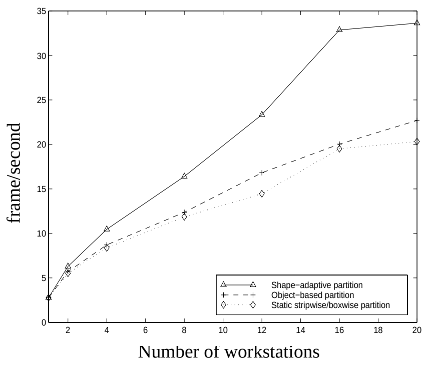

Figure 14: Partitioning performance comparison.

macroblocks that require little computation. Some processors need to process the entire data block that is full of the computationally intensive contour/standard macroblocks. The proposed object-based partitioning method [11] can determine the optimal rectangular grid from the available block-wise/strip-wise partitioning grid, and its performance is better than fixed grid partitioning, especially when the VOP shape changes significantly. One limitation of this method is that the number of possible grids is limited. When the number of processors increases, the same problem with fixed block-wise/strip-wise partition will occur. The shape-adaptive partition can guarantee the workload balancing in such a case since it can equally distribute the computationally intensive macroblocks to each processor without using an exhaustive search. Therefore, a higher encoding rate can be achieved compared to the other two methods.

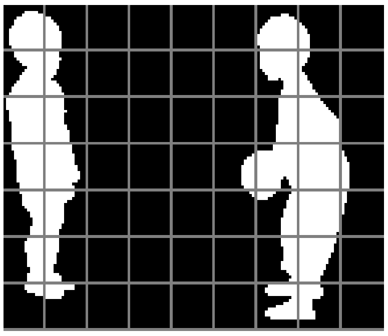
Figure 15: Alpha plane of VOP ‘Children21’ (QCIF).

5 Conclusions

In this paper a software-based parallel implementation of MPEG-4 video VM encoder using a cluster of workstations has been proposed. The experimental results on various test sequences have been provided and an encoding rate higher that real-time has been achieved on most sequences. The contribution of our work includes the use of a shape-adaptive data parallel scheme, and a real-time scheduling algorithm to implement MPEG-4 video encoder. In our present work, we are exploiting dynamic load balancing algorithms for heterogeneous computing environment for MPEG-4 applications.

References

[1] I. Ahmad, “Resource Management of Parallel and Distributed Systems with Static Scheduling: Challenges, Solutions and New Problems,” *Concurrency: Practice and Experience*, vol. 7, no. 5, Aug. 1995, pp. 339-348.

[2] T. Akiyama, *et al.*, “MPEG-2 Video Codec using Image Compression DSP,” *IEEE Transactions on Consumer Electronics*, vol. 40, no. 3, pp. 466-472, Aug. 1994.

[3] S. M. Akramullah, I. Ahmad, M. L. Liou, “A Portable and Scalable MPEG-2 Video Encoder on Parallel and Distributed Computing Systems,” *Proc. of the SPIE*, vol. 2727, PT. 2, pp. 973-984, Mar. 1996.

[4] 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.

[5] 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.

[6] T.E. Anderson, D.E. Culler, and D. Patterson, “A Case for NOW (Networks of Workstations),” *IEEE Micro*, vol.15, no.1, pp. 54-64, Feb. 1995.

[7] M. J. Berger and S. 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] S. Cheng *et al.*, “Scheduling Algorithms for Hard-Real Time Systems - a Brief Survey,” *Hard Real-time Systems*, IEEE Computer Society press, 1988.

[9] L. Chiariglione, “MPEG and Multimedia Communications,” *IEEE Transactions on CSVT*, vol. 7, no. 1, pp. 5-18, Feb. 1997.

[10] H.A. Chow, H. Alnuweiri, “An FPGA-Based Transformable Coprocessor for MPEG Video Processing,” *Proceedings of the SPIE - The International Society for Optical Engineering*. vol. 2914, pp. 308-320, 1996.

[11] Y. He, I. Ahmad and M. 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.

[12] 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.

[13] ITU-T Recommendation H.263 Draft, “Video Coding for Narrow Telecommunication Channel at <64 kbit/s,” Apr. 1995.

[14] ISO/IEC, “Verification Model Development and Core Experiments,” ISO/IEC JTC1/SC29/WG11 N1110, Nov. 1995.

[15] ISO/IEC, “MPEG-4 Proposal Package Description (PPD),” ISO/IEC JTC1/SC29/WG11 N0988, July 1995.

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

[17] D. Kim, *et al.*, “A Real-Time MPEG Encoder using a Programmable Processor,” *IEEE Transactions on Consumer Electronics* vol. 40, no. 2, pp. 161-170, May 1994.

[18] J. Nang and J. Kim, “An Effective Parallelizing Scheme of MPEG-1 Video Encoding on Ethernet-Connected Workstations,” *Proceedings. Advances in Parallel and Distributed Computing (Cat.No.97TB100099)* pp. 4-11, March 1997.

[19] F. Pereira, and T. Alpert, “MPEG-4 Video Subjective Test Procedures and Results”, *IEEE Transactions on CSVT*, vol. 7, no. 1, pp. 32-51, Feb. 1997.

[20] M. Pinedo, “Scheduling: Theory, Algorithms and System,” Prentice Hall, 1995.

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

[22] J.M. Sohn, and G.Y. Kim, “Earliest-Deadline-First Scheduling on Nonpreemptive Real-Time Threads for a Continuous-Media Server,” *Proceedings of High-Performance Computing and Networking. International Conference and Exhibition*, pp. 950-956, 1997.

[23] H.H. Taylor, D. Chin and A.W. Jessup, “An MPEG Encoder Implementation on the Princeton Engine Video Supercomputer,” *Data Compression Conference 1993*, pp. 420-429 1993.

[24] D. W. Walker, and J.J. Dongarra, “MPI: a Standard Message Passing Interface,” *Supercomputer* vol. 12, no. 1, pp. 56-68, Jan. 1996.