Modeling and Scheduling for MPEG-4 Based Video Encoder Using a Cluster of Workstations

Yong He[1], Ishfaq Ahmad[2], and Ming L. Liou[1]

[1] Department of Electrical and Electronic Engineering
{eehey, eeliou}@ee.ust.hk
[2] Department of Computer Science
iahmad@cs.ust.hk
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract. In this paper, we first present an Object Composition Petri Nets (OCPN) based model methodology for describing the dynamic behaviour of the multiple video objects and user interactions during the entire MPEG-4 video session; then, a Group of Video Object Plane (GOV) based periodical scheduling algorithm is proposed to assign the encoder tasks to a cluster of workstations with load balancing guarantee. The scheduling scheme can allocate the tasks efficiently according to the timing constraints as well as user interactions. The performance of the encoder can scale according to the number of workstations used. The experiment results indicate that a real-time encoding rate can be achieved for the sequences with multiple video objects.

1 Introduction

Most current multimedia applications can be viewed as the merging of traditional computer, communications, and broadcasting industries. In order to support the functionalities, such as content-based interactivity, high compression and random access, a new international standard, MPEG-4, is currently being developed by MPEG (Moving Picture Experts Group) and expected to be finalized towards the end of 1998 [1]. With a flexible toolbox approach, MPEG-4 is capable of supporting diverse new functionalities and satisfy various application requirements and hence will cover a broad range of multimedia applications [5].

MPEG-4, due to its content-based representation nature and flexible configuration structure, is considerably more complex than previous standards such as MPEG-1, MPEG-2 and H.263. In addition, MPEG-4 enables user to manipulate and process the objects with various dedicated tools. 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. With the developments in parallel and distributed systems, a higher degree of performance at an affordable cost (such as a network of workstations or PCs) can be achieved, provided the parallelism from the application at hand is effectively extracted. While although MPEG-4 encoding is highly suitable for implementation using parallel and distributed systems, it is nevertheless a non-trivial task because of the unpredictable nature of MPEG-4 workload.

In this paper, we use a Petri net based modeling technology to describe the temporal relationships between various video objects and user interactions. An efficient scheduling algorithm is proposed to improve the performance of the video encoder with load balancing guarantee. With the proposed approaches, our encoder can allocate the tasks efficiently according to the timing constraints as well as user interactions. The performance of the encoder can scale according to the number of workstations used, With 20 workstations, the encoder can achieve a real-time encoding rate on some multiple objects sequences.

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, including the Object Composition Petri Net (OCPN) based modeling and a dynamic scheduling algorithm. Section 4 provides the experimental results and the last section presents the conclusion.

2 Overview of MPEG-4 Video Verification Model

MPEG-4 aims at providing the standard technological elements enabling the integration of the production, distribution and content access paradigms of the multimedia environment [4]. MPEG-4 video is an object-based hybrid natural and synthetic coding standard which specifies the technologies enabling the functionalities such as content-based interactivity, efficient compression and error resilience [8]. Fig.1 illustrates MPEG-4 video coding and composition procedure with user interactions. Both encoder and decoder are based on the concept of video object planes (VOPs).

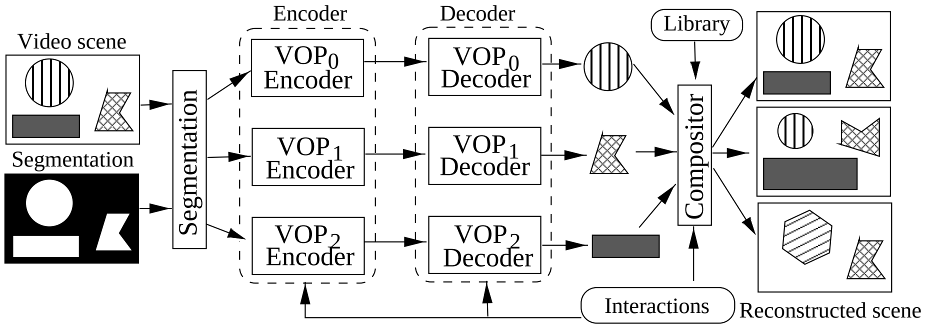

Fig. 1. MPEG-4 video scene coding and composition.

The video encoder is composed of a number of 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 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 and linking can be handled either in the encoder or in the decoder.

Each VOP encoder consists of three main parts: shape coder, motion estimation/compensation, and texture coder. Shape coder is used to compress the alpha plane information which indicates the object region and contour within the scene. Motion estimation and compensation (ME/MC) are used to reduce temporal redundancies, and the techniques such as unrestricted ME/MC, advanced prediction mode and bidirectional ME/MC are supported to obtain a significant quality improvement. The texture coder 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.

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 [6].

3 MPEG-4 Video Encoder Parallelism

In our MPEG-4 based multimedia project, we have implemented an video encoder on a cluster of dedicated workstations that collectively work as a virtual parallel machine. The architecture of MPEG-4 video encoder as shown Fig.1 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 video encoder on a cluster of workstations is a non-trivial task as it requires a careful scheduling of various tasks of the encoder to ensure that spatio-temporal relationships between various VOPs are preserved. Fig.2 is a playout example of a

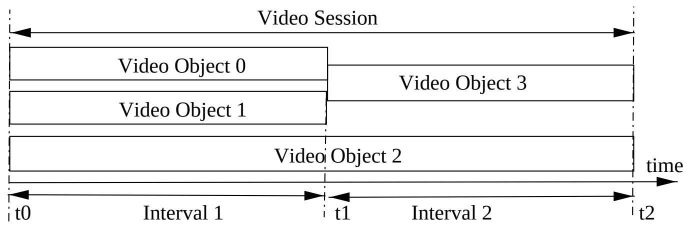

Fig. 2. Playout scenario of a MPEG-4 video session in time chart.

MPEG-4 video session in time chart. The presentation scenario begins with concurrent playout of VO_0 and VO_1, then followed by a new video object VO_3 at the time t_1. While VO_2 can be treated as the background video object throughout the whole session. The distinct characteristic of each video object, such as the shape size, playout duration and spatial allocation, can be different from each other, and the temporal relationship among the objects may also be designed synthetically. For example, we can specify that VO_0 and VO_1 are synchronized during the presentation from t_0 to t_1, and VO_2 and VO_3 are synchronized from t_1 to t_2; the playout frame rate of VO_2 and VO_3 is 5 frames/sec. and the frame rate of VO_0 and VO_1 is 10 frames/sec. According to the states of the video objects, the entire session can be partitioned into a number of presentation intervals as Fig.2 shows.

For MPEG-4 based video presentation, the system must observe and obey the temporal relations among various video objects at the time of playout. The coordination of real-time presentation of video sequences and maintenance of the timing-order among video objects are known as temporal synchronization. There are two styles of synchronization, which can be identified as intra-object synchronization and inter-object synchronization. Intra-object synchronization refers to the maintenance of real-time constraints across a continuous video object sequence and is required to meet the respective playout deadlines within the sequence. Inter-object synchronization refers to the maintenance of real-time constraints across more than one video object sequences and has to be satisfied since the playout of video objects may be dependent to each other in the scene.

The maintenance of temporal synchronization is the most crucial requirement in a distributed multimedia system. To enable a perfect playout manner at the client site, the entire system must be able to operate in a synchronized fashion. Since the encoder is the most computational intensive component of the entire system, we address the problem of designing efficient scheduling algorithms for encoding multiple video objects on the fly that guarantee real-time continuous presentation at the client site. Here, we assume that the network can provide a certain level of Quality of Service (QoS) throughput for the issues of networking communication is beyond this paper.

In real-time applications, the video encoder should be fast and efficient enough to meet the explicit playout timing requirement. It is impossible to achieve such real-time performance with a single PC or workstation, and parallel processing seems to be a viable solution for such problem as mentioned above. Due to the limited number of available processors, we have to determine the processing schedule of the video object tasks and manage the system resources carefully according to the playout requirements of the tasks, so as to guarantee a smooth playout at the presentation clients.

3.1 Modeling MPEG-4 Video with Object Composition Petri Nets

To elaborate a feasible scheduling scheme, a model specifying the timing constraints among different video objects is necessary. Various modeling techniques have been proposed and the Petri net based model has been shown to be a useful technique for specifying object level synchronization requirements [10]. Here, we employ such a model known as the object composition Petri net (OCPN) to represent the occurrence of multiple video objects due to its ability to explicitly capture all necessary temporal relations. In general, an OCPN refers to a Petri net graph, each place represents the playout of a video object while each transition represents a synchronization point. However, in order to model the flow of MPEG-4 based video objects, more definitions are essential to allow the dynamic behaviours such as user interactions and object attributes variation. Therefore, We introduce an augmented OCPN model which is capable of supporting interactive operations while satisfying the complex synchronization requirement. With this model, one can precisely describe temporal information of the video object, user interaction, as well as scheduling procedure involved in the entire real-time video session.

The OCPN is 9-tuple $OCPN = \{P, T, A, M, D, TS, PR, PC, PS\}$ where:
$P = \{p_0, p_1, \ldots, p_m\}$ is a finite set of places with $m \geq 0$;
$T = \{t_0, t_1, \ldots, t_n\}$ is a finite set of transitions with $n \geq 0$ and $P \cap T = \Phi$;
$A = \{P \times T\} \cup \{T \times P\}$ is a mapping arcs between places and transitions;
$M : P \rightarrow I$ represents the tokens distributed in the place where I is the integer;
$D : P \rightarrow R^+$, represents the augmented duration D of the place where R^+ is a non-negative real number;
$TS : P \rightarrow \{IntraS, InterS\}$ defines different transition types. $IntraS$ is the intra-object synchronization and $InterS$ is the inter-object synchronization;
$PR : P \rightarrow R$ represents the request of the clients;
$PC : P \rightarrow C$ defines the model construction operation;
$PS : P \rightarrow S$ defines the scheduling operation in the encoder.

The above P, T and A definitions are the same as that of the Petri net. Number of tokens to be deposited at a given place p is indicated by M. The firing condition of both $IntraS$ and $InterS$ are determined by the token states at the input place.

Generally, a MPEG-4 based video session is much complicated and enforces different characteristic at various syntax levels, such as video session (VS), video object (VO) and video object plane (VOP) as specified by the standard. It is important that such hierarchical levels of synchronization can be depicted by the model to enable the handling of large and complex MPEG-4 scenarios. For OCPN, such hierarchical modeling capabilities can be achieved through subnet replacement: at the highest level abstraction, OCPN can be represented by a set of abstract place, and each abstract place can be decomposed into an OCPN subnet, the abstract places in the sub-OCPN indicate finer-grained data units and timing constraints; similarly, the sub-OCPN places can in turn be an abstraction of the lower OCPN subnet, and such replacement process can be recursively applied until to the lowest level where each place can not be decomposed any more. In our approach, we define three levels OCPN, namely VS-OCPN, VO-OCPN and VOP-OCPN. The abstract place of OCPN at different level is defined as:
$PVS : P \rightarrow VS$ represents the entire video session;
$PVO : P \rightarrow \{VO_0, VO_1, \ldots, VO_{N-1}\}$ represents a set of video objects;
$PVOP : P \rightarrow \{VOP_0, VOP_1, \ldots, VOP_{M-1}\}$ represents video object planes;

VS-OCPN is the highest level with an abstract place PVS which indicates the entire video session. VO-OCPN is the middle level with a set of places PVO representing the video objects, and the transition known as $InterS$ describes the temporal precedence and synchronization between the video objects. The lowest level VOP-OCPN is composed of places $PVOP$ which represent frames of the video object. Both intra-object synchronization $IntraS$ and inter-object synchronization $InterS$ can be depicted by the VOP-OCPN. Fig.3 shows a decomposition process of an OCPN, and Fig.4 is the global VOP-OCPN representation equivalent to the example of Fig.2. The time instance t_1, t_2 and t_3 are set to 0 second, 3 second and 6 second respectively.

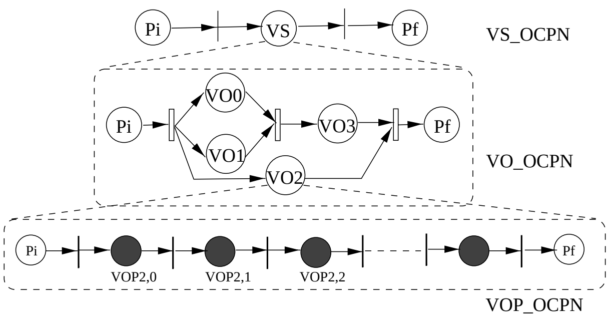

Fig. 3. Decomposition of an OCPN.

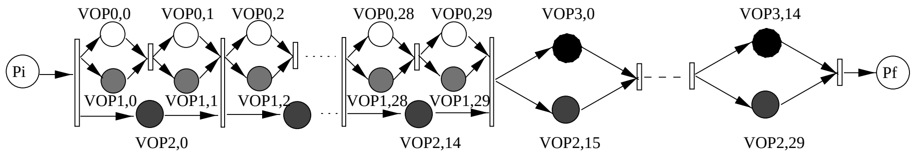

Fig. 4. Global VOP-OCPN example.

For most pre-orchestrated video sequences, as the timing constraints are known a prior, it is possible to determine the structure of a OCPN beforehand. For live video data, however, since related knowledge for constructing the OCPN cannot be verified a prior, a model has to be generated along with the video session. In addition, either pre-orchestrated or live video data can be ceased and new video data can be introduced at any time by the user interactions, and the temporal relationships between the video objects may also be manipulated by the user. Due to such unpredictable behaviours of the user interaction, the OCPN model should be able to support both deterministic as well as imprecise events on the fly.

Fig.5 illustrates the generation strategy of the OCPN (using Fig.3 as an example) which allows user participate actively. Corresponding to the user request as we assumed, the OCPN model can be generated at run-time. At the beginning of time t_0, we can obtain the attributes such as playout deadlines and data dependency of VO_0, VO_1 and VO_2 initially, and construct the OCPN as place PC_0 indicated. Such model may remains the same if all VOs status are stable. Being caused by user's interaction PR_0 at time t_1, VO_0 and VO_1 are halted and a new VO_3 is succeeded and synchronized with VO_2. Thus, the model should be changed by PC_1. The same approach PC_2 is performed at time t_2 and the video session is stopped by the request PR_1. With such dynamic construction process, external interrupts can be quickly responded and scheduling decision can be carried out consequently.

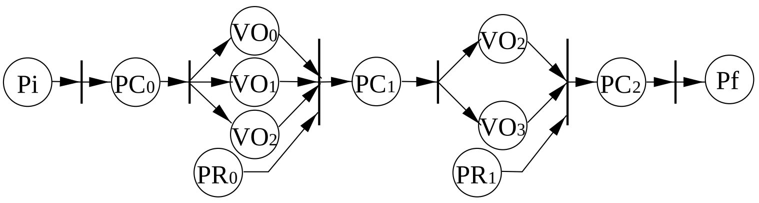

Fig. 5. Construction of OCPN model on VO level.

3.2 Dynamic Scheduling of Multiple Objects

Using the information generated by the model, the video objects need to be scheduled to multiple workstations for concurrent encoding. The objective of a scheduling algorithm in a parallel processing environment is to minimize the overall execution time of a concurrent program by properly allocating the tasks to the processors and sequencing their executions [3]. A scheduling algorithm can be characterized as being either *static* or *dynamic*. Static scheduling incurs little run-time cost but cannot adapt to the indeterministic behaviour of the system. On the other hand, although dynamic scheduling is more flexible as it can be adjusted to system changes, it incurs a high run-time cost. In a MPEG-4 video session, any static scheduling method may cause load imbalance due to the unpredictable behaviour of the video objects with varying computation requirement.

In order to solve such problem, we have to observe object variation and adjust the processor distribution of each object dynamically, while this may also introduce high inter-processor communication cost which may outweigh the benefit achieved. To compromise such trade-off in a natural way, we propose a dynamic scheduling algorithm, called *GOV-Adjusting Scheduling* (GAS) algorithm where GOV stands for Group of VOP, which periodically detects the workload information of the processors and performs the scheduling. In order to minimize the inter-processor communication cost, the period is based on the Group of VOP (GOV). GOV is an optional syntax level specified by the standard for random access and error recovery purpose. The GOV header usually is followed by the I-VOP performing Intra-coding which is independent to the previous VOPs and no data exchange needed. Therefore, the re-assignment of the processors will not introduce additional inter-processor communication.

Some of the additional notations used in the algorithms are presented below:
V: is scheduling sequence of the video session
P_k: is the k-th processor of the system, and total number of processors is K
R_k: is the scheduler on P_k
L_x: is the x-th GOV intervals
$G_i : M \rightarrow g_i$ where $g_i \in I$, is a mapping from the set of tokens to a set of groups G_i containing g_i processors each. Basically, such mapping describes the processors distribution with the scheduling scheme
υ_k: is the partitioned data area of the VOP to P_k
τ_i: is the synchronization interval of the video object VO_i.

GAS Algorithm:

1. Initialize the GOV interval L_x
2. Sort VOPs of each existing video object VO_i in V using EDF rule
3. Measure the shape size of the first I-VOP along the L_x as S_i
4. Initialize $R_k = \{\}$ and calculate the g_i for processors distribution

$$g_i = \begin{cases} \left\lfloor K \times \frac{S_i/\tau_i}{\sum_{i=0}^{N-1} S_i/\tau_i} \right\rfloor & \text{if } \left\lfloor K \times \frac{S_i/\tau_i}{\sum_{i=0}^{N-1} S_i/\tau_i} \right\rfloor > 1 \\ 1 & \text{otherwise} \end{cases}$$

where $i = 0, 1, \ldots, N-2$, and

$$g_{N-1} = K - \sum_{i=1}^{N-2} g_i$$

5. Decide whether task merging should be performed or not
6. Point to the first $VOP_{i,j}$ of V
7. Schedule R_k to the pointing $VOP_{i,j}$ with $P_k \in G_i$ and $\{R_k = R_k \cup VOP_{i,j}\}$
8. Partition the VOP and map the data area υ_k to P_k where $\upsilon_k \approx \frac{S_i}{g_i}$
9. Advance the $VOP_{i,j}$ to the next $VOP_{i,j+1}$ along the V
10. Repeat last three steps until the end of the GOV
11. Goto step 2 and start the next GOV scheduling until the end of the video session

GAS algorithm divides the existing processors into a number of groups and each group handles single video object concurrently. Such allocation is performed periodically on the basis of the GOV. Within each group, a balanced data partitioning method [2] is employed for further encoding speedup.

An earliest-deadline-first (EDF) rule is applied in the second step of GAS algorithm, it specified that the tasks with earlier deadlines are assigned higher priorities and are executed before tasks with lower priorities. In our implementation, VOPs with the earlier playout deadlines or synchronization constraints are encoded and delivered first.

The criterion of processor allocation is tied to the shape size and playout duration as Step 4 shows, namely, the larger object size, the more processors assigned; the shorter the playout duration, the more processors assigned. While this may cause load imbalance between the groups since the distribution of the processors may not be proportional to the size of the video objects due to the excessive difference between the object size. One feasible solution is to merge the smallest object tasks together recursively as Step 5 indicates.

Such task merging is usually performed whenever the number of tasks is greater than the number of physical processors by merging a pair of tasks into a single co-task. There are various merging approaches for different purpose. While In our approach, such task merging step is to guarantee the load balancing among the processors. We define the GOV of each video object as a task, and find a pair of tasks which have minimum workload among all the clusters; then the pair of tasks are merged as one task and such operation is recursively executed until the load balancing can be met, namely, the distribution ratio of the processors are nearly equal to the size ratio of the video objects.

4 Experimental Results

We have tested the GAS algorithm on several composed sequences to demonstrate the performance achieved. All the video objects, such as *Akiyo*, *News1* and *Weather* as labeled in Fig.6, are obtained from the MPEG-4 standard test library with QCIF format and represent various characteristics in terms of spatial detail and movement.

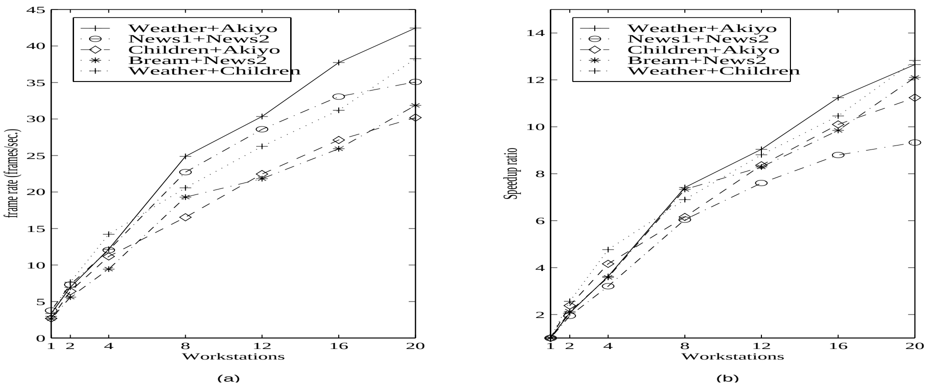

Fig. 6. Encoder performance

Our software-based implementation is applied on the MPEG-4 video verification model (VM8.0) encoder. The parallel platform is a cluster of 20 Sparc Ultra 1 workstations connected by a ForeSystems ATM switch (ASX-1000) which provides fast communication among the workstations. For inter-processor communication and synchronization, we use *Message Passing Interface* (MPI) [9], which ensures the portability of our MPEG-4 video encoder across various platforms. Furthermore, various additional software optimization, such as fast motion estimation algorithm, *Visual Instruction Set* (VIS) and Solaris C compiler optimizations, have been made to further speedup the encoder computation.

Fig.6(a) shows the encoding frame rate achieved by the GAS algorithms for the sequences with two video objects using various number of workstations. We can achieve frame rate higher than the real-time performance (30 frames/second) on most standard test sequences. Fig.6(b) is the overall speedup, a near-linear speedup relationship demonstrates that the performance of the encoder can scale according to the number of workstations used.

5 Conclusions

In this paper a software-based parallel implementation of MPEG-4 video encoder using a cluster of workstations has been proposed. The contribution of our work includes the use of an OCPN model to capture the spatio-temporal relations between multiple objects of MPEG-4 video, and a dynamic scheduling algorithm to implement MPEG-4 video encoder for multiple video objects. In our future work, we will explore MPEG-4 decoders and interactive methodology for supporting multimedia applications.

Acknowledgments

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

References

1. L. Chiariglione, “MPEG and Multimedia Communications,” *IEEE Transactions on CSVT*, vol. 7, no. 1, pp. 5-18, Feb. 1997.
2. Y. He, I. Ahmad and M. L. Liou, “Real-Time Distributed and Parallel Processing for MPEG-4,” *Proceedings of the 1998 International Symposium on Circuits and Systems*, vol. 3, pp. 603-606, 1998.
3. Y. K. Kwok and I. Ahmad, “Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 7, no. 5, pp.506-521, May 1996.
4. ISO/IEC, “MPEG-4 Version 1 Overview,” JTC1/SC29/WG11 N2323, July 1998.
5. ISO/IEC, "MPEG-4 Applications Document,” JTC1/SC29/WG11 N2322, July 1998.
6. ISO/IEC, "MPEG-4 Video Verification Model 8.0,” JTC1/SC29/WG11 N1796, July 1997.
7. T.D.C. Little and A. Ghafoor, “Synchronization and Storage Models for Multimedia Objects,” *IEEE Journal on Selected Areas in Communication*, vol. 8, pp. 413-427, Apr. 1990.
8. T. Sikora, “The MPEG-4 Video Standard Verification Model,” *IEEE Transactions on CSVT*, vol. 7, no. 1, pp. 19-31, Feb. 1997.

9. D. W. Walker, and J. J. Dongarra, "MPI: a Standard Message Passing Interface," *Supercomputer*, vol. 12, no. 1, pp. 56-68, Jan. 1996.
10. M. Woo, N. U. Qazi, and A. Ghafoor, "A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information," *IEEE Network*, vol. 8, pp. 52-61, Jan. 1994.