An Effective Parallelizing Scheme of MPEG-1 Video Encoding on Ethernet-Connected Workstations

Jongho Nang
Dept. of Computer Science
Sogang University
1 Shinsoo-Dong, Mapo-Ku
Seoul 121-742, Korea
e-mail : jhnang@ccs.sogang.ac.kr

Junwha Kim
Computer System Division
Samsung Electronics Co., Ltd
144-17 Samsung-Dong, Kangnam-Ku
Seoul 135-090, Korea
e-mail : kimjuna@janus.sst.co.kr

Abstract

Although MPEG-1 Video is a promising and the most widely used moving picture compression standard, it requires a lot of computational resources to encode the moving pictures with a reasonable frame size and quality. In this paper, we propose and implement an efficient parallelizing scheme of MPEG-1 Video encoding algorithm on an Ethernet-connected workstations which is the most widely available computing environment nowadays. In this parallelizing scheme, the slice-level, frame-level, and GOP (Group of Pictures)-level parallelisms are identified as the attractive parallelisms that can be exploited in Ethernet-connected workstations. Three efficient parallel implementation schemes considering the communication characteristics of Ethernet-connected workstations are also proposed and experimented. A series of experiments using thirty workstations shows that the MPEG-1 Video encoding time can be reduced in proportional to the number of workstations used in encoding computations, although there is a saturation point in the speedup graphs.

1 Introduction

Multimedia systems combine a variety of data, such as voice, graphics, animation, images, audio, and the most importantly full-motion video, into a wide range of applications [1]. Non-trivial amounts of these data in their uncompressed form, especially moving pictures, require storage and network capacities that will not be available in the near future. This fact has stimulated the multimedia researchers to develop several multimedia data compression standards such as JPEG [10], H.261 [7], and MPEG [6, 2]. Since the full-motion video in a raw digital format requires huge amount of storage resources, MPEG-1 Video encoding is the most important one to reduce the storage and network bandwidth requirements of multimedia applications based on full-motion video. Although it can achieve the compression ratios of upto 200:1 by storing only the differences between successive frames, it also requires a lot of computational resources to compress the full-motion video with reasonable frame size and quality. Furthermore, since the MPEG-1 Video standard specification only defines the process of decoding but not the decoder or encoder itself, the encoder or decoder should also be flexible to test several possible implementation schemes.

There have been basically two approaches to implement the MPEG-1 Video compression algorithms; the one is the hardware approach in which the MPEG-1 Video encoding algorithms are fixed in hardware so that they are fast but expensive and inflexible, and the other is the software approach in which the encoding algorithms are implemented in software so that they are flexible but slow. In order to resolve the problems in above two approaches (*i.e.* to develop a flexible and high performance MPEG-1 Video encoding system), there also have been some research efforts to implement MPEG-1 Video encoding algorithm on parallel computers [8, 9], or a network of workstations [4], However, their systems have been implemented on an expensive multiprocessor system while exploiting the fine-grained parallelisms of MPEG-1 Video encoding as in [8, 9], or only the performance of distributed MPEG-1 Video encoding exploiting the coarse-grained parallelism on an Ethernet-connected nine workstations was presented as in [4].

This paper also aims at developing a flexible and fast parallel MPEG-1 Video encoder, but the target computing environment is an Ethernet-connected workstations which is the most widely available computing environments in universities and research institutes. In the proposed parallelizing scheme, the slice-level, frame-level, and GOP (Group of Pictures)-level parallelisms of MPEG-1 Video encoding

0-8186-7876-3/96 $10.00 © 1997 IEEE

are identified as the attractive ones that can be exploited in Ethernet-connected workstations. The main ideas of these parallelizing schemes are to broadcast related data together in order to minimize the number of required communications when exploiting the slice-level parallelism, and to use the original frame as the reference frame in order to break the encoding dependencies between successive frames when exploiting frame- or GOP-level parallelisms. Three efficient parallel implementation schemes considering the communication characteristics of Ethernet-connected workstations such as efficient broadcasting and heavy communication setup time are proposed and experimented.

This paper also presents the experimental speedups of proposed parallelizing schemes on an Ethernet-connected thirty workstations to show the usefulness of our parallelizing schemes. A series of experiments on this computing environment shows that the MPEG-1 Video encoding time can be reduced proportional to the number of workstations used in encoding computations, and the parallelizing scheme exploiting the GOP-level parallelism produces the better speedups than others because it requires less communication overhead during parallel encoding process. We also find out through experiments that there is a cost-effective number of workstations for parallel MPEG encoding on these computing environments, so that even though more workstations are added into the parallel encoding, the speedup ratio is not improved so much.

This paper is organized as follows. First, the basic MPEG-1 Video encoding algorithm and its potential parallelisms are presented in Section 2. Then, three parallelizing schemes exploiting the slice-, frame-, and GOP-level parallelisms are presented one by one in Section 3. The implementations of proposed parallelizing schemes and their experimental speedups on an Ethernet-connected thirty workstation are presented in Section 4, while a summary and contributions of our works are finally presented in Section 5.

2 MPEG-1 Video Encoding Algorithms

The basic idea behind MPEG-1 Video compression is to remove spatial redundancy within a video frame via DCT(Discrete Cosine Transfer)-based compression as in JPEG, and the temporal redundancy between frames via motion compensation which is to encode a video frame based on other video frames temporally close to it. Let us explain the MPEG-1 Video encoding in more detail.

2.1 MPEG-1 Video Encoding

A video stream is a sequence of video frames each of which is a still image. Frames are digitized in a standard RGB format, 24 bits for pixel (8 bits each for Red, Green, and Blue). Frames can be encoded in three types: *intra-frame (I-frame)* which is encoded as a single frame without no reference to any part of future frames, *forward predicted frame (P-frame)* which is encoded relative to the closest preceding reference I- or P-frame, and *bi-directional predicted frame (B-frame)* which is encoded relative to the past reference frame, the future frame, or both frames [5]. A typical IBP pattern (called *coding pattern*) is shown in <Figure-1>, in which the arrows represent the inter-frame dependencies.

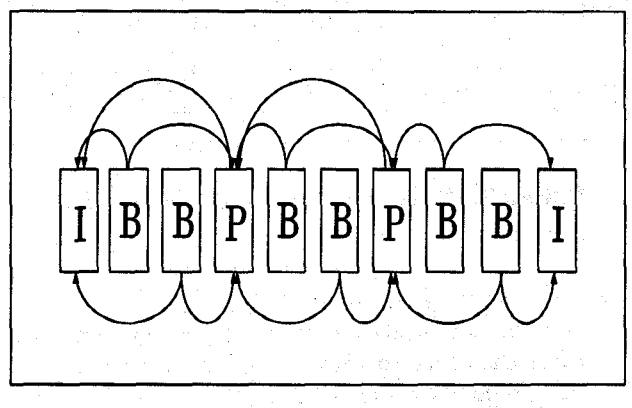

Figure 1. Inter-Frame Dependencies in MPEG-1 Video Encoding

Each video stream is composed of a series of *Group of Pictures (GOP's)* that is intended for random access in the video sequence. A GOP is composed of a sequence of *pictures (frames)*, and a picture is composed of a series of *slices* that is intended to allow decoding in the presence of errors. A slice is composed of series of *macroblocks*, and a macroblock is composed of 6 *blocks* (4 for luminance and 2 for chrominance). <Figure-2> shows these MPEG-1 Video bitstream hierarchy.

2.2 Parallelisms in MPEG-1 Video Encoding Algorithm

The parallelisms embedded in the MPEG-Video encoding can be classified as follows with respect to the granularities of parallelizable computation units.

- **Macroblock-level Parallelism**
  The computations for a macroblock itself (such as motion-vector estimation and DCT) can be encoded in parallel if each macroblock employs independent DC value. However, since the DC value of one macroblock usually depends on the DC-value of the previous macroblock, it can not be parallelized easily. It is the most fine-grained parallelism in MPEG-Video encoding, so that it is usually exploited in massively parallel computers such as MasPar MP-1 [9].

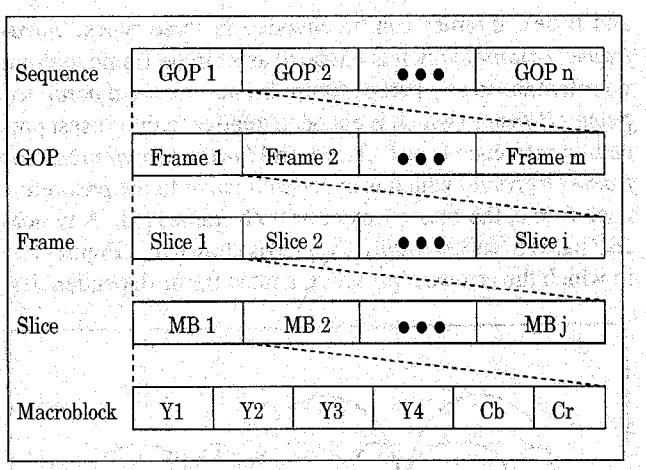

Figure 2. MPEG-1 Video Stream Hierarchy

- **Slice-Level Parallelism**
  The slice is a collection of contiguous macroblocks. Since each slice shares no global information, it can be encoded in parallel. Usually, the length of slice (*i.e.* the number of macroblocks in a slice) is an adjustable parameter in MPEG-1 encoding. The shorter slice implies the more parallelisms.

- **Frame-Level Parallelism**
  Since there are dependencies between successive frames owing to its coding pattern, each frame can not be encoded in parallel in principle. However, if the reference frames for each frame are available when it is encoded, the frame itself can be a parallelizable unit. The number of required reference frames is dependent on the type of encoded frame, *i.e.* the encoding of P-frame requires only one reference frame, whereas the encoding of B-frame requires two reference frames as shown in <Figure-1>. Of course, this assumption is true only if the original frame (not the encoded frame) is used as the reference frame in the encoding process.

- **GOP-Level Parallelism**
  Since each GOP contains almost all information for encoding process, it can be the largest parallelizable unit. However, since the last B-frame references the successive I-frame, there should be some considerations to satisfy this requirement.

The first one is relatively fine-grained parallelism, whereas the last two are relatively coarsed-grained one. A parallel MPEG-1 Video encoding algorithm can exploit the one of above parallelisms, or a combination of them.

3 Parallelizing MPEG-1 Video Encoding Computations

The main ideas of our parallelizing schemes are to use the original frame as the reference frame in order to break the encoding dependencies between successive frames, and broadcast related data together in order to minimize the number of required communications. Let us explain how these parallelisms could be exploited efficiently on an Ethernet-connected workstations.

3.1 Exploiting Macroblock-Level Parallelism

In order to exploit the macroblock-level parallelism on an Ethernet-connected workstations, each macroblock should be distributed across slaves, and the results (encoded with independent DC values) are gathered in the master workstation. Since this parallelizing scheme requires a lot of small communications to distribute the fine-grained tasks and to gather their results via Ethernet frequently, the expected performance gains may be zero or negative. This analysis forces us to give up to exploit this level of parallelism on an Ethernet-connected workstations.

3.2 Exploiting Slice-Level Parallelism

Since each slice is encoded with own DC values, it can be independently encoded in parallel. The most trivial way to exploiting this level of parallelism is to send each slice to different slave, and gather the encoded results. However, since this distribution strategy causes a lot of small amount of communication between master and slaves for distributing the slices, its performance may be low on such an communication environment as Ethernet where the setup time is a dominant factor in communication time when the size of a message is relatively small. In order to resolve this problem, we developed a distribution strategy, in which the master first broadcasts a frame composed of several slices to all slaves and then sends the assigned slice number to each slave. After each slave encodes the assigned slice, it sends the encoded result to master, and is assigned another slice number for encoding. This procedure is repeated until all slices consisting the frame are encoded. This parallelizing scheme alleviates the communication overhead for distributing relatively small sized slice, while exploiting the slice-level parallelism efficiently.

3.3 Exploiting Frame-Level Parallelism

Since the encoding of B- or P-frames requires the encoded results of previous I- or P-frames for referencing, each frame except I-frame can not be independently encoded in parallel. However, if we use the raw frame as

the reference frame when encoding B- or P-frame, each frame can be encoded in parallel. Although this referencing scheme is somewhat different from the original MPEG-1 Video algorithm, there is usually little difference in quality of encoded frames (about 5% differs as shown in [4]). In our parallelizing scheme exploiting frame-level parallelism, the master distributes the frames to each slave together with their reference frames. It means that if a slave is assigned to encode a B-frame, it receives the assigned frame together with two extra frames for referencing. After it finishes the encoding of the given frame, it sends back the encoded frame to master, and is assigned another frame until all frames are encoded. Although this parallelizing scheme sends the extra frames to slave, it can break the encoding dependencies between frames, hence, allows us to exploit the frame-level parallelism efficiently.

3.4 Exploiting GOP-Level Parallelism

Although each GOP contains almost all information for encoding process, the last frame (and/or the first frame) can not be encoded independently if it is B- or P-frame because it requires the first decoded frame of the successive GOP (and/or the last decoded frame of previous GOP) for referencing. However, if we use the original frame as a reference frame, this problem also can be resolved. It means that when master assigns and sends a GOP to slave for encoding, it also sends required reference frames in the original form together. Although the overhead for sending this extra frame may increase the total distribution time, it helps to break the dependency between successive GOP's without degrading the quality of encoded MPEG-1 Video stream as in the case of exploiting frame-level parallelism. Furthermore, this extra communication overhead could be negligible if the length of the GOP is relatively long.

The main design issue exploiting this level of parallelism is the coding pattern of GOP. If we use the coding pattern of BIBBPB, we need two reference frames for encoding this GOP (one for the first B-frame and the other for the last B-frame). On the other hands, if we use the coding pattern starting with I-frame such as IBBPBB, we need only one extra reference frame for encoding the last B-frame. This coding pattern helps to reduce the communication overhead for extra reference frames. Another consideration for deciding the coding pattern of GOP is its length. If its length is long (*i.e.*, the GOP consists of a large number of frames), the granularity of the parallelism could be bigger. Although this encoding strategy may be good for parallelization on the communication environment such as Ethernet, it also introduces the problem of load-balancing between slaves. The effects of these parameters on the speedups will be shown experimentally in the next section.

4 Implementations and Experimental Results

The parallelizing schemes proposed in the previous section are implemented and experimented on an Ethernet-connected thirty workstations. Let us explain the implementations and their experimental results together with analyses in more detail.

4.1 Implementations

As shown in <Figure 3>, our parallel MPEG-1 video encoding system consists of a master workstation which schedules the encoding tasks and distributes the input video streams, and several slave workstations which actually perform the encoding.

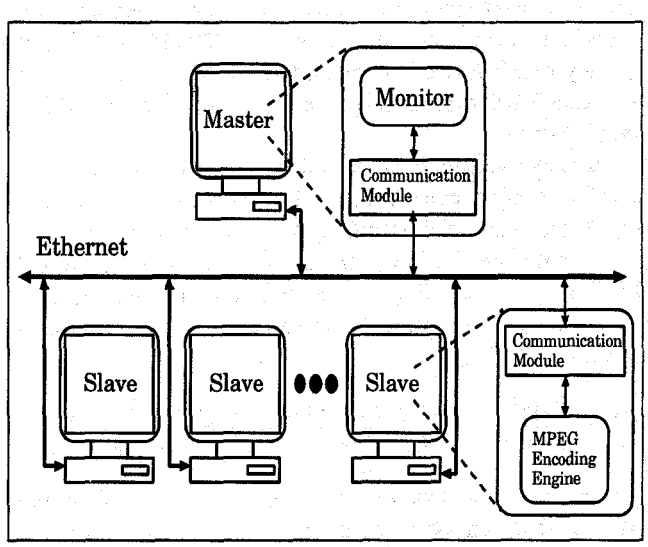

Figure 3. The Structure of Parallel MPEG-1 Video Encoding System

The MPEG-1 Video encoding engine used in slave workstations is a modified version of Berkeley MPEG encoder [4], and the communication library used to distribute the video stream and gather the encoded MPEG stream is PVM version 3.3.4 [3]. The distributed data unit for encoding can be the slice, frame, or GOP according to the exploited parallelism. When the master workstation distributes the video stream to slaves, it first compresses the video stream with GNU gzip on the fly before sending to the slave workstations, and the slave workstation uses the GNU gunzip to get the original input video stream before starting the MPEG-1 Video compression process. Although this distribution strategy requires additional time to compress and decompress the video streams, it helps to reduce the data distribution time between master and slaves.

Since the encoding time depends on the bit patterns of input video stream as well as the computing power of each slave workstations, a dynamic scheduling strategy is used in our implementation. It means that rather than the master workstation statically distributes the part of video stream to slaves, each slave dynamically requests and encodes the input video stream when it becomes idle. Although this strategy may require additional communication for scheduling, it helps to alleviate the heterogeneity of the computing environment such as Ethernet-connected workstations.

4.2 Experimental Results and Analyses

Three parallelizing schemes proposed in the previous section are experimented on an Ethernet-connected one SUN-20 compatible workstation[1] for master and thirty SUN-Classic workstations[2] for slaves. The input data used for experiments is a video stream (size of 320x240 with a frequency of 30 Hz) of 16 seconds long (about 500 frames stored in YUV format) showing a street scene captured at a moving car.

In principle, the exact speedup should be computed as a ratio of the execution time of parallel MPEG-1 encoding to the sequential MPEG-1 encoding. However, in our experiments and analyses, the p-slaves speedup of parallel MPEG-Video encoding, S^p, is computed using following equation in order to show the scalability of our parallelizing schemes;

$$S^p = \frac{Total\ Encoding\ Time\ running\ with\ one\ slave}{Total\ Encoding\ Time\ running\ with\ p\ slaves}$$

<Figure 4> shows the experimental speedups of parallel MPEG-Video encoding when exploiting the slice-level, the frame-level, and the GOP-level parallelisms as a function of the number of slave workstations involved in the parallel encoding. This speedup graph shows that the MPEG-1 Video parallel encoding time can be reduced in proportional to the number of workstations used in encoding process when the number of workstation is less than about 18 (actually with almost linear speedup), however, the speedup is not increased so much when more than 18 workstations are used in the encoding process. This speedup behavior comes from the fact that the ratio of communication time to the total encoding time is increased when more than 18 workstations are used in the parallel encoding process so that the merits obtained from the parallel encoding is offset by increased communication overhead.

<Figure 5> shows this situation in more detail, in which the the ratio of communication time (which also includes the

[1]Hyudai Axil 320 (CPU: Superscalar SPARC (60MHz), Main Memory : 64MB)

[2]Hyudai Axil 220 (CPU: MicroSPARC (50 MHz), Main Memory: 16MB)

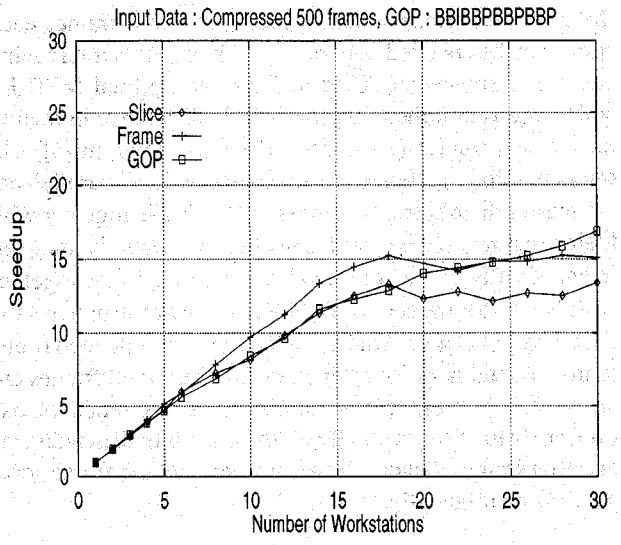

Figure 4. The Experimental Speedup of Parallel MPEG Video Encoding

waiting time) to the total encoding time for each exploited parallelism is shown as a function of the number of slave workstations. We can argue from this experiments that the communication bandwidth of Ethernet becomes a bottleneck of parallel encoding only when the number of workstations used in parallel encoding is more than 18 in our computing environment. This finding is against the usual assumption that MPEG-1 Video encoding can not be parallelized efficiently on Ethernet-connected workstation regardless of the number of slave workstations because of its low communication bandwidth. The main sources of this remarkable speedup are firstly our data distribution strategy in which the input video stream is compressed with simple compression algorithm so that the data distribution time can be reduced dramatically, and secondly its dynamic job assignment strategy which allows the slaves to overlap the communication and computations each other.

The other thing we can noticed from these figures is that the parallelizing scheme exploiting the GOP-level one produces a better speedup than others because the number of required communications between master and slaves is less than others. This number plays an important role in determining the total communication time especially in such a communication environment where the communication setup time is relatively larger than actual data transfer time as in Ethernet. An addition source of the more speedup is the fact that the parallelizing scheme exploiting the GOP-level parallelism requires less reference data relative to the assigned computation[3].

[3]Remember that in the case of exploiting frame-level parallelism, two addition reference frames are required to encode single frame, whereas only

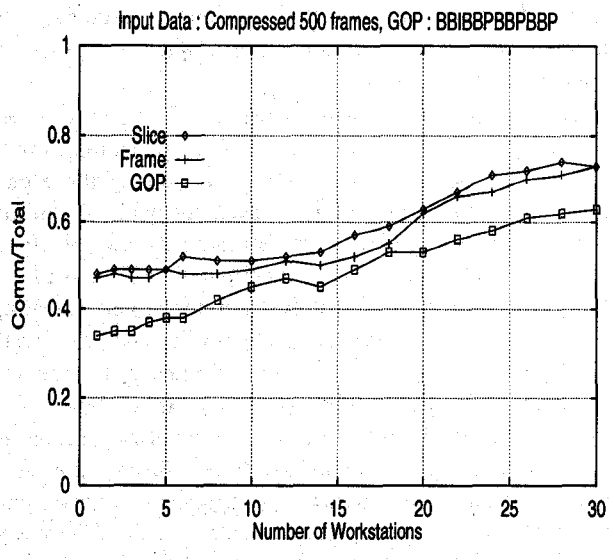

Figure 5. The Ratio of Communication Time to Total Encoding Time

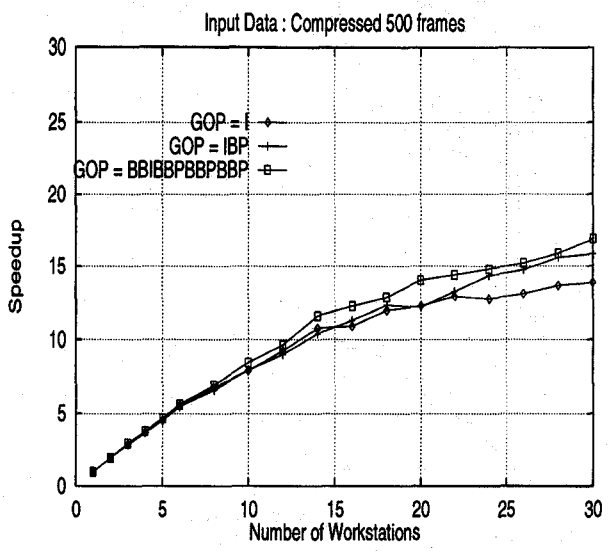

Figure 6. The Experimental Speedups with respect to the GOP Patterns

The GOP pattern is an adjustable encoding parameter in MPEG-Video encoding so that if there are more B-frames in a GOP pattern, more compression ratio can be obtained but more computations are also required. It is caused by the fact that encoding a video frame as a B-frame requires a lot of motion vector searches in order to find the similar macroblock in past or future I- and P-frames. This encoding algorithm implies that more B frames are used in GOP, the more computations required in encoding process of GOP. One can infer from this observation that more speedups can be obtained if a GOP pattern with a larger number of B-frames is used in the parallel encoding in an Ethernet-connected workstations because the slave could do more works for encoding with the same amount of communication overhead for GOP. In this case the ratio of communication time to total encoding time can be reduced so that more speedup can be obtained. <Figure 6> shows the experimental speedup with respect to the used GOP patterns (I-frame only, IBP, and BBIBBPBBPBBP), in which the parallel encoding scheme using GOP pattern of BBIBBPBBPBBP produces a higher speedup but the difference is not so much.

Usually if more extensive (or powerful) motion vector estimation algorithm is used in P- or B-frame encoding, an encoder produces a higher quality MPEG-1 Video stream with a higher compression ratio. It implies that we can also obtain more speedup if more extensive motion vector estimation algorithm is used in the MPEG-1 Video encoding process because it also reduces the ratio of the communication time to the total encoding time. <Figure 7> shows the experimental speedups when more complicated motion vector estimation algorithm is used in the MPEG-1 Video encoding process.

A series of experiment results shown in <Figure 4>, <Figure 6>, and <Figure 7> allows us to argue that the parallelizing schemes presented in this paper can be efficiently implemented on an Ethernet-connected workstations with almost linear speedup when the number of used workstation is less than 18. Note also that there is a cost-effective number of workstations for parallel encoding on Ethernet so that although more workstation are used in the parallel encoding the speedup is not increased so much. Another finding is that more speedups can be obtained if GOP-level parallelism is exploited in a MPEG-1 Video stream which are encoded with a lot of B-frames, and an extensive motion estimation algorithm is used in the encoding process.

5 Comparisons with Related Works

There have been some research works to parallelize the MPEG-1 Video encoding on a parallel computers [9, 8] or a network of workstations [4]. Let us compare their works with the parallelizing schemes presented in this paper. <Table-1> summarizes the comparisons with related works.

M. Tan *et. al.* [9] have proposed and implemented a parallel motion vector estimation algorithm, which is the most time consuming process in MPEG-1 Video encoding, on MasPar MP-1 (a SIMD machine with 16384 PE) and PASM. The work proposed by Philips Research Lab.

one addition reference frame is required in the case of exploiting GOP-level parallelism

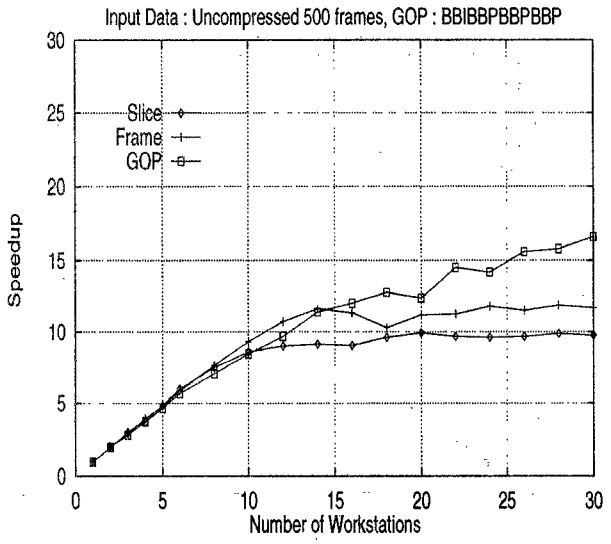

Figure 7. The Experimental Speedups with respect to the Complexity of Motion Vector Searching Algorithm

[8] also exploited a macro-level parallelism (especially focused on the motion vector estimation) in CD-I encoding on POOMA which is a distributed-memory multiprocessor with 100 MC68020 processors. However, since both parallelizing schemes require a lot of frequent communications and the granularities of parallelism are too small, their parallelizing schemes can not be used in a computing environment such as Ethernet-connected workstations.

The most related work to our parallelizing schemes may be the one proposed by Berkeley [4], in which a number of contiguous frames (not necessarily be a GOP) is encoded in parallel on a Ethernet-connected nine workstations. The UNIX TCP socket is used for implementing a dynamic load balancing scheme, and NFS is used to transmit raw frames separately stored in JPEG format on disk to the individual slave workstations for encoding itself and for the reference frames. In this work, however, only a sequence of frame-level parallelism regardless of frame types is considered on Ethernet-connected nine workstations. Furthermore, since their experiments are too small scale so that they did not find the saturation point of the speedups when the number of workstations is larger than a specific number (for example 18 workstation as in our experiments).

6 Summary

Since the MPEG-1 Video encoding process requires a lot of computations, there has been several researches to parallelize it. The objective of our research was to develop a flexible and fast MPEG-1 Video encoding system on a widespread computing environment such as Ethernet-connected workstations.

This paper have presented three parallelizing schemes exploiting the slice-, frame-, and GOP-level parallelisms of MPEG-1 Video encoding. The main ideas of these parallelizing schemes are firstly to send the whole frame to all slaves using the efficient broadcasting facility of Ethernet when exploiting the slice-level parallelism, and to use the original frame as the reference frame when exploiting the frame- and GOP-level parallelism which helps to break the dependencies between adjacent frames while keeping the quality of encoded MPEG-1 Video stream high. The proposed parallelizing schemes were experimented on an Ethernet-connected thirty workstations. From the experimental results, we found that exploiting the GOP-level parallelism produced a better speedup although the differences are not so much, and there is a cost-effective number of workstations for the parallel MPEG-1 Video encoding regradless of the parallelisms exploited. From these experimental results and analyses, we can argue that the MPEG-1 Video encoding could be efficiently parallelized with our parallelizing schemes on an a widespread computing environment such as Ethernet-connected workstations. These parallelizing and implementation schemes are fast because of their encoding capabilities with multiple workstations, and are flexible because of their software implementation natures.

The parallelizing schemes proposed in this paper can be used to develop a multimedia application system with MPEG-1 Video and to test the implementation methods of MPEG-1 Video compression standard, on a broadly available computing environment. Furthermore, since the basic video encoding algorithm used in MPEG-2, which is a standard for digital HDTV-quality broadcasting and its encoding requires more computational resources than MPEG-1 encoding, is almost the same as the MPEG-1 Video encoding algorithm, the proposed parallelizing schemes could be also applied to MPEG-2 Video encoding process directly.

References

[1] B. Furht, "Multimedia Systems: An Overview," *IEEE Multimedia*, Spring 1994, pp. 47-59.

[2] D. L. Gall, "MPEG : A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol.34, No.4, April 1991, pp. 46-58.

[3] A. Geist, A. Beguelin, et. al., *PVM 3 User's Guide and Reference Manual*, Oak Ridge National Laboratory, ORNL/TM-12187, 1993.

Table 1. Comparison with Related Works

	Philips Work[8]	Berkeley Work[4]	Oursfi
Exploited Parallelisms	Macroblock-level	Frame-level	Slice-, Frame, and GOP-level
Computing Environments	POOMA (100 PEs)	Ethernet-connected nine Workstations	Ethernet-connected thirty Workstations
Communications	POOL	NFS, UNIX Socket	PVM
Maximum Encoding Rate	0.6 frame/sec	4 frame/sec	7.8 frame/sec

[4] K. Gong and L. A. Rowe, "Parallel MPEG-1 Video Encoding," *Proceeding of 1994 Picture Coding Symposium,* 1994.

[5] http://www-plateau.cs.Berkeley.EDU:80/mpeg/mpeg_overview.html.

[6] ISO/IEC/JTC1/SC29/WG11, *Coding of Moving Pictures and Associated Audio,* ISO/IEC International Standard 11172-2, Mar. 1992.

[7] M. Liou, "Overview of the px64 kbps Video Coding Standard," *Communication of the ACM,* Vol. 34, No. 4, Apr. 1991, pp. 59-63.

[8] F. Sijstermans and J. Meer, "CD-I Full-Motion Video Encoding on a Parallel Computer," *Communications of the ACM,* Vol. 34, No. 4, Apr. 1991, pp. 81-91.

[9] M. Tan, J. Siegel, and H. Siegel, "Parallel Implementation of Block-Based Motion Vector Estimation for Video Compression on the MasPar MP-1 and PASM," *Proceeding of International Conference on Parallel Processing,* Vol. 3, 1995, pp. 21-24.

[10] G. Wallace, "The JPEG Still-picture Compression Standard," *Communication of the ACM,* Vol. 34, No. 4, Apr. 1991, pp. 30-34.