A Portable and Scalable MPEG-2 Video Encoder on Parallel and Distributed Computing Systems

Shahriar M. Akramullah†, Ishfaq Ahmad†, Ming L. Liou‡

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

Abstract

Traditionally, real-time video compression due to its enormous computing requirement has been done using the special-purpose hardware. On the other hand, software-based solutions have been primarily intended for non real-time applications. In this paper, we present a portable and scalable implementation of the MPEG-2 video encoder, using parallel processing, that can be used for both real-time and non-real-time applications. The portability allows it to run on a wide variety of platforms including a number of high-performance parallel computers as well as networks of workstations. The scalability allows to control the parallelism enabling it to run on a few fast workstations using a coarse granularity or on a massively parallel architecture using a fine grained granularity.

An important feature of our implementation is that we use a data-parallel approach and exploit parallelism within each frame, unlike previous parallel video coding implementations. This makes our encoder suitable for real-time applications where the complete video sequence may not be present on the disk and may become available on a frame-by-frame basis with time. The encoder also provides control over various parameters such as number of processors, size of motion-search window, buffer management and bit rate. Our implementation is flexible and allows inclusion of fast and new algorithms for different stages of the encoder, replacing current algorithms. Experimental results have been conducted on two parallel processing systems: the Intel Paragon XP/S and the Intel iPSC/860 hypercube. Networks of workstations used include the SUN and HP, connected via the Ethernet and FDDI, respectively. Comparisons of execution times, speeds as well as frame encoding rates on these systems are provided. Using maximum parallelism by dividing one block per processor, an encoding rate higher than real-time (30 frames/sec) has been achieved on the Intel Paragon.

1 Introduction

Due to the rapid progress in information technology, the traditional boundaries between the areas of telecommunication, computer and TV/film are blurring. As a result, multimedia-based computer systems, and interactive video are being added to telecommunication. The applications in these areas require ways for
coding standard established by the Motion Pictures Expert Group of the International Standards Organization (ISO). Being very comprehensive it is still evolving and is therefore chosen for our software implementation.

Our implementation is based on parallel and distributed computing, but is independent of any particular hardware. The three main features of our implementation are portability, flexibility, and scalability.

The portability of the code implies that the parallel software should be able to run on a vast variety of high-performance parallel supercomputers as well as general-purpose networks of workstations. Our implementation runs on high-performance parallel processing systems as well as networked clusters of workstations using which are becoming increasingly faster and cost effective. We have performed experiments on the Intel Paragon, the Intel iPSC/860 hypercube, and two different networks of workstations, namely HP 9000/735 and Sun IPX/Sparcstation.

When a problem is implemented on an architecture with multiple processors, it needs to be partitioned into smaller pieces which are then assigned to the processors. The granularity of the problem is the size of the computational piece (in terms of its computing requirement) assigned to a processor. The scalability of the implementation means that the granularity of the problem should be controllable without sacrificing the performance.

Flexibility means that the performance of the software in terms of its processing speed should not depend on a particular hardware configuration, rather it should optimize itself according to the available hardware. Scalability means that by increasing the computing resources, the processing speed should increase as linearly as possible.

An important feature of our implementation is that parallelism is exploited within each frame of a video using a data-parallel or single-program multiple-data (SPMD) programming paradigm, unlike previous parallel approaches ([11], [4], [8], [11], [12], [13]) in which a large video sequence is partitioned into pieces which are then independently encoded on multiple sequential encoders. This makes our encoder suitable for real-time implementations where the complete video may not be present on the disk and may become available on a frame-by-frame basis with time. The encoder provides control over various parameters such as number of processors used in each dimension, size of motion search window, buffer management and bit-rate. The implementation is scalable, and a linear speedup indicates that by using larger number of processors, we can have better performance irrespective of the picture resolution. In our implementation, flexibility is provided for inclusion of faster algorithms for various stages of MPEG-2 in place of current algorithms.

The rest of the paper is arranged as follows. Section 2 gives a brief overview of various video encoders adopted in different international standards. Section 3 introduces the MPEG-2 video encoder (sequential) describing its major computational modules. Section 4 lists related works and for some cases mentions the drawbacks. Section 5 presents the method of porting the sequential MPEG-2 video encoder to parallel and distributed systems. Section 6 provides the performance results while the last section concludes the paper.

2 Overview of Video Encoders

In this section, we present an overview of various encoders defined and/or adopted by different international standards.

2.1 JPEG

JPEG is an international standard developed by the JPEG (Joint Photographic Experts Group) of the ISO (International Standards Organization). This was principally aimed for continuous-tone color-image compression. It exploits the intraframe redundancy in images of any resolution [14]. The JPEG algorithm uses Discrete Cosine Transform (DCT) in order to convert an image from pixel domain to transform domain with a view to achieve spatial decorrelation of pixels (picture elements). The DCT coefficients are then quantized followed by Huffman coding of the quantized values, thus yielding compression. In addition to still images, video data can be compressed by using Motion JPEG, which does not employ any interframe redundancy reduction technique. Therefore, each frame is encoded independently and the encoding process produces higher bit-rate.

2.2 H.261

H.261 is an international standard for video codec developed by CCITT (currently ITU), targeted for low bit rate applications such as video teleconferencing or videophone over ISDN (Integrated Services Digital Network). In order to provide compatibility with ISDN data transmission rate, H.261 is designed for data rates that are multiples of 64 kbps and is therefore, alternatively known as \( p \times 64 \) kbps standard, where \( p = 1, 2, \ldots, 30 \). Unlike Motion JPEG, further to DCT-based spatial redundancy reduction, H.261 does take advantage of the temporal redundancy present among frames of a video sequence. Therefore, it can achieve lower bit-rate. Motion-compensated prediction is used to exploit interframe redundancy in temporal dimension.

2.3 MPEG-1

MPEG-1 is the first video codec standard developed by the MPEG (Moving Pictures Expert Group) of the ISO/IEC (International Standards Organization/International Electrotechnical Commission). This has primarily been intended for applications requiring medium quality (comparable to VHS video-tape with near-CD audio) and medium bit-rate (around 1.5 Mbps) video and audio
encoding. Similar to JPEG and H.261, MPEG-1 also employs a DCT-based codec algorithm in order to reduce spatial redundancy, but it exploits temporal redundancy as well. MPEG-1 introduces the concept of bi-directionally predicted frames, which are predicted from a past and a future reference frame.

2.4 MPEG-2

MPEG-2, developed by the MPEG of the ISO/IEC and ratified in December 1994, is the latest complete international standard available for video/audio codec. It is a generic standard intended for a wide range of applications, including video-on-demand (VOD), high-definition television (HDTV), video communication using asynchronous transfer mode (ATM) networks etc. MPEG-2 is aimed for applications requiring bit-rates of 2Mbps and above with a quality ranging from good quality NTSC to HDTV. MPEG-2 video is structured in six hierarchical layers: Sequence layer, Group of Pictures (GOP) layer, Picture layer, Slice layer, Macroblock layer and Block layer.

The bitstream syntax of MPEG-2 is divided into profiles and levels. Profiles are a set of tools which specify constraints on the syntax. There are five profiles defined by MPEG-2: Simple, Main, SNR scalable, Spatially scalable and High. Levels are sets of constraints imposed on parameters in the bitstream, defined within profiles. MPEG-2 defines four up to four levels: Simple, Main, High 1440 and High. Together, profiles and levels restrain the coding complexity within feasible limits of implementation, yet still address a wide variety of applications.

MPEG-2 provides compatibility with MPEG-1 such that an MPEG-1 compressed bit-stream is decodable using MPEG-2. The MPEG-2 bit-stream syntax, which provides this compatibility is known as non-scalable syntax possessing some extra compression tools. On the other hand, the scalable syntax structures the bit-stream in a base layer and one or more enhancement layers according to scalability modes (discussed below). The codec algorithm is also very similar to that defined by MPEG-1, i.e., DCT-based spatial redundancy reduction supported by motion-compensated temporal prediction including bi-directional prediction. Both the standards also use an abstract model of decoding, known as video buffering verifier, to ensure decodability within reasonable buffering and delay requirement. Nevertheless, MPEG-2 has a number of differences compared to MPEG-1. Major differences are mentioned below.

The principal difference between the two standards is that in addition to progressive video used by MPEG-1, MPEG-2 has the provision for interlaced video as well. Accordingly, it provides frame/field-based motion-compensated prediction and frame/field-based DCT modes. MPEG-2 has improved accuracy in motion estimation as it permits calculation of motion vectors up to half-pixel accuracy. MPEG-2 takes other measures also to improve the picture quality. The most important of these measures are its four scalability modes — namely, Spatial scalability, Temporal scalability, SNR scalability and Data partitioning — which prioritize video data by forming layers of compressed video and by transmitting more important data with larger security in an error-prone environment. In addition, MPEG-2 provides concealment motion vectors for I-pictures in order to increase robustness from bit errors. MPEG-2 introduces variable bit-rate along with usual constant bit-rate. On top of zig-zag scanning pattern, it also offers alternate scanning pattern, which improves entropy coding performance, specially for interlaced video. Moreover, MPEG-2 presents two color spaces, namely 4:2:2 and 4:4:4, in addition to 4:2:0 used by MPEG-1.

3 The Structure of MPEG-2 Algorithm

Obtaining good picture quality at the bit rates of interest demands very high compression, which is not achievable with intraframe coding alone. The random access requirement, however, is best satisfied with pure intraframe coding. This necessitates a delicate balance between intra- and inter-frame coding and between recursive and non-recursive temporal redundancy reduction. This leads to the definition of Intra-coded (I), Predictive coded (P) and Bidirectionally predictive coded (B) pictures. This idea is illustrated in Figure 1.
for future prediction. P-pictures are coded more efficiently from a previous I- or P-picture and are generally used as reference pictures for further prediction. B-pictures provide the highest degree of compression but require both past and future reference pictures for motion compensated prediction. The display order of these frames (I, B, B, P, B, ...), for the example shown in Figure 1, would be different from the transmission order (I, P, B, B, P, ...).

The algorithm first selects an appropriate spatial resolution for the signal. It then uses block-based motion-compensated prediction for the temporal redundancy reduction, which falls into the temporal DPCM (Differential Pulse Code Modulation) category. Motion compensation is used both for causal prediction of the current picture from a previous reference picture, and for non-causal, interpolative prediction from past and future reference pictures. Next in the algorithm is the stage of motion estimation, which covers a set of techniques used to extract the motion information from a video sequence. Motion vectors are defined for each 16-pixel by 16-line region of the picture. In MPEG-2, motion estimation is done by using block-matching technique using the previous frame and the future frame.

In order to achieve spatial redundancy reduction, the difference signal, i.e., the prediction error, is further compressed using the block transform coding technique which employs the two-dimensional orthonormal 8 × 8 DCT (Discrete Cosine Transform) to remove spatial correlation. The resulting 63 AC transform coefficients are mapped in an alternate scanning pattern (or zig-zag scanning pattern when providing compatibility to MPEG-1) before it is quantized in an irreversible process that discards the less important information. Luminance and chrominance components of the frame data share the same quantization tables. In MPEG-2, adaptive quantization is used at the macroblock (16 × 16 pel area) layer, which permits smooth bit-rate control as well as perceptually uniform quantization throughout the picture and image sequence. Finally, the motion vectors are combined with the residual DCT information, and transmitted using variable length codes. The variable length coding tables are non-downloadable and are therefore optimized for a limited range of compression ratios appropriate for the target applications. A block diagram of the complete encoder is given in Figure 2.

4 Related Work

There have been some previous approaches — mostly using special hardware — to parallelize codec operations of video sequences. For example, the CD-I full-motion video encoding (non-standard) has been implemented on a parallel computer [12]. It employs an approach where parallelization is done by dividing the stages of the video codec into tasks and assigning one task to a group of processors. This has the disadvantage that many frames must be read before all processors have some task to execute. Furthermore, this implementation requires a special-purpose hardware.

An MPEG-1 encoder has been implemented in [13] with a special-purpose parallel hardware using DSP processors. This has the drawback that the parallel programs are non-portable and must match the ability of the special hardware which has limited capability in signal processing. Another hardware implementation of MPEG-2 using image compression DSP with good performance has been described in [1].

A parallel implementation of the H.261 video coding algorithm using a single-instruction multiple-data (SIMD) parallel machine has been reported in [8]. It suggests that parallelization can be done either on individual pixels, on blocks or on other groups of the frame data, while this implementation chose to distribute data-blocks among processors. However, due to the nature of SIMD paradigm,
only certain parts of H.261 were parallelized. Consequently the implementation of rate control became the bottleneck in processing and could only achieve a frame rate of about 5 frames/sec.

Parallel MPEG-1 video encoding with a performance of about 4 frames/sec using 9 HP 9000/720 machines connected via ethernet has been documented in [4]. It has been later modified as described in [11] to run on Intel Touchstone Delta and Intel Paragon. Although it has been claimed that faster than real-time performance was achieved, this scheme has several drawbacks. First, it divides the video sequence into different sections and assigns those sections to different processors. Each processor runs the same sequential encoding program, but compresses different part of the video (in particular, different Group of Pictures) in parallel. Therefore, the video sequence should be available before encoding begins. Second, the compressed data from each processor has to be concatenated off-line. Third, since it was found inefficient to open a large number of small files for input and output, this implementation has to group sections of consecutive frames into a single file. To get the most efficient performance using this approach, a processor should open and read from a single file having all the frames assigned to it, including the necessary reference frames. This means one can use only a limited number of processors to encode a video sequence of a given length, which restricts the scalability of the problem. Finally, it is depends on some special I/O operation capability offered by Intel Touchstone Delta or Intel Paragon in order to improve performance, and, therefore, is not portable to other hardware platforms, for instance, a network of workstations.

A slice-based software implementation of MPEG-2 video encoding is described in [16]. This implementation uses socket programming for a Local Area Network (LAN) of workstation for its parallelism.

5 Parallelizing the MPEG-2

In order to have a portable and scalable parallel implementation of the MPEG-2 video encoder, we have used the data parallelism or single program multiple data (SPMD) programming paradigm. Under this paradigm, the data is partitioned into smaller pieces which are assigned to different processors. A single program is written for all processors which execute the program on their piece of data in an asynchronous fashion. Communication of data and synchronization is done through message-passing. We have used Express parallel programming environment. In addition to providing message-passing support, Express’s Cubix programming model is able to set up a virtual processor grid regardless of the underlying hardware topology. In addition, it can map the data on to the virtual processor grid automatically.

The objective is to develop an implementation capable of running on various parallel and distributed platforms, based on a sequential implementation of MPEG-2 video encoder. The sequential implementation, which we have used as a benchmark, can be found in [9].

5.1 Data Partitioning

Since the overhead due to inevitable communication can be the major limiting factor, care should be taken while partitioning the data among the processors such that minimal interprocessor communication is employed. The data-distribution to the processors for the data-parallel approach is very simple. The whole frame is distributed as evenly as possible to each processor. It is possible, however, to partition the data by just apportioning the requisite part of the frame data (one or more 16 × 16 macroblock) to the corresponding processors as the processors are mapped onto the two dimensional grid. This method is shown in Figure 3. But it necessitates inevitable communication among the processors (see Figure 5) as the search window during motion estimation moves to the boundary.

On the contrary, since each processor has enough memory to store the entire search window, it is possible to eliminate use of overwhelming amount of communication. In this case, the frame data is distributed among the processors allowing overlap and using the data partitioning method as depicted in Figure 4. Here, each processor is allocated some redundant data, which is necessary to form the complete search area.

Let us consider \( P \) and \( Q \) to be the height and the width of the frame respectively, and let \( p \) be the total number of processors to be used, with \( p_h \) to be the number of processors in the horizontal dimension and \( p_v \) to be the number of processors in the vertical dimension of the two-dimensional grid. Thus, \( p = p_h \times p_v \). If the search window size is the size of the macroblocks in a particular processor ± \( W \) in both dimensions, with overlapped (redundant) data distribution, given \( p_h \) and \( p_v \), one can determine the size of the local frame in each processor, which is given by

\[
X_{\text{local}} = \left\lfloor \frac{Q}{p_h} + 2W \right\rfloor \times \left\lfloor \frac{P}{p_v} + 2W \right\rfloor
\]

(1)

It is easy to see that if one wants to avoid interprocessor communication when computing motion estimation, some additional memory is required for every processor to accommodate the redundant data necessary to form the search window. This is a realization of the popularly known communication-memory trade-off.

In the current implementation, the number of processors to be used is an input parameter. Therefore, it can be ported to environments with a few powerful processors to those with a large number of relatively slow processors as well as to hardware platforms with limited memory or slow communication.

5.2 An Upper Bound for Number of Processors

The maximum number of processors that may be used can be determined as follows. Let the macroblock size be \( w \times h \) (typically \( 16 \times 16 \)) for which motion vectors are to be determined. The frame data may be divided among
processors if we can make the search window available to the corresponding processor. This can be accomplished by distributing one or more 16 × 16 block to each processor, and then either sending the required boundary data to corresponding processors to form their local frame (see Figure 5), that is search window, or equivalently, storing the redundant data at the local memory of each processor, which is necessary to form the search window to be used for determination of motion vectors for the next frame. Both types of distribution of data are discussed in the previous section. Thus, one can have

\[ p_{h,\text{max}} = \left\lfloor \frac{Q}{w} \right\rfloor \quad \text{and} \quad p_{v,\text{max}} = \left\lfloor \frac{P}{h} \right\rfloor \quad (2) \]

Hence, for \( w = h = 16 \), maximum number of processors is

\[ p_{\text{max}} = \left\lfloor \frac{Q \times P}{16 \times 16} \right\rfloor \quad \Rightarrow \quad p \leq \left\lfloor \frac{Q \times P}{256} \right\rfloor \]

For example, if \( Q = 360, P = 240 \), then \( p_{h,\text{max}} = 22, p_{v,\text{max}} = 15 \) and consequently \( p \leq 330 \).

5.3 Implementation Features

We have parallelized various modules of MPEG-2. For DCT and IDCT, we have distributed the data such that each processor deals with one or more 8 × 8 pixel-block of data. Each processor performs DCT and IDCT on as many blocks belonging to its part of the frame data, while the same program runs on each processor. Therefore, there is no interprocessor data movement.

For motion estimation, we have used 16 × 16 pixel-blocks as basic unit of parallel processing. We have distributed the data among processors in a manner such that each processor has sufficient data (redundant data distribution, see § 5.1) to form the search window. The search window size is a user-defined parameter and used as input to our program. By using redundant data in each processor, we have avoided interprocessor communication.

In case of rate control and bit allocation, the global bit-budget for the current picture being coded is broadcasted to all the processors. In the parallel implementation, coding complexity is estimated locally as the product of the average macroblock quantization step size and the number of bits generated by each processor. The local bit allocation for the current macroblock is based on estimated buffer fullness in a particular processor and the normalized spatial activity, similar to the sequential implementation. The macroblock quantization scale is adapted according to the deviation of the locally generated bits and the estimated uniform distribution of bits in each processor, with the help of a compensation factor (difference between predicted and true buffer fullness of the current macroblock).

Our implementation allows up to three input formats:
separate YUV, combined YUV and PPM (Portable PixMap format). It generates constant bit-rate bit-streams besides being able to generate MPEG-1 bit-streams. It outputs the encoded sequence as well as relevant statistics and verifies legality of the user-given parameters within profile and level. The current implementation does not support variable bit rate encoding, scalable extensions, integer pel motion vectors for MPEG-1 (the implementation always produces half-pel motion vector, which nevertheless, gives better quality), low-delay\(^1\), concealment motion vectors\(^2\), editing of encoded video and scene change rate control.

5.4 Target Systems

We have used the Intel Paragon XP/S, the Intel iPSC/860 hypercube, a network of 4 HP 9000/735 workstations, and a network of 16 Sun IPX/Spacrstations. The systems are described below.

5.4.1 The Intel Paragon XP/S

The Paragon XP/S [5] from Intel Corporation was first delivered in September 1992. Similar to its predecessors the Touchstone Delta (prototypical) and the iPSC/860, the Paragon is also a multicomputer. Its nodes are based on Intel’s i860 XP processor and it primarily supports message passing as the programming model using a new OSF/I-based operating system.

From an architectural point of view, the Paragon is a distributed-memory MIMD machine. Its processing nodes are arranged in a two-dimensional rectangular grid. The system consists of three types of nodes: compute nodes, which are used for the execution of parallel programs; service nodes, which offer capabilities of a UNIX system, including compilers and program development tools; and I/O nodes, which are interface to mass storage and LANs.

Paragon Mesh Routing Chips (MRCs), connected by high-speed channels, are the basis of the communication network, where nodes may be attached. There are two independent channels — one for each direction — between neighboring nodes. The channels are 16 bits wide and have a theoretical bandwidth of 175 Mbps. The MRCs can route messages autonomously and are independent of the attached nodes. Communication is based on wormhole routing with a deterministic routing algorithm.

All the three types of nodes are implemented by the same General Purpose (GP) node hardware. A 32-bit address bus and a 64-bit, 50 MHz (i.e. 400 Mbps) data bus connects all the components of the GP node’s compute and network interface parts. The i860 XP is a Reduced Instruction Set Computer (RISC) processor with a clock speed of 50 MHz and a theoretical peak performance of 75 Mflops (64-bit arithmetic: 50 Mflops add, 50 Mflops multiply) and 100 Mflops (32-bit arithmetic: 50 Mflops add, 25 Mflops multiply). The interface of the GP node to the interconnection network performs such that message-passing is separated from computing. The actual transmission of data between the memory and the network is accomplished by the Network Interface Controller (NIC).

The Paragon’s primary mass storage system is disk array like Redundant Array of Inexpensive Disks (RAIDs) which can accommodate 4.2 Gbyte of data.

The Paragon’s operating system is called Paragon OSF/1, which provides an OSF/1-compatible application interface. On the compute nodes, the operating system and related buffers occupy more than 6 Mbyte. The OSF/1 servers and libraries jointly offer the view of a single UNIX-like system to the user. The NX message passing interface used in Paragon is a super-set of the iPSC/860’s NX/2. For our experiments, we have used a 140 node Paragon at the Hong Kong University of Science and Technology, Hong Kong, and a 512 node Paragon at the California Institute of Technology, USA.

5.4.2 The Intel iPSC/860 Hypercube

The iPSC/860 is an MIMD distributed-memory multicomputer consisting of compute nodes, I/O nodes, and a host computer. The compute nodes are i860-based processor boards. When they work en masse to solve a large problem, they achieve supercomputing performance. By configuration, the compute nodes are independent processor/memory pairs with capability of running distinct programs concurrently. They have distinct memory spaces and use message-passing in order to transfer any data from one node to another. The compute nodes are connected as a hypercube.

Each node consists of a 32-bit i860 RISC processor with up to 16 Mbyte of local memory. The compute nodes can be thought of belonging to a node network, where each node interfaces to the node network through its Direct-Connect Module (DCM). Essentially, communication between nearest neighbors occurs over the DCM channel. In the iPSC/860, communication is done using Direct-Connect Routing (DCR) to guarantee minimal time for interprocessor communication. If the message is not intended for its accompanying node, this hardware module passes the message on without interrupting the node’s processor.

---

1. A low-delay bitstream can be defined according to MPEG-2, where total encoding and decoding delay of less than 150 milliseconds can be achieved. Low-delay is particularly demanded by real-time video communications such as video telephony, video-conferencing, monitoring. The total encoding and decoding delay can be kept low by generating a bitstream which does not contain B-pictures. This prevents the frame reordering delay. By using dual-prime prediction for P-frames, the picture quality can still be high. A low buffer occupancy is needed for low delay, which results in VBV buffer underflow. To keep bit rate constant, some frames may need to be skipped, indicated to the decoder by the state of the VBV buffer.

2. Concealment motion vectors are vectors that may be carried by intra macroblocks for the purpose of concealing errors should data errors preclude decoding the coefficient data. They are intended for use in the case that a data error results in information being lost. It is less likely to be used and the vector (0,0) may be used to reduce unnecessary overhead.
The I/O nodes are Intel’s 386-based processor boards through which the compute nodes have access to the Concurrent File System (CFS) and an Ethernet network. I/O nodes do not participate in numerical work of computation. They communicate with compute nodes over the node-to-node message passing network and with the disk drives over a standard SCSI interface. The CFS is an optional feature of the iPSC/860 system that gives node programs high-speed access to disk storage. It is characterized by large storage capacity (individual files can be up to several Gbytes long and are distributed among the disks of the CFS in 4 kbyte blocks in a round-robin fashion by volume number) and is optimized for simultaneous access by multiple nodes. The CFS consists of one or more disk drives, each of which has an unformatted capacity of 760 Mbytes or 1.5 Gbytes. It also supports 8mm cartridge tape drives, each of which can hold up to 2.2 Gbytes. The CFS is controlled by one or more I/O nodes.

5.4.3 The HP 9000/735 cluster

We used a cluster of four HP workstations. Each workstation is equipped with 144 Mbyte memory, a 400 Mbyte system disk, a 1 Gbyte SCSI disk and a 2 Gbyte fast wide SCSI disk for storage of user’s data. The CPU of the HP workstations are HP 9000 model 735. The HP735 machines use 99 MHz Precision Architecture RISC (PA-RISC) 7100 microprocessor. Their operating system is HP-UX 9.0.

The backbone network of HKUST campus is a collection of high-speed FDDI rings, each running at 100 Mbps. The FDDI rings are interconnected by a gigaswitch, giving an aggregate bandwidth of 3.6 Gbps. The workstations are connected to the backbone via a 10 Mbps Ethernet subnet.

5.4.4 The Sun IPX/Sparc Workstations

We have used a cluster of various Sun workstations, ranging from IPX to Sparcstation 10. In particular, we have used four Sparcstation IPX, four Sparcstation 2 and eight Sparcstation 10. The cluster, therefore, consists of workstations of different hardware capabilities. All these workstations are connected via a 10 Mbps Ethernet subnet to the HKUST campus backbone network.

5.5 The Express Parallel Environment

Express from Parasoft Corporation [10] is a programming environment for parallel processing on message-passing SIMD multiprocessors or network of workstations. Programs can be written using SPMD (Single Program Multiple Data) or pure SIMD paradigms. Express provides a communication system for communicating processes, mechanisms for data sharing, reading files, debugging tools, and performance analyzing tools. An important feature of Express is that these functionalities are carried out in a user-transparent manner. The communication utilities include blocking and non-blocking communication among nodes, exchange, broadcast and collective communication such as reading and writing a vector. The global communication includes concatenation, global reduction operations, synchronization, etc.

Express has been implemented on a variety of machines including Alliant, BBN Butterfly, nCUBE, Symult, Intel iPSC/2 and iPSC/860 hypercubes, Intel Paragon, IBM370/3090, Thinking Machine’s CM-5, KSR-1, and transputer arrays. The network version of Express allows a network of workstations to be used as a “virtual parallel machine”. For our current implementation, choice of this particular programming environment is indeed germane.

6 Performance Results

Experiments were performed on the Intel Paragon, Intel iPSC/860, and SUN and HP workstations, using various number of processors. We used five video sequences: Football (360 × 240), Table Tennis (360 × 240), Salesman (360 × 288), Miss America (352 × 288) and Swing (352 × 288). The Football sequence involves a football game, where players move in fast motion. The Table Tennis sequence is a bouncing pingpong ball with two players playing the ball; it involves camera panning and zooming and a scene change. The Salesman sequence shows a person holding an object while talking; the salesmen moves his hands and the object rapidly. The Miss America sequence is basically a head and shoulder sequence, and the motion involves a very large area of the frame. The Swing sequence consists of a cluster of different charts and graphs, which also involves some movements. All of these sequences are representative of different kinds of motion and are very useful for testing motion estimation.

The measured time was averaged over 50 frames of a video sequence. The time to process 50 frames of a video sequence was not necessarily the same in each processor, so the average was also taken over all the processors. Several such sets of measurements were taken using various number of processors for each set. All of the timings were measured with microsecond granularity.

We used a constant bit-rate of 5 Mbp and a video-buffering-verifier buffer size of 112 as input for all 50 frames, with a GOP of 12 and a I-to-P frame distance of 3, while the search window was ±11 pels for P-pictures and ±10 pels for B-pictures. Both full-search and 2D-logarithmic search were performed and the corresponding performance was monitored. In order to measure the quality of the video, we used the peak signal-to-noise ratio (PSNR) as there exists no good and simple metric for this measure [4]. The PSNR of a video is defined as follows:

\[
PSNR = 10\log_{10} \frac{255^2}{MSE}
\]

where MSE is the Mean Square Error. The larger the PSNR, the better the quality. The average values of PSNR obtained for different sequences using our encoder are
shown in Table 1.

<table>
<thead>
<tr>
<th>Name of Sequence</th>
<th>Average PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Football</td>
<td>34.3281</td>
</tr>
<tr>
<td>Table Tennis</td>
<td>35.7497</td>
</tr>
<tr>
<td>Salesman</td>
<td>34.5039</td>
</tr>
<tr>
<td>Miss America</td>
<td>37.1647</td>
</tr>
<tr>
<td>Swing</td>
<td>37.3657</td>
</tr>
</tbody>
</table>

For each hardware platform, we present three results: processing speedup versus variable number of processors; comparison of the timings of various MPEG-2 modules; and encoding rate versus number of processors. These results are described below.

6.1 Results on the Intel Paragon

Figure 15 gives the overall speedup with various number of processors on the Intel Paragon. It is noticeable that there was a rapid increase in the speedup up to 128 processors. By increasing the number of processors from 128 to 256, an escalation in speedup was still observed. However, a little improvement was obtained beyond 256 processors. The reason is that the times for all the modules (DCT, motion estimation, etc) except the "others" module (described next) approached their minimum limits.

Figure 7 provides a comparison among the times for various modules in terms of the time required by each module. This bar graph indicates where the bottleneck of the encoding is. In this case, it be noticed that motion estimation took 18.1% of the total encoding time which is more than any other module except "others". The "others" module, included various housekeeping functions and some constant overhead. This module did not exhibit very regular behavior and, in fact, gave a speedup less than 1, implying the extra overhead due to parallelization.

Figure 11 shows the encoding rate for various number of processors. As can be seen from this figure, a encoding rate higher than or close to real-time (30 frames/sec) was achieved using 330 processors. The maximum encoding rate was achieved for the Swing sequence (31.14 frames/sec). The peak rate, averaged over all the sequences, was 29.36 frames/sec.

6.2 Results on the iPSC/860

Figure 15 show the overall speedup on the Intel iPSC/860 hypercube parallel computer. The number of processors was varied from 1 to 64, the maximum available on that machine. These curves are almost linearly increasing and thus leave the promise that introduction of more processors will yield increased performance.

Figure 7 shows the comparison among various MPEG-2 modules. The "others" module again is the dominant factor followed by motion estimation.

Figure 11 shows the encoding rate which is
considerably slower than that on the Intel Paragon. This is mainly due to the fact that the Intel Paragon is a superior machine in terms of both the computation and communication speed. The maximum encoding rate achieved was 4.15 frames/sec.

6.3 Results on the Sun Workstations

The experiments on the network of SUN workstations connected through the Ethernet were conducted using 1 to 16 workstations. Figure 15 shows the overall speedup. As can be seen from this figure, the speedup was still increasing. While the workstation-based clusters do not scale very well as compared to the massively parallel processing systems, we expect further improvement in speedup with more workstations.

Figure 7 shows the relative time of various modules. The dominating factor now is motion estimation, not “others”. This is because the inherent parallelism in motion estimation using 16 workstations was not fully exploited and therefore its time was still very high.

The encoding rate is shown in Figure 11. The maximum encoding rate achieved was 1.91 frames/sec. While this rate is not a real-time rate, as mentioned earlier, this kind of environment can be useful for non real-time applications, such as encoding of large archival data.

6.4 Results on the HP Workstations

The speedup on the 4 HP cluster is shown in Figure 15. The speedup again is almost linear. While the number of workstations in this case was only 4, these are faster workstations compared to the SUNs. Furthermore, the network is an FDDI network which is much faster than the Ethernet. We therefore expect a more linear speedup with increasing number of HP workstations as compared to the SUNs.

Figure 7 shows the relative timings for various modules. This figure indicates the time for motion estimation as well as DCT and Quantization becoming more dominant.
The encoding rate for variable number of processors is shown in Figure 11. The maximum encoding rate achieved was 0.64 frames/sec.

Table 2 summarizes the comparison among various parallel platforms for our implementation of the MPEG-2 video encoder.

<table>
<thead>
<tr>
<th>Parallel Platform</th>
<th>Number of Processors</th>
<th>Encoding Rate (frames/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paragon</td>
<td>330</td>
<td>31.14</td>
</tr>
<tr>
<td>iPSC/860</td>
<td>64</td>
<td>4.15</td>
</tr>
<tr>
<td>Sun</td>
<td>16</td>
<td>1.91</td>
</tr>
<tr>
<td>HP</td>
<td>4</td>
<td>0.64</td>
</tr>
</tbody>
</table>

The encoding rate for variable number of processors is shown in Figure 11. The maximum encoding rate achieved was 0.64 frames/sec.

Table 2: Comparison among various parallel platforms

Figure 13: Comparison of modules using 16 SUN workstations.

Figure 14: Encoding rate on the SUN workstations.

Figure 15: The overall speedup on the HP workstations.

Figure 16: Comparison of modules using 4 HP workstations.

Figure 17: Encoding rate on the HP workstations.
7 Conclusions

In this paper a portable and scalable implementation of MPEG-2 video encoder using parallel processing has been described. The data distribution strategy has been discussed. Results on various parallel and distributed platforms have been provided. Exploiting the maximum parallelism with 330 processors on the Intel Paragon XP/S, an encoding rate higher than real-time has been achieved. With only 64 processors of the Intel iPSC/860, we have achieved an average frame encoding rate of 4.15 frames/sec. As reflected from the results, inclusion of more processors will provide increased encoding rate, enabling real-time MPEG-2 video encoding [2]. This indicates the scalability of the implementation.

This implementation is unlike any other previous work and is capable of dealing with even a single frame of input data. This feature shows the effectiveness of this implementation for real-world applications. Current efforts, in addition to investigating various code optimization strategies, are directed towards testing diverse motion estimation algorithms and using other message-passing libraries such as PVM.

References