Architecture for Distributed Video Encoding on the Internet
インタネット上での分散ビデオ符号化アーキテクチャー

Abhay Ghatpande ガトパンデ　アバイ
Hidenori Nakazato 中里　秀則
Hiroshi Watanabe 渡辺　裕
Graduate School of Global Information and Telecommunication Studies, Waseda University
早稲田大学大学院　国際情報通信研究科

Abstract
Distributed video encoding can utilize the high degree of parallelism that exists within the encoder operations similar to parallel implementations of embarrassingly parallel problems. However, the data partitioning, allocation, and scheduling schemes used there cannot be directly applied to video encoding. The primary cause is the Spatio-temporal data dependency between different processors and need for synchronization between the various tasks. There is also a lack of the lowest common unit of computation on which to base the performance model of the system. We propose some parameters and their use for a new performance prediction model.

1. Introduction

The logical architecture of most distributed video encoding systems is the master-slave approach of figure 1.1. This architecture is one of the simplest approaches toward the distribution of computation tasks among a set of distributed computing resources. A single master process controls both the distribution of encoding tasks to slaves, and manages collection of returned results and subsequent processing that may be necessary to form a syntactically correct bitstream.

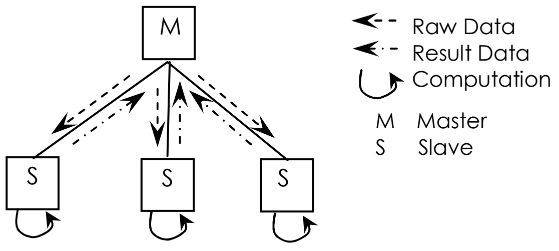

Figure 1.1 Logical Architecture

There have been several attempts to date at encoding video on clusters of workstations, notably [1][2][3][4][5][6]and several others. Almost all of them are based on the above architecture. The major differences lie in the granularity of parallelism. The general approach has been to divide the entire computation into a series of equal sized tasks and then allocate them to the different processors without much consideration for the performance heterogeneity inherent in such systems. Scheduling algorithms along with accurate predictive performance modeling have not been used to improve the performance of distributed systems.

In Section 2, we explain the traditional approaches in management of the task queue and scheduling and discuss why this approach is insufficient in the case of video encoding. In Section 3, proposed amendments are presented. Section 4 concludes with the summary and future work.

2. Traditional Approach and Shortcomings

The granularity of partitioning has varied from macroblock (MB) level [3] to GOP based schemes where each processor is assigned an entire GOP for encoding [6]. These schemes did not take into account the different computational capacity, present processor workload, and network conditions of the individual slaves (and the master), and the results have been sub-optimal.

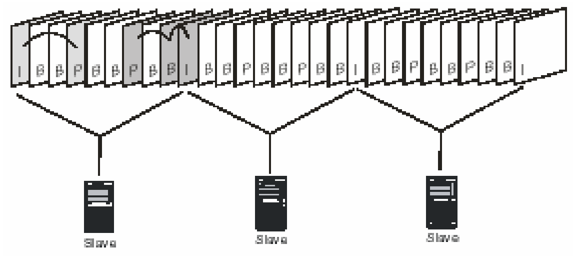

Figure 2.1 GOP Partitioning, from:[6]

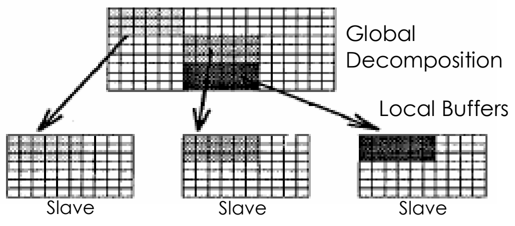

Figure 2.2 Frame Decomposition, from: [3]

To be able to utilize the limited resources available with maximum efficiency, the application requirements and resource capabilities should be thoroughly understood and modeled accordingly.

A simple comparative estimate of the processor capability can be taken by assigning an equal amount of work to the individual processors and measuring the total time taken by each to complete it. The work is then partitioned among them in proportion to their capability so that the time taken by each to perform the work is equal [6]. The work allocated to processor i is:

$$w(i) = W * \frac{t(i)}{\sum_{i=1}^{P} t(i)} \tag{2.1}$$

Where, $w(i)$ is the work allocated to processor i, W is the total units of work available, $t(i)$ is the processing time for unit work on processor i, and P is the total number of processors

This offers improved performance as compared to systems that use equal partition size. The problem with this approach is that it is a *reactive* system. There is no *prediction* of the system performance upon which to base the scheduling. The points in time where the performance is measured, the allocation is decided, and the actual work is done are all different. The dynamic nature of the resources means that the state of the system differs at these points and the performance is different from expected to a large degree.

Another point where this model breaks down is when the communication time is much longer compared to the computation time since the parallel operation gets reduced to simple serial processing [8]. In figure 2.3, T1, T2, T3 are the communication times, while C1, C2, C3 are the computation times per WU for processors 1, 2, 3.

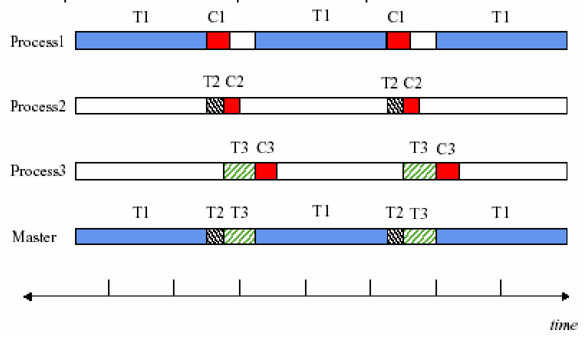

Figure 2.3 Process Serialization, from: [8]

Since the time T1 is very long, only a single round of results can be processed before the second round results are ready to be received. The total execution time becomes

$$T_{serial} = \frac{P}{\sum_{i=1}^{P} T(i)} \tag{2.2}$$

3. Proposed Amendments

To predict application behavior we need at least to:

(a) determine the granularity i.e. basic Work Unit (WU),
(b) measure slave capacity for processing each WU,
(c) find the amount of processing per WU necessary on Master,
(d) know the Network link capacity and condition, and
(e) monitor the present processor workload

We consider the video macroblock (MB) as the work unit (WU) on which all operations are carried out. Since the operation a MB undergoes depends on the type of frame being processed (I/P/B), we propose the creation of three basic work units instead of one – the IWU, PWU, and BWU. For example, a P frame may contain a few I MBs, but for the sake of calculation, it will be considered as a PWU. A test sequence is run on each processor and the time taken to encode individual frames is measured. We calculate the average time taken to encode each type of WU. The capacity of processor P_i to process IWUs, PC_{IWU} when unloaded in terms of IWU/s:

$$PC_{IWU}(P_i) = \frac{N * IF}{\sum_{n=1}^{IF} T_{ENC}(n)} \tag{3.1}$$

Where N is the number of IWUs per I frame, IF is the total number of I frames in the test sequence, and $IT_{ENC}(n)$ is the encoding time per I frame. For example, we can say that slave P_i can process x_i IWU/s, y_i PWU/s, and z_i BWU/s when unloaded. With the help of NWS [9] the present CPU availability A_i can be calculated as a number between 0 and 1. The number of IWUs to be assigned to P_i then is:

$$IWU(P_i) = N * \frac{A_i * x_i}{\sum_{i=1}^{P} A_i * x_i} \tag{3.2}$$

Similar values are calculated for PWUs and BWUs. The average of the three values is taken and rounded to nearest integer. This value is used as the reference WU to be allocated to P_i. After calculating WU values for all the processors, adjustments are made to ensure that the sum does not exceed or fall short of N.

With network load prediction system such as NWS [9], the network link capacity is calculated. Currently we ignore the link latency. With the bandwidth B_i bytes/second for processor P_i and S bytes/WU, we calculate the effective capacity of the network link $Net(P_i)$ in WU/s.

$$Net(P_i) = B_i / S \tag{3.3}$$

We observe that this value is the limiting factor in assigning the WUs to the respective processor. For example, the processor may be capable of processing 200 IWU/s but the network may allow only 130 WU/s to be sent to the processor.

Up to now we have just considered the transmission of raw video data to the processors for encoding. However, the reception of encoded data from the processors is equally important and also consumes large amount of the network bandwidth. Since the compression ratio per WU is variable, it is difficult to quantify the output result data per WU. However, we feel that the effect of "upload" is lower than "download" because the amount of information to be moved is smaller. We are working on how to factor in this parameter into the system.

Similarly, estimation of the processing to be done by the master also needs to be investigated in detail so that the complete performance model can be established.

4. Conclusion

We analyzed the various problems in adapting the traditional partitioning and scheduling approaches for video encoding on distributed systems. We proposed the amendments necessary to these models for video encoding. Our future work will involve detailed simulation testing and implementation for optimal scheduling of distributed video encoding on the internet.

References

[1] Ahmad I, *et al*, "A Scalable Off-line MPEG-2 Video Encoding Scheme Using a Multiprocessor System", Parallel Computing, Elsevier Pub.,Vol 27, Issue 6, May 2001, pp 823-846.

[2] Shen K. and Delp E., "A Spatio-Temporal Parallel Approach for Real-Time MPEG Video Compression", Proc. of ICPP 1996, Vol. 2, Aug 1996.

[3] Akramullah S, *et al*, "Parallelization of MPEG-2 Video Encoder for Parallel and Distributed Computing Systems", Proc. of the 38th Midwest Symposium on Circuits and Systems, Vol. 2, Aug 1995.

[4] Bozoki S, *et al*, "Parallel Algorithms for MPEG Video Compression with PVM", Proc. of Eurosim 1996, Jun 1996.

[5] Farin D, *et al*, "SAMPEG,a Scene Adaptive Parallel MPEG-2 Software Encoder", Proc. of SPIE VCIP, Jan 2001.

[6] Ribeiro M, *et al*, "MPEG-4 natural Video Parallel Implementation on a Cluster", Proc. of 12th Portuguese Conference on Pattern Recognition, Jun 2002.

[7] Berman F, *et al*, "Application-Level Scheduling on Distributed Heterogeneous Networks", Proc. of Supercomputing' 96, Nov 1996.

[8] Shao G, "Adaptive Scheduling of Master/Worker Applications on Distributed Computational Resources", Ph.D. thesis, University of California, San Diego, 2001.

[9] Wolski R,, *et al*, "The Network Weather Service: A distributed resource performance forecasting service for metacomputing", Future Generation Computer Systems, Vol. 15 (5-6), Oct 1999

〒169-0051 Tokyo, Shinjuku-ku, Nishi Waseda 3-1-10, 29-7 Bldg
abhay@toki.waseda.jp, {nakazato, hiroshi}@giti.waseda.ac.jp