An Optimal Scheduling Algorithm for Parallel Video Processing

D Turgay ALTILAR
Queen Mary and Westfield College
Department of Computer Science
Mile End Road E1 4NS London -UK
turgay@dcs.qmw.ac.uk

Yakup PAKER
Queen Mary and Westfield College
Department of Computer Science
Mile End Road E1 4NS London -UK
paker@dcs.qmw.ac.uk

Abstract

We present an optimal scheduling algorithm called ?eriodic Write-Read-Compute (PWRC) scheduling for ıarallel video processing. PWRC scheduling exploits ·ontinuity and periodicity of the video data. Therefore, it is uitable for any type of periodic data over which data ndependent application is to run. The target architecture s a client-server based system having a point-to-point :ommunication between the host any worker processors vhere SPMD type of programming is assumed. PWRC ·equires a high level atomic write-read command for data ransmission. The analysis of the cost model provides nformation either to form a parallel video processing ·nvironment or to predict the overall performance of an :xisting system. Meeting real-time requirements of video ›rocessing under PWRC scheduling is discussed as well.

l. Introduction

A parallel video processing scheduling system can be :onsidered as a real-time processing system with periodic lata input. Input data for such a real-time system consists ›f a number of video sequences that naturally possess :ontinuity and periodicity features. By making use of these 'eatures predictable and periodic scheduling schemes can ›e defined for data independent computation. The ›erformance of a scheduling scheme relies upon both the ;ystem architecture and the application. Architectural ›roperties such as I/O bandwidth, processor power, and lata transmission rate, and properties of the algorithm such ıs data partitioning (overlapped or non-overlapped), the ıeed of consecutive frames to be processed form the basis o define relations among number of processors, required /O time, and processing time. In this paper, an optimal ;cheduling scheme for parallel video processing system is lefined by taking the utilisation of both I/O channels and ›rocessors into consideration. Although it is stated that the ;oal of high performance computing is to minimise the ·esponse time rather than utilising processors or increasing systems throughput [1], we concentrated on utilisation as well as the response time. In the literature, there are a number of cost models [1][2][3]. The analysis of the new scheme proposed in this paper has been inspired by a recent paper of Lee and Hamdi [3] since it comprises generic units and definitions for a cost model. Having the same cost model parameters will also provide us a common platform to compare our algorithm with theirs. Since a scheduling scheme could be defined with some system architecture and algorithm variables, one could define scheduling algorithm prior to the establishment of the parallel processing environment and build the parallel processing machine up by making use of this information. By thinking the other way around, it is also possible to predict the overall performance of a given system.

2. Scheduling for client-server based systems

Parallel video processing has two facets: processing and I/O. Since our concern is to define a scheduling algorithm to utilise both I/O and processing power of a system, the characteristics of an application are important as well as the hardware/software characteristics. We assume that there is a parallel processing machine attached to a data storage such as frame buffer or video disk, running with a client-server model. There is a host processor responsible for reading data from a video storage, partitioning and distributing the frame, collecting and re-composing the output frame, and writing back to video storage. There are n worker processors running in SPMD (Single Program Multiple Data) model. There is no communication among worker processors. Workers have point-to-point host-connection, which can be defined in terms of data transfer rate, latency and packet size. Processor power is defined as computation time per pixel.

The parallel video processor is assumed to be equipped with a high level atomic write-read command which could be easily realised within high level software for a client-server based architecture. A host computer ensures that a data write operation request from a worker processor is immediately followed by a data read operation.

)-8186-8557-3/98 $10.00 © 1998 IEEE

A number of video processing algorithms require two frames of different sizes to be computed. Applications like motion estimation [4],[5], edge detection [6], require surrounding pixels in addition to the base frame. In this paper we are interested in algorithms with both of the above properties. The host processor is responsible for producing overlapped partitions from the raw video data.

2.1. A Scheduling Scheme

By defining a scheduling algorithm we are aiming at producing a number of relationships among system architecture and application parameters.

With the above given architectural requirements, Lee and Hamdi presented a performance prediction model for parallel image processing system running convolution in [3]. They produced two equations in this paper. The number of workstations (n) to achieve minimum execution time for convolution type algorithms is:

$$n = \frac{M^2\gamma}{\alpha / K + \beta} + 1 \qquad (1)$$

and the maximum possible speed-up for parallel execution of image processing algorithm is approximated by:

$$S = \frac{M^2\gamma}{2(\alpha / K + \beta)} \qquad (2)$$

where M is the width of coefficient matrix, γ is computation time per pixel, α is the latency time, β is the data transmission time and K is the packet size to be transmitted through the transmission medium.

They assumed that this program runs over a network of workstations in a "Host-Node" (client-server) manner. The parallel execution time is broken up into four terms; T_a: the host setup time, T_b: the communication time for sending all sub-images (matrices) and coefficient matrix, T_c: the average computation time on a single workstation, T_d: the communication time for receiving the partial results. Thus, $T_{(n)}= T_a+T_b+T_c+T_d$ and if the computation time is too small with respect to communication time $T_{(n)}'= T_a+T_b+n*T_d$. Assuming that $T_{(n)}'$ is the lower bound execution time for parallel processing, they derived further equations using $T_{(n)}'$ instead of $T_{(n)}$. Communication time consists of two components, which are α, the latency time, and β, transmission time per byte. Therefore, for a given data of size P, the communication time is $\alpha+P\beta$. Since data is sent in packets, actual communication time is defined by $T_{comm}= \lceil P/K \rceil \alpha+P\beta$. As the computation is basically a matrix multiplication, the sub-image computation time is declared as $N^2M^2\gamma/n$, where N^2 is the size of image matrix, M^2 is the size of coefficient matrix and γ is the computation time per pixel.

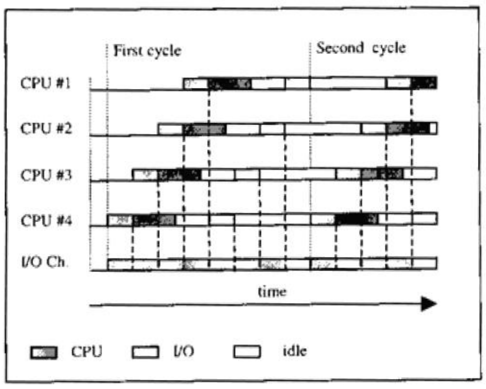

Figure 1. Processor and I/O channel usage for four processors.

Although the I/O channel is kept busy throughout the process, the utilisation of the processors is not so good as seen in Fig.1. Even if the execution time could be made equal to the I/O duration of all other processors, waiting for data between cycles introduce an idle duration T_{idle} that is the sum of all other processors' data read time.

The PWRC scheduling scheme makes use of continuity and periodicity characteristics of a video stream to increase the utilisation of processors while keeping the I/O channel fully occupied so that the overall application would run with less number of processors than the value computed by Eq.1.

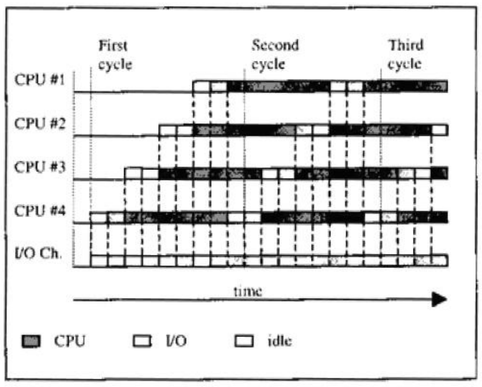

Figure 2. The new scheduling scheme providing theoretically maximum processing unit and I/O channel utilisation.

To keep processors as busy as possible, they should be served with new data just after receiving processed data, which we call indivisible write-read mechanism. This ensures that processors will be busy while a processor is using the I/O channel. A timing diagram for 4 processors running under the proposed scheduling scheme is sketched in Fig.2. It is seen that I/O channel is fully utilised.

A process running on a CPU takes its turn by writing output which is immediately followed by a data read which is then proceeded by a computation stage. Exceptionally, in the first cycle, single void write sessions take place for the sake of sycnchronisation. Given the timing diagram, one can derive the relationship among computing time, data read time, data write time, and the number of the processors. Taking the second cycle in Fig.2. as an example, it is seen that the computation time for the 4th processor corresponds to the sum of other three processor's I/O time. In general, compute time, T_c can be defined as:

$$T_c = (n - 1)\,(T_{dr} + T_{dw}) \qquad (3)$$

where T_{dr} is data read and T_{dw} is data write time.

We assume that the application requires two separate data blocks: a base frame of size NxN and a sub-frame of size MxM pixels. Due to the algorithm, we also assume that, the base frame is accompanied with a surrounding frame of width M. Transmission of surrounding pixels produces a overhead of O_d. The algorithm consists of data independent operations, thus compute time is propotional with M^2N^2. For a parallel processing system comprising n CPUs, T_c, T_{dr}, and T_{dw} can be given as:

$$T_c = \frac{M^2N^2}{n}\gamma \qquad T_{dw} = \left\lceil \frac{N^2}{nK} \right\rceil \alpha + \frac{N^2}{n}\beta$$

$$T_{dr} = \left\lceil \frac{M^2}{K} \right\rceil \alpha + M^2\beta + \left\lceil \frac{N^2 + O_d}{nK} \right\rceil \alpha + \frac{N^2 + O_d}{n}\beta$$

where K is the data packet size, α is the latency time, β is data transmission time per byte, γ is the processing time per pixel, N^2 is the size of the frame in pixels, O_d total data overhead due to boundary conditions.

Substituting the given three equations for T_c, T_{dr}, and T_{dw} in Eq.3 a second degree equation of n is derived. The two roots of the function is given as follows:

$$n_{1,2} = -\left(\frac{\left(\left(\frac{2N^2+O_d}{K}-1\right)\alpha + \left(2N^2+O_d-M^2\right)\beta\right)}{2(\alpha+M^2\beta)}\right) \pm \frac{\sqrt{\left(\frac{\left(\left(\frac{2N^2+O_d}{K}-1\right)\alpha + \left(2N^2+O_d-M^2\right)\beta\right)}{(\alpha+M^2\beta)}\right)^2 + 4\left(\frac{\left(\left(2N^2+O_d\right)\left(\frac{\alpha}{K}+\beta\right)+M^2N^2\gamma\right)}{(\alpha+M^2\beta)}\right)}}{2}$$

Two roots are functions of O_d, K, N, M, α, β,and γ. The positive valued root gives the value of n, the number of processor required. Although the value of O_d depends on the number of partitions, i.e. n, an iterative computation for n and O_d beginning with upper bound of O_d yields a solution. The value of O_d is at most 10-20% of the actual data size. Computation of the overhead, O_d, and its minimisation is investigated in a further paper.

A numerical example is given as follows to compare the performance of the PWRC algorithm with the one proposed in [3]. For a group of typical values of a convolution process; N=1024 pixel (of bytes, i.e., grey-level), M=11 pixel, K=1024byte, α=2ms, β=2µs, and γ=2µs, 63 processor (partition) is required for best performance according to Lee and Hamdi's equation. However, it dramatically falls to 32 processors (partition) for our scheduling algorithm.

If the coefficient matrix is sent once every factor related to the coefficient matrix size can be ignored since it happens once at the beginning of the application, it would take negligible time considering the overall processing time. Therefore, there would not be any contribution of data transfer time due to M. Assuming that the upper values are close to the actual values, one can derive the following through the same substitutions:

$$n = (M^2\,\gamma / 2(\alpha/K+\beta)) + 1 \qquad (4)$$

Constants reflecting system architecture specifications could be defined as "Coefficient of Architecture", i.e., C_A.

$$n = (M^2\,C_A / 2) + 1 \qquad (5)$$

As T_s is processing time of the computation for a single processor and T_n is processing time for a parallel system comprising n processors, the speed-up S_n can be computed as follows:

$$S_n = T_s / T_n = T_c / (n\,(T_{dr} + T_{dw})) = M^2\,C_A / 2 \qquad (6)$$

4. Deciding System Architecture Parameters

For real-time video processing, cycle time, which should be less than 40ms, is the constraint which must be met. Considering Fig.2, a way of defining cycle time is:

$$T_{cycle} = T_{dr} + T_c + T_{dw} \qquad (7)$$

$$T_{cycle} = \frac{1}{n}\left(\left(2N^2 + O_d\right)\left(\frac{\alpha}{K} + \beta\right) + M^2N^2\gamma\right) \qquad (8)$$

Since Eq.8 includes both system parameters and application parameters, a system architecture can be defined for a defined application in the light of this equation. As cycle time is a standard in video processing environment, we can derive Eq.8 for n. Thus;

$$n = N^2\,(2(\alpha/K+\beta)+\gamma M^2)/T_{cycle} \qquad (9)$$

$$n = N^2\,(C_c+\gamma M^2)/T_{cycle} \qquad (10)$$

where C_c is the communication constant of the system.

Considering the Eq.10, there is a linear relationship between system constants and application constants. However, processing constant has a greater impact on the number of processors than system communication constant as it has M^2 as a multiplier.

If a distributed system, with typical values of N=1024 pixel (of bytes, i.e., grey-level), M=11 pixel, K=1024byte, α=2ms, β=2μs, and γ=2μs, is running to achieve computation performance of T_{cycle} = 10 seconds, 25 worker processors would be sufficient. For a distributed video processing system with 32 worker processors, above given application would run over a single frame in 8.1875 seconds, i.e. T_{cycle} = 8.1875 seconds.

This result does not satisfy real-time processing constraint of 40ms per frame for PAL. Even an increase in the number of processor would not allow his system to run in real time. The bandwidth of the system to transfer data is not sufficient for the given typical characteristic values. The time elapsed to transmit (read and write) the frame by neglecting the overhead is defined as:

$$T_{transmission} = 2N^2(\,(\alpha/K)+\beta\,)$$

As $T_{transmission}$ is 8 seconds for the values given above, real-time processing requirements can not be fulfilled by this particular system. However, a parallel system connected via a high-speed bus or a dedicated network can meet real-time processing constraint of $T_{transmission}$< 40 ms.

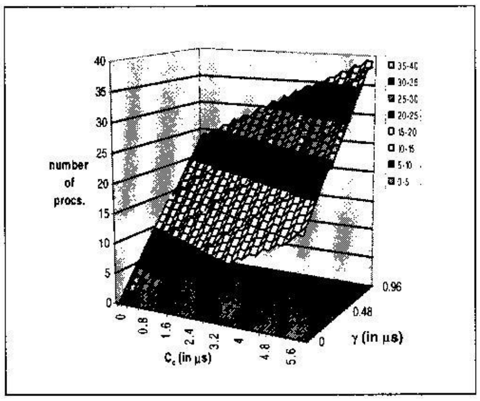

Fig 3. The number of processors wrt C_c and γ for M^2=9, N^2 =288x360

To design a system, utilising both I/O channel and processors, and running in real-time, the number of processors can be computed by Eq.10 for the given system and application parameters with $T_{transmission}$< 40 ms.

The number of processors required for real time video processing, i.e., achieving a processing rate of 25 frames/second, with respect to the system parameters (C_c and γ) for M^2=9, N^2 =288x360 is given in Fig.3

5. Conclusion and Further Research

In this paper, we are proposing an optimal scheduling algorithm called Periodic Write-Read-Compute (PWRC) scheduling algorithm, for real-time video processing. The basic processing algorithm requires two sub-frames of different sizes. The generic system architecture is based on client-server model running SPMD type of programs. A high level atomic write-read command is the only ad hoc requirement to realise the PWRC scheduling algorithm. It is proven that the PWRC algorithm takes the same time to process video sequence with the half of the number of processor required in Lee's algorithm. A number of relations among system characteristics and application characteristics have been provided by the further analysis of PWRC scheduling algorithm. It is shown that either a parallel video processing system can be built for a given type of application or the performance of an established parallel processing system can be examined for applications with different characteristics by the use of these relations. The algorithm is currently being extended to cover other types of applications encountered mainly in video encoding and decoding.

6. References

[1]. Crandall P. E., Quinn M. J., A Partitioning Advisory System for Networked Data-parallel Processing, Concurrency: Practice and Experience, Vol.7(5),479-495,August 1995.

[2] Weissman J.B., Grimshaw A. S., A Framework for Partitioning Parallel Computations in Heterogeneous Environments, Concurrency: Practice and Experience, Vol.7(5),455-478,August 1995.

[3] Lee C., Hamdi M., Parallel Image Processing Applications on a Network of Workstations, Parallel Computing, 21 (1995), 137-160.

[4] Weiping L., Chapter 3.2: Motion and Texture Coding, Circuits and Systems in the Information Age.

[5] ISO/IEC JTC1/SC29/WG11, MPEG 4 Video VM Version 7.0, MPEG97/N1642, Bristol, April 1997.

[6] Weems C.C., Brown C., Webb J.A., Poggio T., Kender J.R., Parallel Processing in the DARPA Strategic Computing Vision Program, IEEE Expert, 23-38, October 1991.