# An Optimal Scheduling Algorithm for Stream Based Parallel Video Processing

D. Turgay Altılar[1] and Yakup Paker[2]

[1] Dept. of Computer Engineering, İstanbul Technical University
Ayazağa Campus, Maslak, 34457, İstanbul Turkey
altilar@itu.edu.tr

[2] Dept. of Computer Science, Queen Mary, University of London
Mile End Road, E1 4NS, London, United Kingdom
paker@dcs.qmul.ac.uk

**Abstract.** We present a new optimal scheduling algorithm called Periodic Write-Read-Compute (PWRC) scheduling for stream based parallel video processing. Although PWRC scheduling exploits the properties of the video data, it is applicable to any type of periodic data over which a data independent application is to run. The PWRC algorithm is designed considering a bus based parallel architecture allowing point-to-point communication between host and workers. The PWRC requires a high level atomic write-read command for data transmission which can be created in various ways. The analysis of the PWRC provides information either to form a parallel video processing system or to predict the overall performance of an existing system in order to meet real-time requirements of video processing.

## 1 Introduction

A parallel video processing system can be thought as a real-time processing system with periodic data input. We believe that scheduling for such a system should exploit input-output characteristics in order to cope with real-time requirements. Input data for such a real-time system naturally possess continuity and periodicity features. In this paper, we dealt with data independent computations over video streams, i.e., computation time is proportional with the data size. Continuity and periodicity of input and output decoupled with data independency of the application provide us with a base to define an optimal scheduling algorithm.

The performance of a scheduling algorithm relies upon both the architectural properties of the system such as I/O bandwidth, processor power, memory size, and the properties of the application such as data dependency, data partitioning. For example the need of consecutive frames makes a great difference in the design of scheduling algorithms for video processing. In this paper, we dealt with only stream based video processing algoritms. Initial results of such algorthms is discussed in [3]. Scheduling for frame by frame processing were given in another paper[2]. However, our approach is the same for both: utilising both I/O channels and processors while minimising the response time.

Given a number of cost models [1],[4],[5],[6],[7], and [8] the analysis of the proposed schemes has been inspired from a recent paper of Lee and Hamdi [6] comprising generic units and definitions for a cost model. Having the same cost model parameters will also provide us with a common base to compare our algorithm with theirs.

In this paper, we will show that for a given system and algorithm, a scheduling method can be defined and the overall system performance can be predicted. Since a scheduling scheme could be defined according to system architecture and algorithm variables, parallel processing architecture variables such as processor power and bus rate can also be calculated, for a given algorithm and a given scheduling method.

The proposed scheduling algorithm relies upon an indivisible write-and-read command to access continuous data storage medium. The considered parallel processing environment has a write-and-read command, which can be implemented as a high level atomic command. The algorithms mentioned in this paper were designed, developed and analysed for a bus based parallel system having a client-server model running Single Program Multiple Data (SPMD) type programs. The target architecture is a client-server based parallel system having a point-to-point communication between the server and client processors. A typical hardware configuration comprises a server processor, a frame buffer and a number of client processors connected via a high speed I/O bus and a signal bus. Data transfer occurs over the high speed I/O bus to and from the frame buffer. The frame buffer is the medium that a video stream is written via an input device such as video player or camera. Processed data is also written to the frame buffer. No communication or data transfer exists between client processors. A client is allowed to read/write data from/to the frame buffer under the control of the server.

The rest of the paper is organised as follows: Section 2 introduces the cost model with reference to Lee and Hamdi [6]. PWRC Scheduling is defined, discussed, analysed and performance comparisons are given in Section 3. Section 4 explains the use of the PWRC to decide system parameters of a parallel system to be build. Paper ends with conclusions and further research.

## 2 A Scheduling Scheme

For a loosely coupled, client-server type programming environment, Lee and Hamdi presented a performance prediction model for a parallel image processing system running convolution in a recent paper [6]. The application they considered is an image convolution program running over a network of workstations in a "Host-Node" (client-server) manner. The number of workstations, $n$, in order to achieve the minimum execution time and the approximate value of maximum speed-up, $S$, for parallel execution are given as follows:

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

where $M$ is the width of coefficient matrix to be used for convolution, $\gamma$ is computing time per pixel, $\alpha$ is latency time, $\beta$ is data transmission time per byte and $K$ is data packet size.

For a given data of size of P bytes, the communication time is is defined by $T_{comm} = \lceil P/K \rceil \alpha + P\beta$. The sub-image computation time is declared as $N^2 M^2 \gamma / n$, where $N^2$ is the size of the image matrix.

Input/output time is longer than the processing time per processor. Therefore, processors are likely to queue up for input/output which yield long idle durations for processors. The scheme aims at the full utilisation of the I/O channel without paying much attention to the utilisation of CPUs (Fig.1a). Although the I/O channel (or host CPU dispatching the partitioned data) is being kept busy throughout the process, the processors are not highly utilised (Fig.1a). Even if execution time could be made equal to the sum of I/O durations of all of the other processors, waiting for data between cycles introduces an idle duration $T_{idle}$ for a system with $n$ client processors:

$$T_{idle} = \ (n-1) \ x \ data \ read \ time \ per \ processor$$

This duration becomes very significant with finer granularity. This scheduling scheme does not consider continuous and periodic nature of the video sequences. Processing with a frame starts after the previous one. Waiting between frames introduces idle durations for processors. The approach that we adopt is to minimise the idle duration of processors, if not to totally eliminate it.

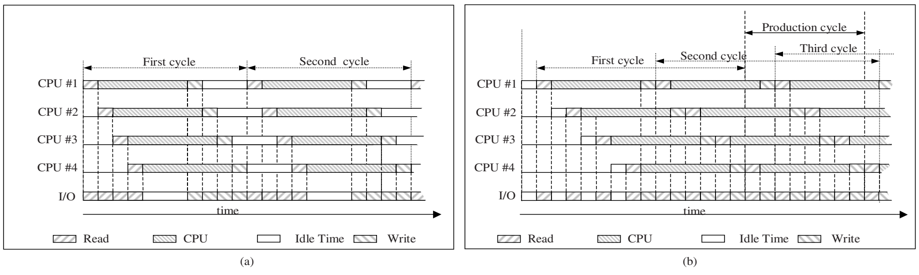

**Fig. 1.** Processor and I/O channel uses for 4 processors for a)Lee&Hamdi b)PWRC

## 3 Periodic Write-Read-Compute Scheduling

The PWRC aims to increase the utilisation of the processors while keeping the I/O channel fully occupied by exploiting the continuity and periodicity of the input data. In order to keep a processor as busy as possible in such a parallel system, a processor should receive the new data just after it sends (writes back) processed data. An indivisible write-read mechanism could be implemented even at high level programming. Since the processors are supplied with new data as soon as it writes the previous result, the I/O channel will be released and processing will start immediately. If the processing times for the processors are overlapped with the I/O channel accesses of the other processors, full utilisation of the processors and the I/O channel will be achieved. A timing diagram for 4 processors running under the proposed scheduling scheme is sketched in Fig.1b. Except for the very first cycle of the processing, which is negligible considering the whole process the I/O channel is kept fully utilised. Given the timing diagram in Fig.1b , the utilisation of processing units are at their maximum and the I/O channel is also fully utilised after the first cycle.

## 3.1 Performance of the PWRC Scheduling

We are going to consider two metrics to compare the performance of two scheduling methods: cycle time and production time. The cycle time is dominated by the total data transmission (read and write) time required by the client processors for processing systems since I/O is dominant. The production time is as the same as the cycle time for Lee and Hamdi's method (Fig.1a) However, in PWRC, production time corresponds to lifetime of a frame in parallel processing system(Fig.1b). Accepting that data transmission time per processing unit is the same for every processor, cycle time, $T_{cycle}$, and computation time, $T_c$, for $n$ processors are defined as follows:

$$T_{cycle} = n(T_{dr} + T_{dw}) \tag{2}$$

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

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

Considering the convolution application and given metrics in [6], it is assumed that the application requires two separate data blocks: a base frame of size $NxN$ and a sub-frame of size $MxM$ pixels. Transmission of surrounding pixels produce data overhead of $O_d$. Computing time is propotional to $M^2N^2$. Therefore, the processing time for each CPU, $T_c$, can be given as follows:

$$T_c = \frac{M^2N^2}{n}\,\gamma \tag{4}$$

Data read $T_{dr}$ and data write $T_{dw}$ durations are:

$$T_{dr} = \frac{M^2}{K}\,\alpha + M^2\beta + \frac{N^2+O_d}{nK}\alpha + \frac{N^2+O_d}{n}\beta \tag{5}$$

$$T_{dw} = \frac{N^2}{nK}\alpha + \frac{N^2}{n}\beta \tag{6}$$

Substituting Eq.4, Eq.5 and Eq.6 in Eq.3 and solving for $n$, we obtain

$$n^2(\alpha + M^2\beta) + n\left(\left(\frac{N^2+O_d}{K} - 1\right)\alpha + \left(2N^2 + O_d - M^2\right)\beta\right) - \left(\left(2N^2+O_d\right)\left(\frac{\alpha}{K}+\beta\right) + M^2N^2\gamma\right) = 0 \tag{7}$$

The roots of such a second degree equation can be given as follows:

$$n_{1,2} = -\left(\frac{\left(\frac{2N^2+O_d}{K}-1\right)\alpha + \left(2N^2+O_d-M^2\right)\beta}{2(\alpha+M^2\beta)}\right)$$
$$\pm\sqrt{\left(\frac{\left(\frac{2N^2+O_d}{K}-1\right)\alpha + (2N^2+O_d-M^2)\beta}{2(\alpha+M^2\beta)}\right)^2 + \frac{\left((2N^2+O_d)\left(\frac{\alpha}{K}+\beta\right)+M^2N^2\gamma\right)}{\alpha+M^2\beta}} \quad (8)$$

The positive valued of $n$ shows the required number of processors. 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 the upper bound of $O_d$ yields a solution. The value of $O_d$ is at most 10-20% of the actual data size.

A numerical example is given below to compare the performance of the PWRC algorithm with the one proposed in [6]. For a group of given typical values for a convolution process; $N = 1024 pixels$ (of bytes, i.e., grey-level), $M = 11 pixels$, $K = 1024 bytes$, $\alpha = 2ms$, $\beta = 2\mu s$, and $\gamma = 2\mu s$, 63 processors (partitions) are required for the best performance with respect to Lee and Hamdi's equation. However, we find that this figure dramatically falls to 32 processors (partitions) if PWRC is used under the same conditions.

## 4 Deciding System Hardware Architecture Parameters

On the other hand, it would be sufficient to send the coefficient matrix once in the cases of video processing processes. If the coefficient matrix is sent once, every factor related to the coefficient matrix size would be ignored. Assuming that the upper values are close to the actual values and $O_d \ll N^2$ read and data write time could be defined as follows:

$$T_{dr} = \left(\frac{N^2}{n}\right)\left(\frac{\alpha}{K}+\beta\right) \qquad and \qquad T_{dw} = \left(\frac{N^2}{n}\right)\left(\frac{\alpha}{K}+\beta\right)$$

Having the above equations, $n$ can be computed as follows:

$$n = \frac{1}{2}M^2\frac{\gamma}{(\frac{\alpha}{K}+\beta)}+1 \quad (9)$$

Eq.9 indicates the relation among the number of processors, the coefficient matrix size and the system coefficients. Surprisingly, the number of processors becomes independent of the size of the image. Eq.9 also indicates that the minimum execution times can be achieved by using half of the processors with PWRC. We defined the system architectural constants under a single name "Coefficient of Architecture", i.e., $C_a$ and compute the speed-up value for a parallel system having $n$ processors as follows:

$$S = \frac{1}{2}M^2C_a + 1 \quad (10)$$

The speed-up value is also independent of the number of processors in the system. However, both Eq.9 and Eq.10 includes hidden interdependences. Considering Fig.1b and having the same assumptions given above we can derive $T_{cycle}$ and compute it in another way:

$$T_{cycle} = T_{dw} + T_c + T_{dr} = \frac{N^2}{n}\left(2\left(\frac{\alpha}{K} + \beta\right) + M^2\gamma\right) \tag{11}$$

Since $\alpha$, $\beta$, and $K$ are communication related parameters, we defined "Coefficient of Communication" ,i.e. $C_c$, by those parameters. Thus, the number of processors became:

$$n = \frac{N^2}{T_{cycle}}\left(2\left(\frac{\alpha}{K} + \beta\right) + M^2\gamma\right) = \frac{N^2}{T_{cycle}}(C_c + M^2\gamma) \tag{12}$$

When Eq.12 is solved for $T_{cycle}$

$$T_{cycle} = \frac{N^2}{n}(C_c + M^2\gamma) \tag{13}$$

If a processing system, with typical values N=1024 pixels (of bytes, i.e., grey-level), $M = 11 pixels$, $K = 1024 bytes$, $\alpha = 2ms$, $\beta = 2\mu s$, and $\gamma = 2\mu s$, runs to achieve computation performance of $T_{cycle} = 10 seconds$, 25 client processors would be sufficient. For a video processing system with 32 client processors, the above given application would run for a single frame in 8.1875 seconds, i.e. $T_{cycle} = 8.1875 seconds$. It is obvious that these values are far from the real-time processing constraint of 40 ms per frame for PAL standard. Even an increase in the number of processors would not allow this system to run in real time. The bandwidth of the system for transferring 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:

$$T_{transmission} = N^2 C_c \tag{14}$$

$T_{transmission} = 8 seconds$ for the above example. Therefore, a more powerful data transmission system is required to provide real-time processing. Actually a parallel system connected via a high speed bus or a dedicated network would provide the required system characteristics for a real-time processing system, i.e., $T_{transmission} = 40ms$. To design such a system, utilising both I/O channel and processors, and running in real-time, one should calculate the number of processors by Eq.12 for a given system and application parameters providing $T_{transmission} = 40ms$.

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 $\gamma$) for different values of $M^2$ and $N^2$ are given in Fig.2a for $M^2 = 9$ and $N^2 = 288x360$; Fig.2b for $M^2 = 25$ and $N^2 = 288x360$; and Fig.3 for $M^2 = 9$ and $N^2 = 576x720$.

Fig.4a and Fig.4b show the value of $C_c$ with respect to real system parameters of $\alpha$, the latency time and $\beta$, data transmission time per byte. The comparison of the two graphs shows that the impact of the latency time is dominant as well as the packet size in determining the system communication parameter, $C_c$.

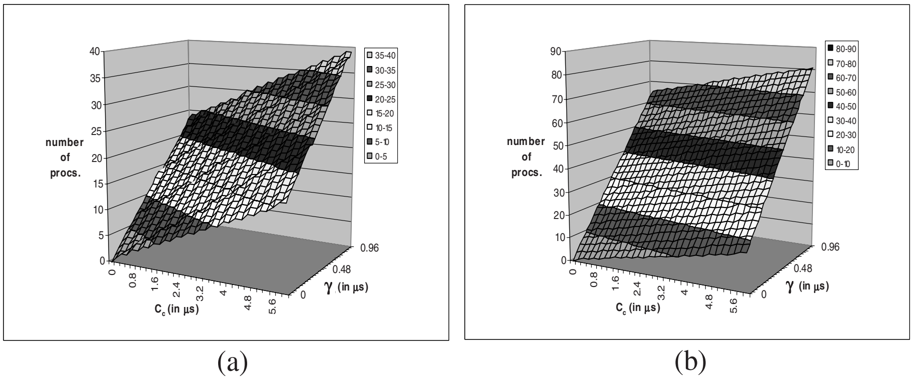

**Fig. 2.** Number of processors wrt $C_c$ and $\gamma$ for $N^2$=288x360 a)$M^2$=9 b)$M^2$=25

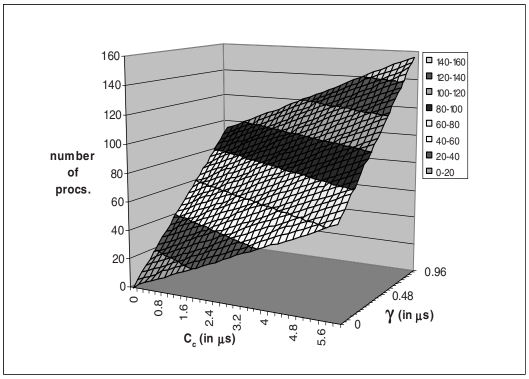

**Fig. 3.** Number of processors wrt $C_c$ and $\gamma$ for $N^2$=576x720 $M^2$=9

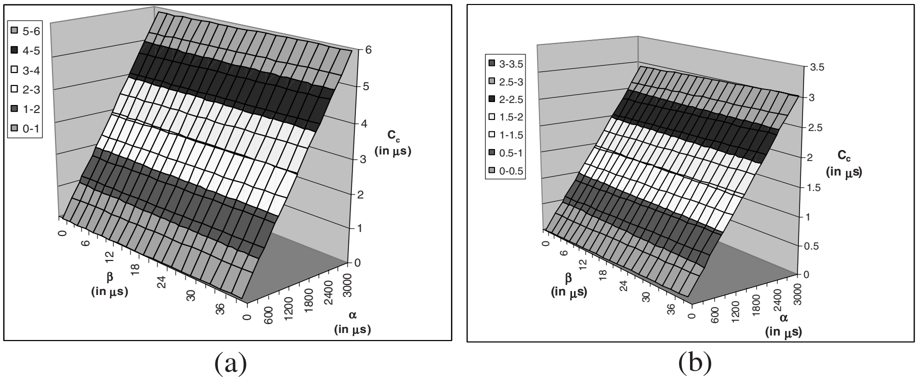

**Fig. 4.** $C_c$ wrt $\alpha$ and $\beta$ for a)$K$=1024 bytes b)$K$=2048 bytes

## 5 Conclusion

A new and optimal scheduling algorithm called Periodic Write-Read- Compute (PWRC) scheduling algorithm for real time video processing is defined and analysed. It requires a high level atomic write-read command. Since it is easy to implement such an indivisible command in high level programming, it does not introduce a new problem. The generic system architecture is based upon a client-server model having point-to- point communication between the host and every client processor. It has been shown that the proposed scheduling algorithm takes the same time to process video sequence with half the number of processors required in [6].

Further analysis of the PWRC scheduling algorithm yields a number of equations expressing the dependencies of the system characteristics, a communication constant, $C_c$, a processing constant $\gamma$, an architectural constanti, $C_a$ and application characteristics such as the size of the frame and width of the surrounding pixel frame. It is shown that either a parallel video processing system can be built up 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 equations.

## References

1. Agrawal R, Jagadish H V, Partitioning Techniques for Large-Grained Parallelism, IEEE Transactions on Computers, Vol.37, No.12, December,1988.
2. Altilar D T, Paker Y, Optimal Scheduling Algorithms for Communication Constrained Parallel Processing, Lecture Notes in Computer Science 2400, Euro-Par 2002, 27th – 30th August, Paderborn, Germany.
3. Altilar D T, Paker Y, An Optimal Scheduling Algorithm for Parallel Video Processing, Proceedings of International Conference on Multimedia Computing and Systems'98, Austin Texas USA, 245–258, July 1998.
4. Crandall P. E., Quinn M. J., A Partitioning Advisory System for Networked Data-parallel Processing, Concurrency: Practice and Experience, 479–495, August 1995.
5. Culler D, Karp R, Patterson D, Sahay A, Schauser K, Santos E, Subramonian R and Eicken T, LogP: Towards a realistic mode of parallel computation, Proceedings of 4th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming, Vol.28, May 1993.
6. Lee C., Hamdi M., Parallel Image Processing Applications on a Network of Workstations, Parallel Computing, 21 (1995), 137–160.
7. Moritz C A, Frank M, LoGPC: Modeling Network Contention in Message-Passing Programs, ACM Joint International Conference on Measurement and Modeling of Computer Systems, ACM Sigmetrics/Performance 98, Wisconsin, June 1998.
8. 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.