ENHANCING THE THROUGHPUT OF MESH-TYPE MULTIPROCESSOR SYSTEMS

M. Atiquzzaman

King Fahd University of Petroleum & Minerals, Saudi Arabia

ABSTRACT: Multiprocessor systems are employed to perform computation-intensive tasks, like image processing in real-time. Interprocessor communications has always been a performance bottleneck in bus-oriented multiprocessor systems. A two-dimensional array (mesh) of simple processors, operating in the SIMD mode has been found very suitable for low-level image processing. Degradation in the throughput of mesh-type multiprocessor systems due to interprocessor communications has been determined. It is observed that the ratio of inter-processor communications to the actual computation increases with an increase in the number of processors. Concurrent processing with overlapped inter-processor communication has been shown to significantly reduce the degradation, and has been illustrated in the case of a low-level image processing algorithm on a 2-dimensional array of processors having dedicated inter-processor links.

1. INTRODUCTION

Multiprocessor systems are used to enhance the execution speed of computer systems used for computation-intensive tasks. A typical example of such an application is image processing which involves handling of a massive amount of data, and characterised by CPU-bound algorithms. The requirement of tremendous amount of computing power gives rise to the need for multiprocessor systems.

Image processing can can be broadly divided into high and low-level operations. High-level operations operate on lists, graphs and trees rather than arrays, and often require sophisticated decision making. Low-level algorithms are usually shorter in length, but are highly repetitive in nature i.e. the same operation has to be performed on a massive amount of data. Typical examples of low-level image processing are image smoothing, enhancement etc. The repetitive nature of low-level algorithms lends itself to parallel processing using multiprocessor systems. Moreover, the two-dimensional nature of images are very suitable for being mapped onto a 2-dimensional array of processors - either Single Instruction Multiple Data (SIMD) or Multiple Instruction Multiple Data (MIMD) stream type architecture (1,2,3).

A 2-dimensional array of processors (along with local memories) with dedicated inter-processor links as shown in fig 1 may be used to process a picture. A processor may be employed behind each pixel of the picture, or the picture may be broken down into a number of regions (sub-images), each processor being assigned to a region (fig 2). Pixels of the image **f** will be represented by *f(i,j)*. A processor processes all the pixels of the region assigned to it.

Image averaging will be taken as an example of a low-level vision algorithm to illustrate the effect of the number of processors on performance of an array-type multiprocessor system. Image averaging on an image **f** can be represented by

$$g(x,y)=\frac{1}{(2k+1)^2}\sum_{i=-k}^{k}\sum_{j=-k}^{k}f(x+i,y+j)$$

where, the averaging window is taken to be of size $(2k+1)\times(2k+1)$, and **g** is the averaged picture. If the averaging is carried out by a single processor system $O(N^2)$ smoothing operations are required, each operation consisting of $(2k+1)^2$ additions and 1 division.

At first sight it might appear that addition of processors in a 2-dimensional array of processors would increase the throughput of the system linearly. But, in actual practice, the inter-processor communication overhead considerably slows down a system. The ratio of communication overhead to the actual processing time increases as the number of processors increases in a multiprocessor system. This paper discusses the trade-offs between the number of processors, the processing speed, and the communication overhead when the above illustrating vision algorithm is executed in a mesh-type machine.

Two methods are proposed to reduce the overhead due to inter-processor communication. Reduced inter-processor communication overhead will enhance the throughput of the system. The first method is concurrent processing and inter-processor communication over multiple inter-processor links, while the other is based on data replication at the time of loading data into the machine. Reduction of interprocessor communication overhead and increase in throughput for a 512 x 512 picture has been presented.

2. PERFORMANCE OF ARRAY-TYPE MULTIPROCESSOR SYSTEMS

Processor assignments to pixels may vary depending on the processing speed required. A processor may be employed behind one pixel or behind a group of pixels. Let the multiprocessor system consist of a $P \times P$ array of processors, and the image **f** to be processed is of size $N \times N$. Let $M \times M$ be the sub-image size such that $M \times P = N$. All the processors will be processing in parallel. Each processor has to perform $O(M^2)$ smoothing operations. In order to compute the new value of the edge pixels of the regions, pixels from neighbouring regions are required. The number of neighbouring pixels required for each region is $4M + 4$ as shown in fig 3. The same number of pixels are also to be passed on to the neighbouring sub-images. Exchange of a pixel means one inter-processor communication. Therefore, the total number of inter-processor communications required by each processor is $2(4M + 4)$. Assuming that one communication requires a time equivalent to k smoothing operations, the number of operations required by a processor to process its sub-image $= M^2 + 2(4M + 4)k$. If no communications were required, the number of operations required would be simply M^2. Therefore, speed-up due to parallel operation (assuming no interprocessor communication) as compared to a single processor system $= N^2 / M^2$. In the presence of inter-processor communications, speed-up due to parallel operation $= N^2 / (M^2 + 2(4M + 4)k)$

Table 1 shows as a function of the number of processors (i) the speed-up with and without communications, (ii) the ratio of the speed-up with interprocessor communications to the speed-up without interprocessor communications, and (iii) the ratio of the time taken for interprocessor communications to the time taken for the smoothing operations. Figures 4, 5

& 6 represent the data of Table 1 in graphical form. The following conclusions can be drawn from figures 4, 5 & 6.

1. As the number of processors in the array increases (i.e. the sub-image size becomes smaller) the overall operation of the machine becomes more and more I/O bound as compared to the actual computation (fig. 5).

2. As the operation of the system becomes increasingly I/O bound, the ratio of speed-up with inter-processor communications to speed-up without communication drops sharply (Fig 6). With 32 x 32 processors, the speed-up in presence of communications is about 79% of that without communication, while with 256 x 256 processors the percentage is about 25%.

This indicates a sharp decline in the relative throughput of the system as the number of processors increases i.e. though the overall computing power of the system increases, the throughput does not increase linearly with the increase in the number of processors. The rate of increase of interprocessor communications is higher than the rate of increase of the computing power. It is solely due to the fact that with large number of processors, a substantial amount of time is spent as overhead in inter-processor communications.

3. PROPOSED TECHNIQUES TO INCREASE THROUGHPUT

It is apparent from the previous section that due to the inter-processor communication arising due to the exchange of border pixels between regions the throughput of the system does not increase linearly with the addition of processors. It should be noted that the throughput of a machine will increase linearly with the number of processors for algorithms which do not require inter-processor communications. The following two techniques are proposed to increase the throughput, assuming that the algorithms require inter-processor communications.

3.1 Method 1: Concurrent Processing and Communication

The first proposed method is to use processors with concurrent communication capability over multiple links. A typical example of such a processor is the Transputer (4). The Transputer is a 32-bit microprocessor from Inmos with concurrent communication capability. It has been designed as a building block for multiple-processor systems using point-to-point communication between processors. It has 2 Kbytes of on-chip RAM and 4 dedicated links having concurrent communication capability. The CPU and the 4 links can operate in parallel. The links make the Transputer a very attractive candidate for being used as processors in an array-type architecture. Processors with 8 links capable of concurrent processing and communication over inter-processor links can be designed. Processors used in machines like CLIP and MPP have links for inter-processor communication, but the links lack the capability of concurrent communications. With a processor having 8 links with concurrent communication capability, the exchange of pixels between regions will take place as follows.

Step 1 Communication along thick arrows (fig 7(a)) will take place in M phases and along the thin arrows will require 1 communication. The first phase will have 8 concurrent communications and the rest M-1 phases each will have 4 concurrent communications. At the end of the M phases, a processor will receive the required elements from the north, west, north-west, and south-west regions, and transmit its values to the east, south, south-east, and north-east regions.

Step 2 Same as step 1 except pixels are received from east, south, south-east, and north-east regions, and transmitted to north, west, north-west, and south-west regions (fig 7(b)).

The total communication phases in order to exchange pixels is therefore $2M$. In this method, the number of operations required = $M^2 + (2M)k$. The processing time remains the same as before, but the communication time is reduced to less than one-fourth. Speed-up due to parallel operation (assuming inter-processor communication) = $N^2 / (M^2 + (2M)k)$. Using this proposed method, Table 2 gives the same variables as in Table 1. The speedup without inter-processor communications remain the same as in Table 1, and hence has not been repeated.

It has been assumed throughout that the computation of the averaging operation starts after all the required pixels have been exchanged between neighbouring processors. With a Transputer the computation of the pixels other than the border pixels can proceed in parallel to the communication. This will further increase the overall speed of the system. Table 3 shows the speed-up and other parameters when execution and communication are carried out in parallel using a processor like the Transputer.

It is found from Table 3 that there is no advantage of concurrent processing & communication when the processor array size is 256 x 256 or more. This is because processing can not proceed until pixels from neighbouring sub-regions have been received.

3.2 Method 2: Replication of Data at Loading Time

The second method to reduce the bottleneck due to inter-processor communication is to replicate pixels while a picture is being loaded into the system. The pixels which are required by multiple processors should be replicated and distributed among those processors at the picture-loading time. In this case, border pixels should be distributed among all those processors which will require them at processing time, thereby, eliminating inter-processor communications. Details of this method can be found in (5).

4. CONCLUSION

The throughput of a multiprocessor system does not increase linearly with the increase in the number of processors. This is due to a substantial amount of time spent for communication between processors for exchange of pixels of neighbouring sub-images. As the number of processors increases, the size of the sub-images decreases thereby increasing the number of inter-processor communications required. It is found that the rate of increase of inter-processor communications is higher than the increase in the processing power when the number of processors are increased. As a result, the ratio of inter-processor communication to the processing power increases, thereby, preventing a linear increase in throughput of the system. The optimum number of processors required in a multiprocessor system is a trade-off between the cost of the system and the computing power required.

Using processing elements capable of concurrent communication over inter-processor links considerable improves the throughput of a system. Processing elements capable of concurrent program execution and communication further improves the throughput upto a certain limit. This limit is reached at the point where the CPU can not proceed with program execution until data is available. At that limit the throughput is the same as that of a machine capable of only concurrent inter-processor communications. For an image smoothing operation, this happens when the sub-image size becomes too small.

Replication of pixels while a picture is being loaded into a multiprocessor system eliminates the need for inter-processor communications during the first pass of a picture, thereby increasing the throughput of the system linearly with the number of processors.

5. ACKNOWLEDGEMENT

This research was supported by King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

REFERENCES

1. FOUNTAIN, T.J., and GOETCHERIAN, V.: 'CLIP4 parallel processing system', *IEE Proc.*, Pt. E, 1980, **127**, 5.

2. POTTER, J.L.: 'Image processing on the massively parallel processor', *Computer*, 1983, **16**, 1, pp. 14-20.

3. SIEGEL, H.J.: 'PASM: A reconfigurable multimicrocomputer system for image processing', in *Computing structures for image processing*, DUFF, M.J.B. (Ed), (Academic Press, London, 1983).

4. Transputer reference manual, Inmos Ltd., Bristol, UK, 1986.

5. ATIQUZZAMAN, M.: 'Algorithms and architectures for automatic traffic analysis', Ph.D. Thesis, Univ. of Manchester Inst. of Science & Technology, England, Dec 1986.

TABLE 1 -Serial communications between processors, N = 512, k = 0.5

Number of processors *P x P*	Subimage size *M x M*	Speed-up (without interprocessor communication)	Speed-up (with interprocessor communication)	Speedup (with inter-processor communication) / Speedup (without inter-processor communication)	communication time / operation time
16 x 16	32 x 32	256	226	0.880	0.129
32 x 32	16 x 16	1024	809	0.790	0.265
64 x 64	8 x 8	4096	2621	0.639	0.562
128 x 128	4 x 4	16,384	7282	0.444	1.25
256 x 256	2 x 2	65,536	16,384	0.250	3
512 x 512	1 x 1	262,144	29,127	0.111	8

TABLE 2 -Concurrent communication between processors, N = 512, k = 0.5

Number of processors *P x P*	Speed-up (with interprocessor communication)	Speedup (with inter-processor communication) / Speedup (without inter-processor communication)	communication time / operation time
16 x 16	248	0.969	0.031
32 x 32	963	0.940	0.063
64 x 64	3640	0.889	0.125
128 x 128	13,107	0.799	0.250
256 x 256	43,690	0.667	0.500
512 x 512	131,072	0.500	1.000

TABLE 3 -Concurrent execution and communication between processors, N = 512, k = 0.5

Number of processors *P x P*	Number of communications overlapping with program execution	Speed-up (with interprocessor communication)	Speedup (with inter-processor communication) / Speedup (without inter-processor communication)
16 x 16	64	256	1
32 x 32	32	1024	1
64 x 64	16	4096	1
128 x 128	8	16,384	1
256 x 256	0	43,690	0.666
512 x 512	0	131,072	0.500

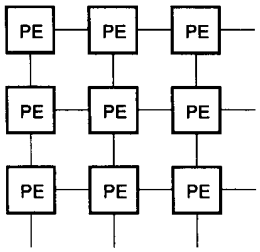


Figure 1 A 2-dimensional array of processors

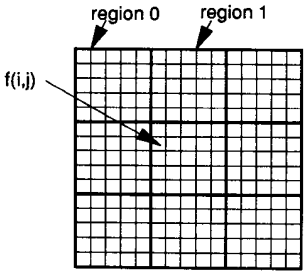


Figure 2 An image f divided into regions

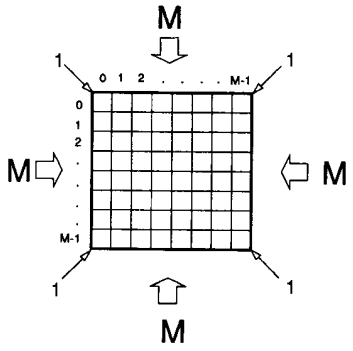


Figure 3 Number of neighbouring pixels required for each region

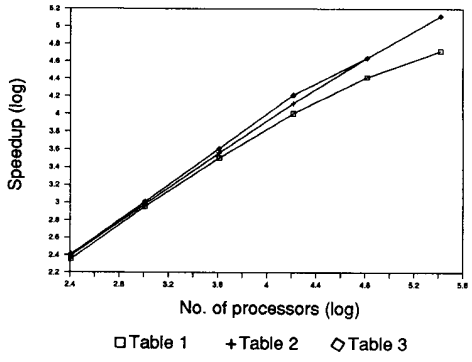


Figure 4 Speedup versus number of processors in the proposed techniques

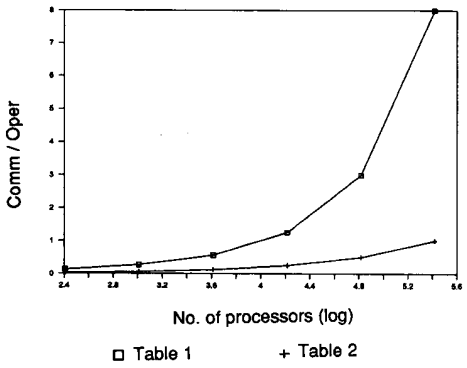


Figure 5 Ratio of communication to operation in the proposed techniques

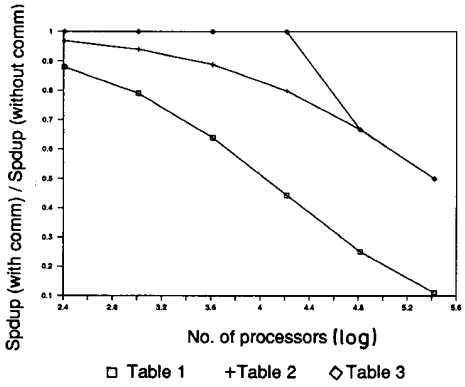
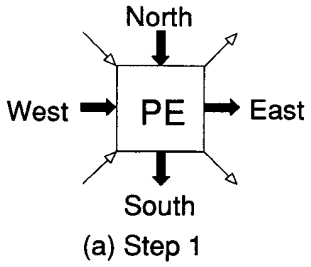


Figure 6 Ratio of speedup with and without communications



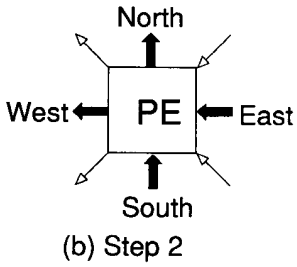


Figure 7 Flow of data in concurrent communication