MVA '92 IAPR Workshop on Machine Vision Applications Dec. 7-9,1992, Tokyo

A 'Hyper-Pyramid' Architecture for Massively Parallel Image Processing

Mohamed Akil and Eric Dujardin

Groupe E.S.I.E.E. Laboratoire I.A.A.I. 2 Bd Blaise Pascal
B.P.99, 93162 NOISY-LE-GRAND Cedex (FRANCE)

Abstract

The image processing involves many computation on a large data volume. We show that the "divide and conquer" method may be used to decrease the execution time of algorithms. We need to have a suitable topology to profit fully of data parallelism. The topologies as mesh or pyramid are too much restraining on the possibilities of allowed movement data. In other way the hypercube topology has a large wiring complexity. Also we propose a hyper-pyramid topology, which is a compromise between wiring complexity and algorithmic complexity. Thus the histogram and component labeling algorithms are respectively an algorithmic complexity in O(Vlog n) and in O($log^2 n$). Where V is the number of colour in image of n pixels.

1 Introduction

The image processing is an application requiring many computation with a large variety of communications: local, global, etc. For this, many parallel architecture were proposed. The best know is the mesh connected computer, for example: MPP [1], CLIP4 [8]. These machines are efficient for local processing, but, their performances are deteriorated by global communications. The pyramid has a diameter of 2 log n. An example of machine using that topology is SPHINX [11],[12], and PAPIA [2]. But at the apex there is a strong reduction of communication. And with the same way, as the mesh connected computer, the communications inside each level can not be irregular. Suitable architectures for this communication pattern is necessary for intermediate level and high level [7]. As a large amount of data are processed, we can use the "divide and conquer" method. For this we estimate the brought profit by this method.

The algorithm complexity of "divide and conquer" algorithm can be written by the following recurrency:

$$Tc(n) = Tc(n/2) + t(n)$$

With Tc(n) is the processing time to compute n data, t(n) is the merge time of two Tc(n/2) subsolutions. If we have a previous algorithm with an algorithmic complexity in O ($\sqrt{n}$), we must calculate the n_0 corresponding to the minimal amount of data from which the "divide and conquer" method is interesting to use. We find:

$$t(n_0) = \sqrt{(n_0)}(1 - 1\frac{1}{\sqrt{(2)}})$$

Thus, the merging function must be inferior to $\sqrt{n_0}(1 - 1\frac{1}{\sqrt{2}})$.

If for example $t(n_0)$ is in $log(n_0)$ then the method may be used with images of than more of 512 pixels. When we know that satellite images as SPOT hold 50 million of pixels, we deduce that this method can be suitable for image processing. A parallel algorithm on a suitable machine can get a real profit of execution time.

A lower bound of complexity algorithmic of an algorithm is in first case its complexity. In second case, the data rearrangement, and so the data movement on a parallel computer, can be a source of bad execution time.

Thus a machine with a mesh topology has a lower bound complexity in O($\sqrt{n}$). Whereas in a pyramid topology the lower bound complexity is in O($\sqrt[3]{n}$).

In the "divide and conquer" method the merging function may be a complex reorganization data function. The irregular movements can not carry out well on topologies as mesh and pyramid. An example of complex function is the sort function. The hypercube has a complexity time in O($log^2 n$) for sorting n data. But his wiring complexity is high.

For this we introduce a new topology, the hyper-pyramid, which is suitable for "divide and conquer"

method with a low wiring complexity. In the follow of this paper, we present the hyper-pyramid topology: section 2. Next in section 3 we show the implementation of the method and we exhibit its performance. In section 4, we show a functional simulator, which is a step of building of a hardware machine.

2 Architecture description

We present the structure of hyper-pyramid by comparing with one of pyramid. A pyramid topology, P(r, h) is characterized by its reduction factor r and its height h. The h factor is the number of level in the pyramid, h= log(n) with n is the number of processors at the base. The hyper-pyramid, HP(r, h') is characterized by h' and r factors. The r factor is the same as the one of pyramid. The h' factor is the number of level, but $h' = \frac{1}{2}h$.

For this, at the apex (k=0) the structure has r^h processors. At the base (k=h'), a hyper-pyramid has $r^{2h'}$ processors.

Moreover, we define sets of processors in each level, which we call cells, inside the ones the merging function will be executed. For this, we have choose an efficient topology for cells with which the sort function may be executed in O($log^2 s$) time, s is the size of cell. Because the "divide and conquer" method merges subproblems higher and higher until merging all data, the size of cell increases from the base to the apex level.

This cells are organized as pyramid topology p(r^2, h'), with h' is the same heigh as the one of hyper-pyramid topology. Thus at the apex there is only one cell with $S_0 = r^{h'}$ processors. At the level k=1, there is r^2 cells, each cell has $S_1 = \frac{S_0}{2^{log(r)}}$ processor. Thus we built recursively cells until the level k=h where $S_h = 1$ processors.

For example on Table 1, we show the organization of processors and cells at each level in the hyper-pyramid of reduction 4 and with 4 levels: HP(4, 3).

Level	3	2	1	0
Number of Processor	4096	1024	256	64
Number of Cell	4096	256	16	1
Size of cell	1	4	16	64

Table 1: Example of structure of a HP(4, 3)

In conclusion, we may describe the hyper-pyramid with recursive way too.

HP(r, h)= r^2 HP(r, h-1) + one cell of r^h processors
HP(r, h) = one processor

This equation means that the hyper-pyramid of h height is composed of r^2 hyper-pyramids of h-1 height. And addition the apex consist of r^h processors organized in one cell.

3 Divide and conquer method

The structure of algorithms recognizes some methods used to solve many problems. The "divide and conquer" method is the one of those methods. In the same way the "bottom-up divide and conquer" method is used in image processing [16]. To process the solution of an image $\mathcal{I}(n, n)$, of $n \times n$ pixels, we calculate recursively the solutions called $S(\{1:\frac{n}{2}\},\{1:n\})$ and $S(\{\frac{n}{2}+1:n\},\{1:n\})$ of each sub-images of size $\frac{n}{2} \times n$ pixels. On the figure 1 we show the divide and conquer method on an image.

The implementation of this method on hyper-pyramid is the same as the one of Miller and Stout on pyramid [15]. The data of image are spread out on processors at the base. Each processor has 2^c data, c is a small constant ($2^c = n_0$). At the base, processors preprocess that data. Next, the intermediate results are sent to its father. The father processors combine comming data of its child processors.

We note:

$$\mathcal{S}(\{1:n\},\{1:n\}) = \mathcal{S}(\{1:\frac{n}{2}\},\{1:n\}) * \mathcal{S}(\{\frac{n}{2}+1:n\},\{1:n\})$$

Then the solution of an image $\mathcal{I}(n, n)$ is:

$$\begin{array}{rcccc}
\mathcal{S}(\{1:n\},\{1:n\}) = & \mathcal{S}(\{1:\frac{n}{2}\},\{1:n\}) & * & \mathcal{S}(\{\frac{n}{2}+1:n\},\{1:n\}) \\
= & [\mathcal{S}(\{1:\frac{n}{2}\},\{1:\frac{n}{2}\}) * \mathcal{S}(\{\frac{n}{2}+1:n\},\{1:\frac{n}{2}\})] & * & [\mathcal{S}(\{1:\frac{n}{2}\},\{\frac{n}{2}+1:n\}) * \mathcal{S}(\{\frac{n}{2}+1:n\},\{\frac{n}{2}+1:n\})] \\
\vdots & \vdots & & \vdots \\
= & [..[\mathcal{S}(\{1:2\},\{1\}) * & :. & *\mathcal{S}(\{n-1,n\},\{n\})]..] \\
= & [..[[\mathcal{S}(\{1\},\{1\})] * [\mathcal{S}(\{2\},\{1\})]]* & :. & *[[\mathcal{S}(\{n-1\},\{n\})][\mathcal{S}(\{n\},\{n\})]]..]
\end{array}$$

Figure 1: divide and conquer method on an image

Thus data of $4 * 2^c$ size are coherency (We have choose a hyper-pyramid with 4 reduction). The next step data are merged on the same level by all processor defined in each cell. Their results are sent to their fathers. And so forth, till at the apex where all data are merged by all processors in the apex.

We have describe in [5] that the multiresolution algorithms are been suitable on the hyper-pyramid topology. We showed it by calculating the vertical data movements in multiresolution algorithm which are smaller than the ones in "divide and conquer" algorithms.

For component labeling algorithm we get on hyper-pyramid topology an algorithmic complexity in O($log^2 n$)[5] where n is the number of pixel in image. On hypercube topology this complexity is the same [3]. The pyramid topology has a complexity in O($\sqrt[4]{n}$) [15], and in mesh topology the algorithmic complexity is in O($\sqrt{n}$). For the histogram of a grey level image we get on hyper-pyramid topology has an algorithmic complexity in O(Vlog n) with V is the number of grey level in image [6]. For pyramid and hypercube topologies the algorithmic complexity is the same [3]. For mesh topology, the complexity is in O($\sqrt{n}$).

4 Functional Simulator

We choose for the target machine a passing message architecture. As the connection machine [10] proves it, with its hypercube network and routing algorithm, a passing message machine is suitable for a large range of application.

We wrote a functional simulator to estimate performances of target machine. The first purpose is to get communication performances. The execution model for processors has not been studied. But the divide and conquer method shows some communication instruction model. To get a range of comparison, we may chose a topology between, hypercube, mesh, pyramid, hyper-pyramid. We estimated the general performance of network as the latency. The latency is the time taken by a message to get throught the network from source processor to destination processor. Each processor sends a message to a randomly chosen processor. Each processor sends an average of 50 messages in 100 cycles. The simulation stops when the average message latency is stable.

The figure2 show the latency in function of number of processors. The number of processor on x axis is from 100 processors to 2000 processors in a logarithmic form. The curve of hypercube is in linear. The hyper-pyramid is between the mesh and hypercube. The performances on the chart, are the performances are least good that the ones of hypercube, because the communication pattern are not suitable for the hyper-pyramid topologie. But theses performances for that parttern commmunication are acceptable.

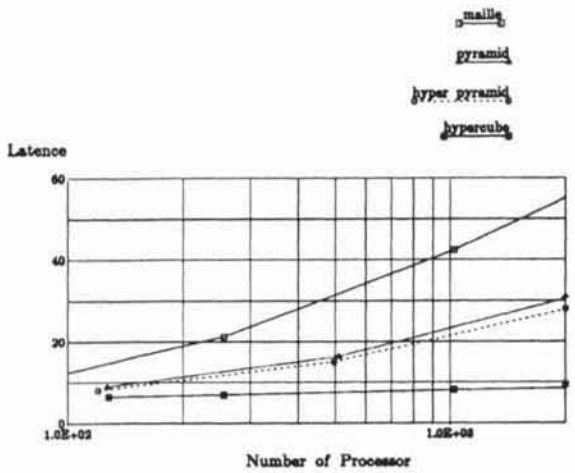

Figure 2: latency of number of message for mesh, hypercube, pyramid of 2 reduction and hyper-pyramid of 2 reduction too.

Conclusion

In this paper, we have shown the that "divide and conquer" method is well suitable for image processing algorithms. Moreover, it may consider as an algorithmic tool by giving a task distribution on parallel machine. The hyper-pyramid topology is a topology with a good matching communication task and organization processor, with a low wiring complexity. Now, we have developed a software simulator, thus we can develop applications and we get performances with any topologies. The asset of this topology is its compromise between wiring complexity and algorithmic complexity that we can get. We have demonstrated in [5] that the wiring complexity is much smaller then hypercube.

References

[1] K.E Batcher, "Design of Massively Parallel Processor", IEEE Transactions on computers, vol. C-28, No 2, 1980 pp 836-840

[2] V. Cantoni, M. Ferreti, S. Levialdi and F. Makberti. " A pyramid project using integrated technology" in Integrated Technology for parallel image, Processing Academic Press 1985.

[3] R. Cypter and J Sanz, "SIMD Architectures an Algorithms for Image Processing and Computer Vision", IEEE Transactions on acoustics, speech and signal processing vol. 37, No 12, December 1989 pp2158-2174

[4] E. Dujardin and M. Akil, "Hyper-pyramid routing chip", internal report (in french)

[5] E. Dujardin and M. Akil, "A Hyper-Pyramid Network Topology for Image Processing", to be published in The fourth Symposium on the Frontiers of Massively Parallel Computation, Virginia, october 92

[6] E. Dujardin and M. Akil, "Programmation de l'histogramme sur differentes topologies" internal report (in french)

[7] "Intermediate-Level Image Processing", Edited by M.J.B DUFF Academic Press 1986.

[8] T.J Foutain, "A Survey of bit-serial array processor circuits", in "Computing structures for image processing", Ed. Academic Press 1983, pp 1-14

[9] J.Foutain, "Array Architeures for Iconic and Symbolic image Processing", 8th Int. Conference on Pattern Recognition 1986, pp24-33

[10] D. Hillis, "The conection machine", MIT Press 1985

[11] A. Merigot, B. Zavidovique and F. Devos "SPHINX, A Pyramidal Approch to Parallel Image Processing", in Proc IEEE Workshop Comput. Architecture Analysis Image Database Management November 1985, pp 107-111

[12] A. Merigot and al., "SPHINX, a massively parallel pyramid machine for artificial vision", 7^{eme} congrès RFIA, Nov 1989, pp 185-196

[13] R. Miller and Q. Stout, "Pyramid computer algorithms for determining geometric properties of images", Proc. 1985- ACM Symposium on computer geometry, pp 263-271

[14] R. Miller and Q. Stout, "Geometric algorithms for digitalized pictures on a Mesh-connected computer", determining geometric properties of images", IEEE Trans. on pattern analysis and machine intelligence, vol. PAMI-7, no 2, March 1985, pp 216-228

[15] R. Miller and Q. Stout, "Data Movement Techniques for the Pyramid Computer", SIAM J. Comp., vol 16, No. 1, February 1987, pp 38-60

[16] D.Nassimi and S. Sahni, "Finding connected components and connected ones on a mesh-conneceted parallel computer" Siam J. Computer. Vol 9, no. 4, pp 744-757, 1980

[17] Q. Stout, "Algorithm-guided design considerations for meshes and pyramids" in "Intermediate-Level Image Processing", Ed M. Duff, Academic Press 1986.