Journal of Computer Science 2 (8): 615-618, 2006
ISSN 1549-3636

Parallelisation of Algorithms of Mathematical Morphology

Abdallah Boukerram and Samira Ait Kaci Azzou
Department of Informatique, University F. Abbas, 19000 Sétif, Algérie

Abstract: The tools of mathematical morphology developed within the framework of the image processing, require of big capacity of data and the very high costs of execution. So today, the limits of the sequential machines are not to be any more to show, the passage in the new parallel machines of type Simd, Mimd, clusters or grids is imperative. This paper deals with problems related to the parallelisation of the algorithm of mathematical morphology and highlights the resources influencing over the computing time. This study leans on the various levels of parallelisable calculation to evaluate the awaited profits then in term of processing time. An implementation of a whole of algorithms of reference is carried out on a cluster and a simd computer.

Key words: Mathematical morphology, parallel algorithms, time of processing, clusters, simd computer

INTRODUCTION

The developers of parallel systems hesitate between two approaches: it is necessary to conceive computers from the classes of algorithms or then to conceive architecture to determine then, the applications the best adapted to these machines. The first approach results generally in a high-performance machine in a domain restricted by applications; the second has the advantage to be able to answer needs of more diversified applications. To answer at this problem, as regards the tools of the mathematical morphology; one must determine all calculations which can make object of a parallelism. This study, made a projection on the existing parallel architectures on the market so as simd processors, mimd processors, vectoriel processors, clusters and grid computing[1].

Characteristics of the morphological operators: The mathematical morphology is an ensemblist theory used in image processing. This theory is built from two basic operators who are the erosion and the dilatation. Without for it give an exhaustive list of all the algorithms of the mathematical morphology, we remind that the mathematical morphology was developed from two basic operations which are the Erosion and the Dilatation[2,3]:

Erosion: We call morphological Erosion of an image X, according to a structuring element B, the set E, formed by points x Є X, satisfying the condition of inclusion {Bx⊂X }.
Bx = structuring element centred in the point x
Notation: E (X, B) = {x Є X: / Bx included in X}

Dilatation: We call morphological Dilatation of an image X, according to a structuring element B, the set D, constituted by points x Є X such as the intersection of Bx (B centred in x) with X, is not empty.
Notation: D (X, B) = {x: (Bx ∩ X ≠ Ø}

Notion of structuring element: the structuring element constitutes the basic element of the mathematical morphology. Of flat geometrical shape (disk, squared, segment, etc.), he is defined by a reference point or origin, one or several directions and a dimension

The algorithms of image processing used since the acquisition of the data until the final phase of processing are very varied. A simplistic classification given in the literature lists them in two big families; the low-level algorithms and the high-level algorithms.

This classification is established according to the structures of treated data and the used operators. We group together in the low treatments level, the algorithms manipulating arithmetic or logic operators, applied to images organized according to regular wefts. We group in treatments high level, the algorithms treating structures of more complex data so as list, arborescence or graph. The algorithms of morphology are low treatments level which manipulate data images presented in the form of two dimensionally structures of pixels.

* The punctual operators: this class contains the algorithms of threshold and all the ensembles operators applied to the binary images.
* The local operators: all the transformations making use of structuring elements is local operations (erosion, dilation, skeleton, outlines, etc...): the processing of a pixel depends on this one and on a certain number of her definite neighbours formed it and the dimension of the structuring element.

Corresponding Author: Abdallah Boukerram, Department of Informatique, University F. Abbas, 19000 Sétif, Algérie

* The iterative operators: let us remind that the morphological basic transformation are iterative: erode or dilate an object by an element structuring of dimension n, mean eroding it or dilating it, n time by the same element structuring of dimension 1. We also count the algorithms constituted by successive suites of iterations, such as the skeleton by the method of thinning down, the calculation of centre of gravity (successive erosions). The reconstruction of an image damaged from the other one (series of conditional dilations) as iterative algorithms.

We codify advantageously these iterative algorithms, when the number of iterations to be made is known at the time of the analysis of the application. It is the case of the algorithms of filtering from which often, the number of iterations is determined by the dimension of structuring elements according to objects to extract or to eliminate in a given scene. On the other hand for the algorithms the number of iterations of which depends on the context of the image, the number of iterations is not known. Two possibilities offer themselves for their coding:

According to probability methods, we estimate the number necessities of iterations for the obtaining of the aimed results. Often, we tend to foresee more iterations than one needs. Consequence: it results from it, of the wastes of time caused by the useless operations.

We observe the result obtained after every step of iteration until obtaining of the waited results. Consequence: method punished by the number too much of input-output.

MATERIALS AND METHODS

Although, the morphological operators are simple. But their integration in a complete chain of image processing take fast big proportions in times of processing. It is thus necessary to think of the parallelism: simultaneous execution of number of spots, which we can envisage at several levels (bit, neighbourhood, operator and program or level data).

Parallelism at the level spaces pixels: A morphological application is constituted by a succession of elementary alterations, which often, consist in applying the same process or the operator to all the pixels of the image. Algorithms appearing under the shape:

```
begin
 for any pixel "x" of the image do
 {
 Calculation of "x" on a neighborhood Vi
 }
 endfor;
endbegin
```

We just measure a phase of iteration to (K*256*103) cycles of operations for an image (512*512) pixels with K the number of instructions necessary for the pixel processing.

The morphological operators are local. For a given operator we make the same processing on all the points of the image. Of more the treatments of a point embellish with images is independent from the context of the image. Intrinsically, we think of the massive parallelism on pixels: computer consisted of elementary processors (EPs), working in mode Simd (simple stream of instruction, multiple stream of data), correspond perfectly to this mode of parallelism[4].

Example: Processing of an image (M*N) pixels on a system of (Px*Py) EPs working in synchronous mode. The image is partitioned blocks of dimensions equal to those of the processor who are in charge of in the local memories of various EPs. The functioning Simd, made that an elementary operation is made in parallel on every EP, on each of the pixels who is attributed to him.

Once the processing of a block ended we pass in the following block until the image is completely treated[5]. So, the complete realization of an operator necessitates K cycles of operations by pixel would take: K (M / Px) * (N / py) = K (M*N) / (Px*Py) cycles of instructions, instead of: (K*M*N) in the case of a sequential treatment.

In these times of calculation, add the times necessary for the loads of data in the memories of EPs and possibly the time of adaptation of the data on the geometry of processors. But at this level already, we can suspect the influential features on the performances of such a system:

* The size of the computer in number of EP
* The power of calculation and the set of instructions which offers the system
* The functions of access to the nearby processors which offers the network of interconnection
* The management of input-output is often penalizing in the transfers times of data.

Parallelism concerning the operators: Significant earnings of calculation can be waited, if we parallelise the operators enjoying a big frequency of use. In mathematical morphology, we think of the local operators leaning on the notion of structuring element. In the classic computers, the ALU work generally on two operands of data in entry to deliver a result from it in release. To realize a local operation on a neighbourhood (Vx*Vy) pixels, it would be necessary to realize exactly [log2 (Vx*Vy)] under elementary operations operating each on a pair of data.

There is two solutions, for operators parallelism: The multi-operators or the technique of the pipeline The multi-operators is exploited well in supercomputers and techniques of the pipeline is widely integrated into the architecture of the current computers.

Approaches macro-pipeline of the parallelism: This technique is similar to the pipeline method. The operators apply to streams of images instead of data. The system is configured in macro-pipeline: put in waterfall of processors some following the others. The images are introduced one after the other and circulate of a processor in a neighbour according to the producing/consumer mode. In every passage in a processor, the image undergoes a treatment and passes on its result in the following processor. The processor composing the last floor of the macro-pipeline supplied the final result of processing.

Once the macro begun pipeline, a theoretical earning of a report of p (p=number of processors of the system) can be expected, if the following hypotheses are satisfied:

* Continuous streams of images in entry, with sequential chain of treatments
* Quantum of time of execution of the various equal or rather homogeneous modules
* Space out memory being enough in each of the processors for containing data and program
* Compatibility of the structures of data presented to the entry and to the exit of a processor, with points of synchronization on buffers used in the exchanges of data.

Method of parallelism: We decompose the program into a suite of modules where the enchainment of execution respects the sequential of the initial algorithm.

For the operators on which is made certain number of iterations (erosion, dilation, thinning down, conditional alterations, etc...), it will be a question of distributing the total number of iterations on all the processors. So on a computer in p processors, a morphological transformation according to a structuring element B of dimension n, would decompose p elementary alterations according to an element structuring of dimension 1. Each of the processors will have the same load of calculation and will make (n/p) operations.

Example: We configure advantageously a system in four floors, for the algorithms of filtering, skeleton, transformation top hat or the others. The algorithm FILTRE+(B, n, m) described by Serra[2], distributes in a optimal way with a load balancing of of calculations (number of iterations by operator) on a cycle in four processors as follows :

operator EROSION	n iterations
operator DILATION	(n+m)/2 iterations
operator EROSION	(n+m)/2 iterations
operator DILATION	m iterations

The algorithmic parallelism: We decompose a program a set of processes. Then, we isolate the independent processes to be treated in parallel.

A process is a module, a procedure or simply a sequence of executables instructions by a processor. Two processes are said independents if they can be executed in parallel.

Two processes arisen from a decomposition of a program can use common resources of type files, space out memory or the others. To be divided these resources, the processes need to communicate between them and to synchronize[6,7].

The mechanisms of synchronization are of order multiple (semaphores, events, acknowledgement of receipt ...). They differ from a machine in the other one according to the used operational systems[1]. The rival programming brings a solution of the codification of the parallelism algorithmic. We can quote the following languages, the C Parallel, the Fortran Parallel, OCCAM3 or ADA[8]. In the rival programming, there is no general method to determine the processes independent from a given algorithm, the methods heuristics remains the only appeal to the users.

Method

* Decompose the algorithm a set of processes
* Determine the exchanges of data to inter-processes
* Isolate the independent processes
* Allocation of the logical processes in the physical entities
* Execution simultaneous of the rival processes.

Example: The algorithm “morphological gradient” decomposes easily into two independent processes to know the operator INF and SUP who are respectively the operators EROSION and DILATION of images at several levels of grey.

Parallelism at the level of the data: The parallelism of data is not specific to the treatment of images; he can be envisaged for any application called to process or to manipulate big files or big data base. We divided a file of data a set of sub-files or an image in sub-images to be treated in parallel on system multi-processors. Each of these processors has a unit of calculation and a local memory where are stored programs and data such as clusters[8,9].

Method of parallelisation: The image is partitioned in blocks according to the geometry of the network interconnection of the system. Then we load these blocks in the local memoirs of the various processors to be treated in parallel: models known under the name of "geometrical parallelism ".

For the morphological algorithms, we apply the same processing to each of the parts of the image: parallelism Spmd (simple stream of program, multiple streams of data) model says. For the processing, the same copy of a program is loaded with in each of the processors. The various sub-images are treated so simultaneously on all the processors of the system.

Material constraints

* Local memory of a processor of a capacity of at least equal to the size of the window to be treated for the punctual operators and with twice the size of the window for the local operators.
* Correspondence of the geometry of processors with those of the image or the mechanism of adaptation of data on the geometry of the processors of the system.

Estimation of the time of processing: To consider time of calculation of architecture spmd, we settle the same hypotheses as those used by J.L Basile for the estimation of system multi-processors[10].

p= number of processors of the system
Tc= time of calculation of a par of the image
Tt= time of transfer of a part of the image
R= number of regions of the partitioned image
TT=time of total processing of the image
R= K*p (K integer > = 1)

The time of processing of a sub-image is determined by time necessary for its load followed by the time of calculation necessary for its treatment followed by time taken by the transfer of the results.

Time of a cycle sub-embellish with images = (Tt + Tc + Tt) = (Tc + 2*Tt)

Case 1: In the case or the system has a network of interconnection, in measure to be able to load the data in parallel on all the processors. The time of complete treatment of a requiring image K iterations will be equal to: (R/p) * (time of a cycle of treatment of one under-image).

Total time of execution of an image = (R/p) * (Tc + 2*Tt) = K (You + 2*Tt).

Case 2: In the case of simple configuration, or processors are connected by a unique bus, it is necessary to add time necessities in transfers of data and to conjugate the waits caused by the last processors served in data. We distinguish two possibilities:

a. Tc < (p-1) *Tt: In that case handbook, we have a total covering of the time of calculation by that of the transfers of data. The complete treatment of P regions, takes then just the time necessary for the transfers of p regions. Let be a time: (p*Tt) + (p*Tt) = (2p*Tt)

b. Tc > = (p-1) *Tt: The time of calculation taken by the last processor covers the totality of time necessary for the transfers of (p-1) regions of the antecedent processors. The time of treatment of p regions is then given by:

(p-1) Tt + Tt + Tc + Tt) = Tc + (p+1)* Tt

The total time of an image processing:

TT = K(Tc+ (p+1)*Tt) if Tc/Tt > = (p-1)
= 2 (K*p*Tt) else

To make profitable such systems, it is necessary to envisage solutions of load balancing which nowadays are in the research state[12].

CONCLUSION

The parallelism on the algorithms of mathematical morphology can be envisaged at several levels. Each of the levels requires particular architecture:

* Parallelism on pixels of the image: Simd architecture
* Parallelism on parts of the images: Cluster or spmd computer
* Technique of the pipeline for the local operators
* Macro-pipeline for a parallelism of streams of data.

The federation in a grid computing of the unit of these heterogeneous system simd computer and cluster is a possible solution for the optimal parallelism of morphological algorithms. We shall also underline the many dependences of the algorithmic to the architecture of machines (instruction set, interconnected network, geometry of processors and the management of the input-output), observed along this study.

REFERENCES

1. Hwangs, B., 1984. Computer Architecture and Parallel: Processing. McGraw Hill.
2. Serra, J., 1998. Image Analysis and Mathematical Morphology. Academic Press London .
3. Soila, P., 1999. Morphological Image: Analysis. Springer Verlag, Berlin.
4. Han, J. and P. Jonker, 2004. From massively parallel image processors to fault-tolerant nanocomputers pattern recognition. ICPR 2004. Proc. 17th Intl. Conf., 3: 2-7.
5. Kontoghiorghes, E., A. Sameh and D. Trystram, 2002. Parallel Matrix Algorithms and Applications. Elsevier.
6. Le Cun, B., S. Rajopadhye, J.L. Roch and , C. Roucairol, 2000. Algorithme parallele in: IHPerf. Applications Paralleles Hautes Performances: Analyse, Conception et Utilisation de Grappes Homogènes ou Hétérogènes de calculateurs, J.R.J, L. Pazat, S.Rajopadhye CNRS Aussois, France.
7. Csajkowski, K. and I. foster, 2002. A ressource management architecture of metacomputing system. Lectures Notes in Computer Science.
8. Boukerram, A., 1991. Morphologie mathématique et architectures parallèles en traitement d'images: implantation d'algorithmes et comparaison de performances. Ph. D. Thesis. University of L. Pasteur Strasbourg, France.
9. Aumage, A. and G. Mercier, 2003. A cluster of clusters enabled MPI implementation. In 3° IEEE/ACM Intl. Symp. Cluster Computing and the Grid, Tokyo.
10. Basile, J.L., S. Castan and J.Y. Latil, 1994. Structures Paralleles en traitement d'Images. Prem Colloque Image, Biarritz.
11. Jiming, L., J. Xialing and W. Yuanshi, 2005. Agent based load balancing on homogenious mini-grids. IEEE Trans. Parallel and Distribted System, 16: 6.
12. Foster, I. and C. Kesselman. Clobus: A Metacomputing Infrastructure Toolkit. http://www.globus.org/