MASSIVELY PARALLEL PROCESSING APPROACH TO FRACTAL IMAGE COMPRESSION

Paolo Palazzari

ENEA - HPCN Project - C.R.CasacciaVia Anguillarese, 301 - 00060 S.Maria di Galeria (Rome)
E-mail palazzari@casaccia.enea.it (Tel ++39-6-3048 3167)

Moreno Coli

University "La Sapienza", Electronic Engineering Department,Via Eudossiana, 18 - 00184 Rome (Italy)
E-mail coli@die.ing.uniroma1.it

ABSTRACT

In the last years Image Fractal Compression techniques (IFS) have gained ever more interest because of their capability to achieve high compression ratios while maintaining very good quality for the reconstructed image. The main drawback of such techniques is the very high computing time needed to determine the compressed code. In this paper, after a brief description of the IFS theory, we discuss its parallel implementation by comparing the different level of exploitable parallelism. In the paper we show that Massively Parallel Processing on SIMD machines is the best way to use all the large granularity parallelism present in this problem. Finally, we give some results achieved implementing IFS compression technique on the MPP APE100/Quadrics machine.

1. INTRODUCTION

Image compression fractal techniques were introduced by Barnsley [Bar 88]. The image is represented through a piecewise linear contractive function F and is reconstructed by iteratively applying F to a randomly chosen starting image: this technique is called Iterated Function System (IFS). Compression is achieved exploiting as much as possible the autosimilarities in the image. IFS has been widely used (see, for example, [Bar 88], [Jaq 92], [Mon 92a,b]) because of the high compression rates achievable.

The main drawback of IFS is the very high computing time needed in the compression phase (i.e. the solution of the so-called IFS inverse problem): in fact, a NxN image is partitioned into nxn blocks (called range blocks R_i) and, for each block R_i, the contractive function w_i and the 2nx2n blocks D_k which minimize $\|R_i - w_i(D_k)\|$ are searched. For example, the determination of the IFS code for a 512x512 image with 8x8 range blocks requires about 325×10^9 floating point operations(flops): such a very huge number of flops clearly explains the complexity of the exact solution of the IFS inverse problem.

In order to reduce the amount of computations required, some sub-optimal techniques were proposed. In [Mon 92a] and [Hur 93], for example, the search is not performed over the whole space (the domain pool, DP) but in a small subset of DP (determined through a neighbourhood basis). An alternative way to reduce DP is classification [Jaq 93] [Jac 92]: in such a case dimensions of DP are reduced by introducing in it only elements belonging to a certain class. A completely different approach to speedup the coding phase is the Nearest-neighbour search [Sau 96a], [Kom 95], based on the interpretation of the coding nxn sub-images as vectors of a nxn Euclidean space and on a logarithmic searching procedure defined on it; in such a case we cannot take into account the effect of quantization on the coefficient parameters. Speedup procedures can also use the fast convolution method suggested in [Sau 96b].

In this paper we show that massively parallel processing is a feasible and practical way to afford the exact solution of the IFS inverse problem; some experimental results, derived from the implementation of the IFS algorithm on the SIMD Quadrics machine (512 processors, peak computational power 25.6 Gflops) are given.

2. IFS : THEORETICAL BASES

A M x N digital image with L grey levels is represented through the function

$$z = \Phi(x,y),\ x \in [0,N-1],\ y \in [0,M-1],\ z \in [0,L-1].$$

Image domain is defined as R=[0,N-1]x[0,M-1] and can be partitioned into several subdomains R_i. We define the space of the digital images as the space Y of all the functions Φ; a metric function $d(\phi,\xi)$ measures the distance between the images $\phi,\xi \in Y$; (Y,d) is a complete metric space.

A contractive transformation on a complete metric space (Y,d) is defined as a function $F: Y \rightarrow Y$ such as

$d(F(\phi),F(\xi)) \leq s\, d(\phi,\xi)$ for each $\phi,\xi \in Y$, being $0 \leq s < 1$. The following two theorems subsist [Bar 88]:

contractive function theorem: a contractive function F defined on a complete metric space (Y,d) has one, and only one, fixed point Ω such that $F(\Omega)=\Omega$. If we consider the

sequence $\{\xi_0,\xi_1,...,\xi_k\}$, being $\xi_i=F(\xi_{i-1})$, it results that

$$\lim_{k\to\infty} \xi_k = \Omega \quad \text{for each } \xi_0 \in Y.$$

Furthermore, if a constant $C\in\mathbf{R}$ exists such that $d(\phi,\xi)<C$ for each $\phi,\xi\in Y$, it results that $d(\xi_n,\Omega)\le s^n C$.

collage theorem: for a given contractive function F defined on a complete metric space (Y,d), the distance between the fixed point Ω and an image $\xi\in Y$ is limited by the following

$$d(\Omega,\xi) \le (1-s)^{-1} d(F(\xi),\xi).$$

Contractive functions theorem ensures the existence and the unicity of an attractor Ω for the contractive function F. The image Ω can be obtained through the iterated application of F to any starting image $\xi_0\in Y$; the sequence $\{\xi_0,\xi_1,...,\xi_k\}$ generated by the iterated application of F is an IFS (Iterated Function System) which converges to Ω.

Collage theorem is useful when we want to solve the inverse problem, i.e. when we want to find the contractive function F which has a given image ξ as attractor. In fact, by testing $d(F(\xi),\xi)$, we have a measure of the distance between the image ξ and the actual attractor of F. ξ is an attractor of F only if $F(\xi)=\xi$.

As F is a contractive function (with $s<1$) it can result $F(\xi)=\xi$ only if F is the union of k contractive functions f_i (i=1,2,...,k), that is if $F(\xi) = f_1(\xi)\cup f_2(\xi)\cup...\cup f_k(\xi)$.

Each f_i has contractivity s_i; contractivity of F is given by $s=\max(s_i)$. Usually $f_i(\xi)$ is a local function, i.e. f_i is applied to a (nxn) subdomain $\xi_i = R_i$ x[0,L-1]. $f_i(\xi_i)$ is the composition of two functions and is defined as it follows : $f_i(\xi_i) = W_i(A(\xi_i))$, being

- $A(\xi_i)$ a function which maps, through an averaging operation, a (2nx2n) subdomain ξ_i into a (nxn) subdomain ξ_i';
- $W_i(\xi_i') = \bigcup_{(x,y)\in\xi_i'} w(x,y,z=\Phi(x,y)) =$

$$\bigcup_{(x,y)\in\xi_i'} \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & \alpha_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ \beta_i \end{bmatrix}$$

The contractive function W_i can be thought as the composition of two different functions: a spatial affine transformation $\left(\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}\right)$ and a linear luminance transformation $(z' = \alpha_i z + \beta_i)$ ($\alpha_i<1$). As A is a fixed function, the coding of an image is given by the coefficients of the W_i functions.

3. IFS IMPLEMENTATION AND COMPLEXITY EVALUATION

On the base of the theory explained before, solution of the inverse IFS problem is described through the following algorithm:

1. partition the image into blocks of size nxn; these blocks are called Range Blocks (RB);
2. individuate the set containing all the possible 2nx2n blocks defined over the image ξ (or a subset of it if a sub-optimal solution is acceptable); this set is called Domain Pool (DP) and the blocks are called Domain Blocks (DB)
3. for each RB, extract from DP a DB; reduce DB to a nxn block through an averaging operation, then apply to it one of the 8 isometric transformations (identity, reflection about x and y axis, about first and second diagonals, +90°,+180°,+270° rotations around the center) and transform the so obtained block through the luminance transformation; the α and β coefficients for the transformation are computed solving the linear system obtained by setting to zero the derivative with respect to α and β of the rms error, i.e.

$$\begin{cases} \dfrac{\partial}{\partial\alpha}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\left(RB_{ij} - (\alpha DB_{ij} + \beta)\right)^2\right) = 0 \\ \dfrac{\partial}{\partial\beta}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\left(RB_{ij} - (\alpha DB_{ij} + \beta)\right)^2\right) = 0 \end{cases}$$

which give

$$\alpha = \frac{n^2\sum_{i=1}^{n}\sum_{j=1}^{n}(RB_{ij}DB_{ij}) - \sum_{i=1}^{n}\sum_{j=1}^{n}(RB_{ij})\sum_{i=1}^{n}\sum_{j=1}^{n}(DB_{ij})}{n^2\sum_{i=1}^{n}\sum_{j=1}^{n}(DB_{ij})^2 - \left(\sum_{i=1}^{n}\sum_{j=1}^{n}(DB_{ij})\right)^2} \quad (1)$$

$$\beta = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(RB_{ij}) - \alpha\sum_{i=1}^{n}\sum_{j=1}^{n}(DB_{ij})}{n^2} \quad (2)$$

4. for all the possible pairs of RBs and DBs, evaluate the rms error defined as

$$d_{rms} = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(RB_{ij} - (\alpha DB_{ij} + \beta)\right)^2 \quad (3)$$

and, for each RB, save the coordinates of the DB, the isometric transformation and the α and β coefficients which minimise d_{rms}.

As step 4 of previous algorithm must be repeated for all the RBs and, for each RB, for all the DBs, in the case of a NxN digital image it must be repeated a number of times

$$N_{step} = 8\left(\frac{N}{n}\right)^2 (N-2n)^2$$

being 8 the number of isometric transformations, $\left(\frac{N}{n}\right)^2$ and $(N-2n)^2$ the number, respectively, of RBs and DBs.

By inspecting expressions (1) and (2), we see that only the term $\sum_{i=1}^{n}\sum_{i=1}^{n}\left(RB_{ij}DB_{ij}\right)$ in (1) must be computed; in fact, all the other summations $\sum_{i=1}^{n}\sum_{i=1}^{n}X_{ij}$ can be pre-computed in the initialization phase (i.e. outside the critical loop). If we neglect initialization phase, computation of (1) requires $2n^2+5$ floating point operations (flops) and (2) requires 3 flops; furthermore, on the base of step 4, for each pair (DB,RB) we have to compute d_{rms} and such a computation involves $5n^2$ flops. Consequently, IFS algorithm requires a number of flops given by

$$N\_flops = 8\left(\frac{N}{n}\right)^2 (N-2n)^2\left(7n^2+8\right) \qquad (4)$$

Even if *N_flops* could be diminished either by adopting some more auxiliary storage for pre-computed data or by lowering dimensions of DP, *N_flops* remains still too large to be afforded with conventional computers without occurring in very long compressing times or in too noisy reconstructed images.

On the base of previous reasoning, Massively Parallel Processing (MPP) seems to be a reasonable answer to the IFS coding problem, allowing the achievement of both fast compression times and high quality reconstructed images.

4. PARALLELIZATION STRATEGIES

Usually, when we want to exploit the parallelism present in a problem that we are solving, we can individuate several types of parallelism at different levels of granularity, being the granularity a (machine dependent) measure given by the ratio between the computing and the communication times involved in the parallel algorithm; in the IFS coding procedure, for example, we can individuate at least 4 level of parallelism:

1- Hardware Level Parallelism (HWLP): this kind of parallelism is present in almost all the applications and is incorporated in all the microprocessors constituting the computing devices (pipelined RISC processors, vector processors, superscalar CISC processors, VLIW processors); we do not discuss further HWLP because it is a feature of the computer and does not affect programming style and higher level parallelization strategies (usually optimization concerning HWLP are demanded to the compiler);

2- Low Level Parallelism (LLP): if we look at expressions (1) and (3), we see that we can distribute RB_{ij} and DB_{ij} among several processors (not more than n^2 processors); if we have pxp processors ($p \le n$), $\frac{n}{p} x \frac{n}{p}$ values are assigned to each processor and we can compute the first term of expression (1) with $\frac{n}{p} x \frac{n}{p}$ multiplication and summation steps plus $\log_2(pxp)$ communication and summation steps (tree summation scheme); by assuming multiplication time equal to summation time equal to t_{exe} and communication time equal to t_{com}, we have that the global time to compute the first term of expression (1) is given by

$$T_{(1)P} = 2\left(\frac{n}{p}\right)^2 t_{exe} + \left(t_{exe} + t_{com}\right)\log_2(pxp);$$

on a similar single processor, the time to compute the same expression would be

$$T_{(1)S} = 2n^2 t_{exe};$$

clearly LLP is useful when $T_{(1)P} < T_{(1)S}$; the greater is t_{exe} with respect to t_{com}, the bigger is the speedup achievable; for example, in the case of $n=p$, we obtained a Speedup

$$S = \frac{T_{(1)S}}{T_{(1)P}} = \frac{2n^2 t_{exe}}{2t_{exe} + 2\left(t_{exe} + t_{com}\right)\log_2(n)};$$

it is clear that the increasing of the granularity (i.e. of t_{com} with respect of t_{exe}) makes S to become smaller. Similar considerations could be derived for the parallelization of expression (3).

3- Medium Level Parallelism (MLP): in such a case we can partition DP among p processors, performing in each processor, in a sequential way, the computation of expressions (1), (2) and (3) between each RB_{ij} and the DBs belonging to the subset of DP assigned to the processors; after this phase, which is p times faster than the sequential computation, we need a communication phase to determine the DB which gives the less rms error; again, this communication phase requires $\log_2 p$ comparisons and communication steps. If we indicate we Ts the sequential time to find in DP the DB which minimises the rms error, we can derive for the speedup S the following expression:

$$S = \frac{Ts}{\frac{Ts}{p} + \left(t_{exe} + t_{com}\right)\log_2 p};$$

also in this case we see that S diminishes when the granularity of the process increases.

4- High Level Parallelism (HLP): in such a case the image is partitioned among the p processors,

distributing equally the RBs through all the processors; the whole DP is contained in (or accessible from) each processor. The algorithm in the HLP case is the following:

do in parallel in each processor p_i
  do for each RB assigned to p_i
    find the DB and the affine contractive transformation which minimise the given distorsion measure (step 3 and 4 of the IFS algorithm)
  enddo for each
enddo in parallel

In such a case it is easy to verify that no communications are required (we have the so called embarrassing parallelism) and that the Speedup S is exactly equal to the number of processors *p*.

From previous reasoning, we can say that HLP is the best approach for the parallel IFS coding; in fact MLP and LLP present some communication overheads that give rise to poorer performances. In general we can say that HLP is the best approach to parallelism; MLP and LLP are taken into account only when the HLP is not sufficient to use all the processors available.

The adoption of HLP in the IFS case, for a NxN image partitioned into *nxn* RBs, allows us to keep busy $\left(\frac{N}{n}\right)^2$ processors; so, for the typical case N=512 and *n*=8, we can effectively use a number of processors as large as 4096. This very large number of processors makes the IFS coding problem a good candidate for efficient utilisation of Massively Parallel Processors.

5. CHOSING THE PARALLEL ARCHITECTURE

In the last years a lot of MPP architecture were proposed (CM-5, Cray T3D(E), IBM SP/2, Meiko CS-2, APE100/Quadrics); all these machines provide configurations with hundreds or thousands of processors and support HWLP.

Following the classical Flynn classification, we can divide parallel machine into two general classes: SIMD and MIMD machines. The first ones have only one control processor which broadcast the same instruction to all the Processing Elements, while the second ones are built with general purpose processors connected through fast interconnection networks. Clearly MIMD machines are more general (and more costly) than SIMD machines, because their HW supports the contemporaneous and asynchronous execution of different processes; in particular each processor in MIMD machines has HW to manage the control flow of the processes, the caching policy and asynchronous communication.

In the case of IFS coding, the analysis of the algorithm at the HLP shows that:

- it is completely synchronous;
- all the processors execute the same instructions on different data (RBs);
- no interprocessor communication is required;
- no caching policy is needed.

Previous points make clear that MIMD machines are over-dimensioned for such a coding problem; in particular they have HW resources that remain almost unutilised while all the HW (and consequently all the computational power) of SIMD machines is fully exploited. As a consequence, we chose to implement IFS coding on the APE100/Quadrics SIMD machine [Bar 93].

The APE100/Quadrics machine was originally designed as a QCD machine and, successively, transformed into a more flexible machine by substantially improving its I/O capabilities. It is constituted by VLIW pipelined processors which have a peak performance of 50 Mflops. The machine is connected according to a 3-dimensional toroidal topology and, in its largest configuration, uses 2048 processors giving a peak power of 100 Gflops. The machine is hosted by a Sparc20 and I/O is performed through an HiPPI channel (High Performance Periferal Interface) which offers a bandwidth of 20 MByte/sec. Following the idea that MIMD machines are not well suited for MPP, being usually HLP better implemented on SIMD machines, the APE100/Quadrics will be updated by one year with the APEmille machine, a SIMD MPP with peak power of 1 Tflops.

6. RESULTS

As discussed in section 4, we have implemented the IFS coding procedure at the HLP on the SIMD massively parallel processing APE100/Quadrics machine. We used the configuration with 512 floating point processors, offering a peak power of 25.6 Gflops.

As test case we compressed the 512x512 image Lenna, using 8x8 range blocks. The compression time, with DP step set to 2 (see fig. 1), was 11.2 seconds: the sustained computational power was 7.3 Gflops, i.e. we achieved a parallel efficiency in the machine utilisation E=0.28. In this case we had a compression ratio CR=18.3 and SNR=30.3 dB. When we diminished the size of DP, by using step=8 to build it, we obtained the compressed image in 0.7 seconds; in this case CR=21.3 and SNR=30 dB (fig. 2).

In both the examples we used 3 bits to represent α and 8 bits to represent β.

We also tested the method on a 128 processor machine; in such a case we obtained compression times exactly 4 times greater than the ones in previous examples; this fact demonstrates the very good scalability of the algorithm.

7. CONCLUSIONS

After recalling IFS theory and its advantages (high compression ratios, very good quality) and drawbacks (very high computing times), we gave a quantitative measure of the complexity of the IFS coding procedure.

Because of this very high computing complexity, we suggested to use already consolidated parallel processing architectures to obtain fast compressing times without loss in the quality of the reconstructed image.

We analysed the parallelism inherent to this problem at 4 different levels of granularity and we showed that we have enough High Level Parallelism (HLP) to keep efficiently busy a massively parallel processor.

Finally, we have briefly discussed the target architecture on which implement the algorithm, arguing that SIMD machines are better suited for such kind of problems than MIMD machines.

The results we obtained from an actual implementation of the method on the APE100/Quadrics SIMD supercomputer demonstrate that MPP is an affordable way to have fast compressing times avoiding quality degradation introduced by the usual fast fractal compressing techniques.

fig. 1

fig. 2

REFERENCES

[Bar 88] Barnsley,M.F. : 'Fractals everywhere'. New York : Academic Press, 1988.

[Bar 93] Bartoloni,A. et al: 'A hardware implementation of theAPE100 architecture'. International Journal of Modern Physics, C4, 1993.

[Jaq 92] Jaquin,A.E. : 'Image coding based on fractal theory of iterated contractive image transformations'. IEEE Trans. On Image Processing, vol. 1, n. 1, Jan. 1992.

[Jaq 93] Jaquin,A.E. : ' Fractal Image coding: a review'. Proc. Of the IEEE, vol. 81, n. 10, 1993.

[Mon 92a] Monro, D.M., Dudbridge,F. : 'Fractal block coding of images'. Electronic Letters, vol. 28, n. 11, May 1992.

[Mon 92b] Monro, D.M., Dudbridge,F. : 'Fractal approximation of image blocks'. Proc. ICASSP 3, 1992.

[Hur 93] Hurtgen,B., Stiller,C. : 'Fast hierarchical codebook search for fractal coding of still images'. EOS/SPIE Visual Communication and PACS for Medical Applications 93, 1993.

[Jac 92] Jacobs,E.W., Fisher,Y., Boss,R.D. : 'Image compression: a study of the iterated transform method'. Signal Processing, 29, 1992.

[Sau 96a] Saupe,D. : 'Fractal image compression via nearest neighbor search'. In Proc. NATO ASI Fractal Image Encoding and Analysis. Springer-verlag, 1996

[Sau 96b] Saupe,D. Hartenstein,H. : 'Lossless acceleration of fractal image compression by fast convolution'. Proc. Of the IEEE International Conference on Image Processing (ICIP 96), 1996.

[Kom 95] Kominek,J. : 'Algorithm for fast fractal image compression'. Proc. Of SPIE, vol. 2419, 1995.