# Feasibility evaluation of image retrieval parallel implementation

A. Amato[1], T. Delvecchio[1], V. Di Lecce[2], D. Gagliardi[1], A. Giancaspro[3], A. Guerriero[2], G. Milillo[4]
[1]Politecnico di Bari - Viale del Turismo n.8 - 74100 Taranto Italy
[2]DEE- Politecnico di Bari - Via Re David 200 - 70121 Bari - Italy
[3]Telespazio- Localita' Terlecchia - 75100 Matera - Italy
[4]Agenzia Spaziale Italiana - Centro di Geodesia Spaziale "G. Colombo" - 75100 Matera - Italy

*Abstract*: - Parallel programming is very interesting today. Many employment can be individuate for this technology. For example in physical processes modeling or in the simulation of cardiac dynamics. It is clear that heavy load computational applications are appropriate for test parallel programming and parallel architectures. In the Polytechnic of Bari our research group, cooperating with Italian Space Agency and Telespazio, are investigating on parallel application in remote sensing image retrieval for the research project ASI-PQE2000[1].

*Key-Words*: Parallel Programming, Skeleton, Image Database, Sorting, Image Features.

## 1 Introduction

Due to the technological progress in parallel processing systems and their increasing availability for scientific and commercial applications, the appropriate development of optimum parallel programs is a challenging task for software engineers of parallel systems.

Besides other aspects, the objective to design and implement parallel programs showing optimum performance when being executed on a parallel processing system, is the most crucial one in the whole development cycle.

Within the parallel processing activities, the major goal is to establish a software development methodology performance oriented. The methodology is based on high level, graphical, hierarchical parallel program specifications supporting predictability of the execution behavior. Tools under development in Pisa University (research on "Architectures and Programming Tools for High Performance Computing" at University of Pisa under the guide of professor Marco Vanneschi) will provide possibilities of expressing the 'algorithmic idea' of a parallel application in a graphical fashion, focusing the communication behavior of the program. Performance predictions based on this application skeleton should help to make performance efficient design choices ahead of the full implementation.

It is clear that parallel processing is ideal for high load computational application. According to ASI[2] and Telespazio, and starting from previous research activity of the authors of this paper, it has been selected the image data-base retrieval as application for parallel implementation.

Considerable advances in information technology, broadband networks, high-powered workstations, and data compression standards have made it possible to realize huge image databases. The main problem in the creation and management of large image databases regards the modality of efficient image indexing.

Traditional indexing by keywords, due to the high cost and the dependence on operator's vocabulary should be replaced by more efficient automatic indexing techniques. The majority of the techniques presented in literature works on pictorial images.

During these years our research group was addressed to the study of new automatic indexing techniques and image retrieval in pictorial image database.

Image features commonly used to describe the image syntactical content in automatic processing (no human assistance) are color, shape, texture, brightness, angular spectrum distribution, etc.

Recently, some indexing and retrieval techniques was applied on geographic database with interesting performance on satellite images, including Thematic Mapper (TM), MultiSpectral Scanner (MSS), Synthetic Aperture Radar (SAR) and satellite photos[5,6].

According to the previous remarks it has been started the rewriting stage of tested algorithms, written for Matlab environment. A software timing simulator has been developed in simulink-matlab to evaluate the parallel implementation performance.

This paper is organized as follow, in section two the parallel programming style and tools are discussed, in section three the features extraction techniques used to index the database are summarized, the image database retrieval system is shown in section four, the parallel architecture, simulators and experimental data are described in section five. In section six we conclude with the future objects of our research.

## 2 Parallel programming style and tools

There are two large classes of parallel programs: programs whose specifications describe ongoing

[1] under the grant n. 2001211843 of Telespazio S.p.A.
[2] Italian Space Agency

behavior and interaction with an environment, and programs whose specifications describe the relation between initial and final states.

A low impact approach is in initially development of primary programming model that combines the standard sequential model with a restricted form of parallel composition that is semantically equivalent to sequential composition. This programs can be reasoned about using sequential techniques and executed sequentially for testing.

They are then transformed for execution on typical parallel architectures via a sequence of semantics-preserving transformations, making use of programming models (composition of stream parallel skeletons such as pipelines and farms).

The transformation process for a particular program is typically guided and assisted by tools that predict or evaluate performance of the final parallel program and its matching on the parallel architecture.

Transformations may be applied manually or via a parallelizing compiler. It is clear that the real (operative) problem is in the looped/unbalanced cases. In these cases an expansive tuning process can be performed.

A well tested strategy is the use of composition of stream parallel skeletons such as pipelines and farms [23].

By looking at the ideal performance figures assumed to hold for these skeletons, [16] shows that any stream parallel skeleton composition can always be rewritten into an equivalent "normal form" skeleton composition, delivering a service time which is equal or even better to the service time of the original skeleton composition, and achieving a better utilization of the processors used.

Skeleton based programming models represent an interesting research field in parallel programming models. Cole introduced the skeleton concept in the late 80's [15]. Cole's skeletons represented parallelism exploitation patterns that can be used to model common parallel applications. Many researchers investigated skeletons as constructs of an explicitly parallel programming language [17, 18, 19,20].

In general a skeleton can be understood as a higher order function taking one or more other skeletons or portions of sequential code as parameters, and modeling a parallel computation out of them. The skeleton can be applied therefore to different parallel applications, all exploiting the same kind of parallelism: the one encapsulated by the skeleton used.

In order to implement parallelism using skeletons on parallel architectures, compiling tools based on the concept of the implementation template have been developed [15, 21, 22]. Therefore, the process of generating parallel code from a skeleton source code can be performed by selecting the skeleton matching the source code and deriving a corresponding set of implementation templates [15].

It is clear that different rewriting techniques/style can been developed that allow skeleton programs to be transformed/rewritten into equivalent ones, achieving different performance when implemented on the target architecture [17].

The rewriting can be also driven by analytical and/or heuristic performance models associated with the implementation templates of the skeletons.

In our test an MPI based parallel architecture is used to evaluate the performance of rewritten sequential Matlab and C code in SkIEcl [8, 13] and ASSIST [12] environments

# 3 Retrieval algorithms

Aim of this experience is the evaluation of parallel implementation of tested [1-5] retrieval algorithm based on image features like: angular spectrum, color histogram, Hough transform, Gabor transform, pattern directionality, local brightness and roughness.

In the following the features extraction algorithms are shortly presented as basic functions. A complete presentation of algorithms and performance can be found in [6,7].

## 3.1 Angular spectrum module

Raster image→ 0.1 filtering (convolution) → resampling →2D Fourier transform → Cartesian/polar spectrum transformation → application of a boundary threshold and "cleaning of the spectrum" → reading by band of the angular spectrum → signature generation.

## 3.2 Hough transform

Raster image → 0.1 filtering (convolution) → resampling → finding edge with Canny algorithm→ computing Hough transform → signature generation.

## 3.3 Colors-brightness histogram

Raster image→ color space transform from RGB to CEILAB → color component separation of color histogram computing (for every component) → signature generation.

## 3.4 Local directionality

Raster image→ 0.1 filtering (convolution) → undersampling→ 25 sub image extraction →
the following steps are repeated 25 times:
$|\Delta G| = (|\Delta_H| + |\Delta_V|)/2$ is calculated, where $\Delta_H$ and $\Delta_V$ represent Sobel operator 3*3 (vertical and horizontal)→ $\theta = tan^{-1}(\Delta_V / \Delta_H) + \pi/2$ is calculated → thresholding ,
→ signature generation.

### 3.5 Local brightness

Raster image→ 0.1 filtration Fn (convolution) → undersampling → 25 sub images extraction →
the following step is repeated 25 times:
mean local brightness determination,
for each subimage a 16 level quantization is applied → signature generation.

### 3.6 Roughness

Raster image → 0.1 filtering (convolution) --> undersampling → 25 sub image extraction →
the following step are repeated for 25 times :
local roughness determination in accordance with Tamura [5],
→ signature generation.

## 4 Image database retrieval system

All retrieval system are based on the two following steps:

- Signature generation as just see in the previous paragraph shown in figure 1.
- Image Retrieval where a metric that allows comparison of different signatures (that of the query as opposed to those of the database images), is used. The results ad sorted from similar to dis-similar, as shown in figure 2.

It is clear that in a large database the sorting procedure can be represent a large computational load. Although sorting is one of the more studied information theory problems, it has been characterized by a high interest in parallel programming.

There are several algorithms for the sorting.:dicotomic, shellsort, quicksort, sample sort. All the sorting algorithms are based on the concept to divide the list to order in touched sub-lists (which have about the same number of elements) when processors ($2^d$) constituent parallel machine on which operates. Each processor will order the sub-list of his competence (in this method ~~are~~ $2^d$ locate order lists are obtained). Successively, it is necessary to decide how make a merge of ~~this~~ these lists to ~~obtaine~~obtain only one ordered list.

~~5. Sorting~~This goal is reached though some comparison-exchange passages. For each passage, two contiguous processor exchange some elements and reorder the obtained lists. Repeating these passages, when the end of process is reached, every calculation-node has got an ordered list and all the elements in a sub-list are more greater than the ones of another.

Parallel sorting algorithm efficiency and its applicability are function at least of this parameters:

- number of elements that constitute the list to order and their dimension;
- number of processors and parallel architectures;

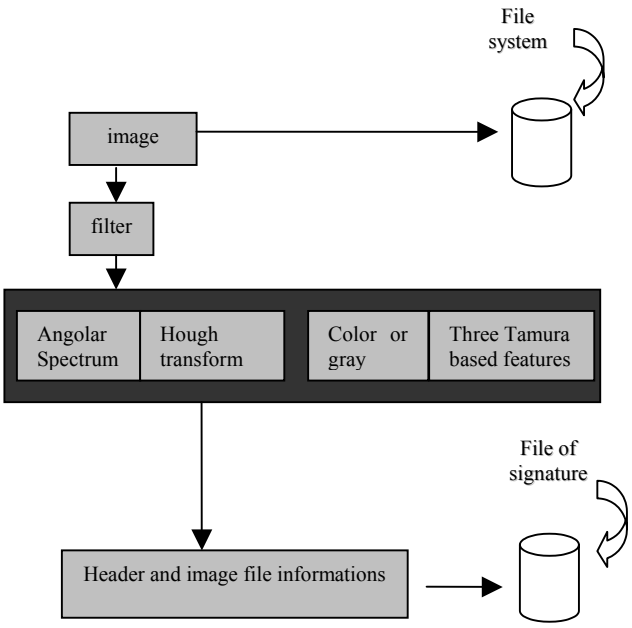


Fig. 1 schematic diagram of the database population phase

- starting distribution of the numeric values (i.e. if the list to order have some particular propriety like a known distribution or also contains many redundant elements, then particular algorithms can be more efficient than other);
- starting allocation of the data in memory (~~This~~ this factor may have a notable impact on the efficiency of the sorting algorithm, i.e. in the sample sorting, ~~very~~ better performances are obtained according to which method is implemented to obtain the *splitting keys*: an optimal selection of these brings a better load balancing among the various processors).

An interesting thing is that sorting procedure have become a used benchmark to test parallel machine performances; in fact they have , to such intention the following features:

- they are a simply describable problem and are well known in literature;
- their dimension can be easily escaladed to make a progressively more binding test;
- kindly, are used to tests exchange and the matching of great number of data skills of the systems.

Parallel sorting, may be used to test the efficiency of the parallel environment~~first part of this research.~~ ~~It~~ it is possible to find a relationship between the number of steps (Flops) needed to have an ordered list and the algorithm execution time.

Basic idea ~~for~~ of our performance evaluation ~~of proposed~~ approach is ~~to~~ the comparison~~e~~ of the sorting time to the signatures extraction time.

All experimental activities are in progress using the computing resources of the Taranto Engineering Faculty.

## 5 The Parallel Machine

A parallel machine formed by 8+1 workstation based on 1 GHz Athlon CPU that support MPI/Linux (S.O. Red Hat Linux 7.1 with LAM on MPI 6.5.1) has been assembled. The array structure is shown in figure 3.

In the current evaluation all the algorithms are converted in C language. The test of these procedures shows the correspondence of result with the previous version in Matlab.

Classical way in MPI based parallel implementation development, is manual balancing of the computational load; in this test the working group used the Vanneschi and Danelutto approach [8-9] based on the tools useful in the predictive evaluation (P3L, Skie) [10,11] or automatic load balancing [12].

Waiting for ASSIST delivery, to find the best skeletons composition (characterized by the lowest work time as to hardware resources), a timing

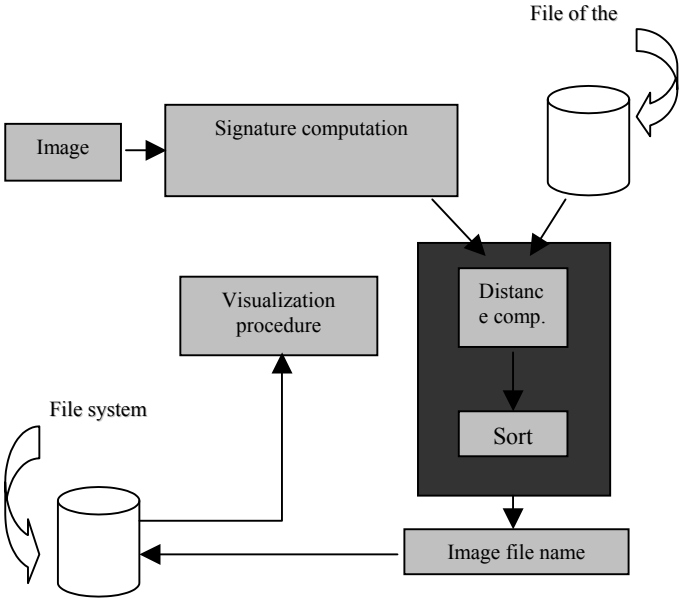


Fig.2 Query result sorting phase

simulator for parallel machine, based on Simulink environment has been developed .

A profiling of the database signatures population has been obtained under Matlab and C environment. Data so obtained represent the time (flops) of each procedures in figure 1, these values are characterized by a high variance due to algorithms that are used in

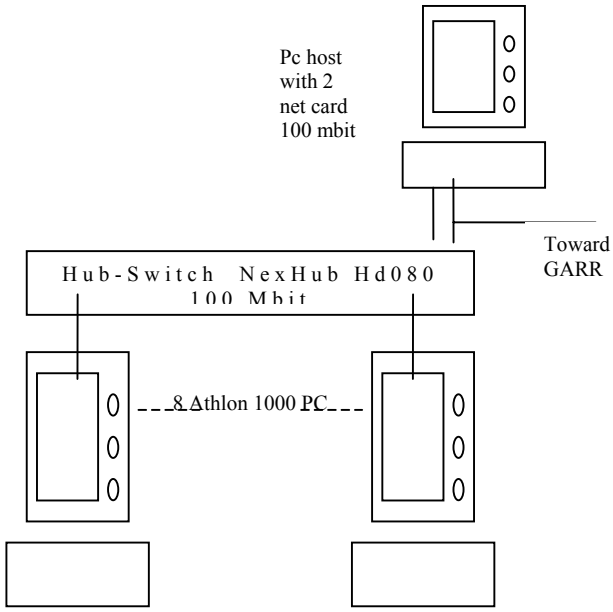


Fig. 3. The parallel machine

this application, i.e. Hough transform uses the Canny filter which computing time depend on image. These values are input for the skeleton evaluation (the skeleton farm is represented in figure 4).

Table 1 shows the performance of different skeletons composition.

| Skeleton | M Flops |
|---|---|
| Farm with 8 processors | 1.2147e+010 |
| Farm-Loop-Seq. with 8 processors | 1.0885e+010 |
| Pipe with 11 processors | 4.5239e+010 |
| Sequential on 1 processor | 7.8410e+010 |

Table 1. Skeletons performance

Results show that the 8 processors farm with a nested loop that work sequentially on a set of 1/8 of the whole number of the input images, is the best solution. This solution reduces the variance of computing times.

Simulink, is not the only practicable way (Pelagatti, Zavanella [14]), yet it is a valid way to find the best skeleton composition, at least till the ASSIST (Vanneschi [12]) will be implemented.

Using ASSIST tools, there will be a bevel difference between data-parallelism and stream parallelism [12]; the programs will be like "graph Skie" with stream skeletons (pipe, farm, loop) serial or parallel.

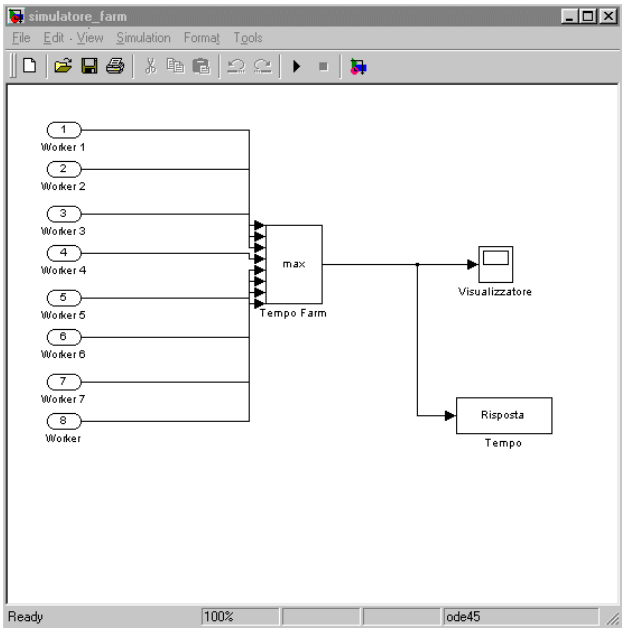


Fig. 4 The skeleton farm

## 6 Conclusions

In this paper, we have discussed the porting problem of a sequential application of image retrieval in a parallel environment using the new paradigm of programming introduced by born of new structured program languages (ski,p3l,Assist). We have evaluated several skeletons composition to optimize the performance of our application using a skeleton simulator. The results obtained show that the best solution is a farm with 8 processors with a nested loop that work sequentially on a set of 1/8 of the whole number of the input images. This solution, reducing the variance of the computing times, is 10% better than farm with 8 processors (every one work on a single image). The pipe solution seem be the worst respect all other skeleton except the sequential one. Of course this results are function of the "mapping" of our specific problem.

In these evaluation, the communication times have been ignored because it is negligible compared to computing times [12].

The Gabor-based signature are under developement and the C version of the software is under test.

*References:*

[1] A. Celentano , V. Di Lecce "A FFT based technique for image signature generation".Proceedings on Storage and Retrieval for Image and Video Database V SPIE Vol.3022 13-14 February 1997

[2] P.V.C. Hough, Method and Means for Recognizing Complex Patterns, U.S. Patent 3069654, December 18, 1962

[3] M. Porat and Y. Y. Zeevi "The generallzed Gabor scheme for image representation in biological and machine vision", IEEE Trans. on Patt. AnaI. and Mach. Intel., 10, 452-468 (1988)

[4] X. Wan and C.C.J. Kuo "Color analysis and quantization for image retrieval", Storage Retrieval Still Image Video Databases IV 2670, February 1996

[5] H. Tamura, S. Mori and T. Yamawaki "Textural Features Corresponding to Visual Perception", IEEE Trans. on Systems, Man and Cybernetics, Vol.8,No. 6, June 1978

[6] A. Guerriero and V. Di Lecce "An Evaluation of the Effectiveness of image Features for Image Retrieval", J. Visual Communication and Image Representation 10,351-362 (1999)

[7] Vincenzo Di Lecce, Andrea Guerriero, "A Comparative Evaluation of Retrivial Methods for Duplicate Search in Image Database", Journal of Visual Languages and Computing, Dicembre 1999, pp.105 – 120.

[8] M.Danelutto, R. Di Meglio, S. Orlando, S. Pelagatti, M.Zanneschi "A methodology for the development and the support of massively parallel programs" Technical Report TR-25/91 Università degli Studi di Pisa, Dipartimento di Informatica,Dicembre 1991.

[9] M.Danelutto, S.Pelagatti, M.Zanneschi "Parallel Programming Models based on the Restricted Computational Model Approch." CNR. TR-R/3/133 Progetto finalizzato C.N.R. "Sistem iinformatici e Calcolo Parallelo",sottoprogetto "Architetture PArallele",Roma 1994.

[10] B.Bacci, M.Danelutto, S.Pelagatti "Automatic balancing of strema-parallel computation in **P3l**" Technical Report TR-35/93, Università degli Studi di Pisa, Dipartimento di Informatica,Dicembre 1993.

[11] B.Bacci,B.Cantalupo,N.Guerrini e S.Pelagatti "L'ambiente di programmazione **P3L** nel sistema di spporto PVM su rete di calcolatori eterogenea" Technical Report: TR-14/95,Università degli Studi di Pisa, Dipartimento di Informatica,ottobre 1995.

[12] Vanneschi. Programma ASI-PQE2000 Metodologie per la Programmazione Parallela e ASSIST. Seminario Bari,13 giugno 2001.

[13] B. Bacci, B. Cantalupo, P. Pesciullesi, R. Ravazzolo, A. Riaudo, and L. Vanneschi "SKIE: user's guide (version 2.0).Technical report", QSW Ltd. Rome, Italy, December 1998.

[14] D. B. Skillicorn, M. Danelutto, S. Pelagatti, and A. Zavanella "Optimising Data-Parallel Programs Using the BSP Cost Model". In D. Pritchard and J.

Reeve, editors,Euro-Par'98 Parallel Processing, pages 698-703. Springer Verlag, 1998. LNCS No. 1470.

[15] M. Cole. Algorithmic Skeletons: Structured Management of Parallel Computations. Research Monographs in Parallel and Distributed Computing. Pitman, 1989.

[16] M. Aldinucci & M. Danelutto."Stream of parallel skeleton optimization". Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems. Mit-Boston, Usa, november, 1999.

[17] J. Darlington, A. J. Field, P.G. Harrison, P. H. J. Kelly, D. W. N. Sharp, Q. Wu, and R. L. While. Parallel Programming Using Skeleton Functions. In M. Reeve A. Bode and G. Wolf, editors, PARLE'93 Parallel Architectures and Langauges Europe. Springer Verlag, June 1993. LNCS No. 694.

[18] M. Danelutto, R. Di Meglio, S. Orlando, S. Pelagatti, and M. Vanneschi."A methodology for the development and support of massively parallel programs". Proceedings of Future Generation Computer Systems, 205-220, July 1992.

[19] H. Burkhart and S. Gutzwiller."Steps Towards Reusability and Portability in Parallel Programming". In K. M. Decker and R. M. Rehmann, editors, Programming Environments for Massively Parallel Distributed Systems, pages 147{157. Birkhauser, April 1994.

[20] M. Vanneschi. "PQE2000 HPC tools for industrial Applications". IEEE Concurrency, (4):68-73, 1998.

[21] B. Bacci, M. Danelutto, S. Orlando, S. Pelagatti, and M. Vanneschi. "P 3 L: A Structured High level programming language and its structured support". Concurrency Practice and Experience, 7(3):225-255, May 1995.

[22] S. Pelagatti. Structured Development of Parallel Programs. Taylor & Francis, 1998.

[23] S. Ciarpaglini, M. Danelutto, L. Folchi, C. Manconi, and S. Pelagatti. "ANACLETO: a template-based P3L compiler". In Proceedings of the PCW'97, 1997. Camberra, Australia.