PARSAR: A SAR PROCESSOR IMPLEMENTED IN A CLUSTER OF WORKSTATIONS

A.Martínez, F.Fraile
Remote Sensing Dep., INDRA Espacio
C/ Mar Egeo s/n. 28850-S.Fernando de Henares, SPAIN
Tlf.+34 1 396 3911. Fax+34 1 396 3912
e-mail: amar@mdr.indra-espacio.es

J.J.Mallorquí, L.Nogueira, J.Gabaldá, A.Broquetas, A.González(*)
Signal Theory & Communications Dep., U.Politecnica Catalunya
(*)Computer Architecture Dep., U.Politecnica Catalunya
C/Sor Eulalía de Anzizu s/n, Ed. D-3. 08071-Barcelona, SPAIN
Tlf.+34 3 401 7229. Fax+34 3 401 7232
e-mail:mallorqu@voltor.upc.es

ABSTRACT

A parallel SAR processor is presented in this paper. The target configuration is a cluster of UNIX workstations, available in most user sites. This fact allows to obtain an increased computing performance without the need of dedicated hardware investment.

1. INTRODUCTION

Synthetic Aperture Radar, SAR, is a remote sensing instrument capable of obtaining high resolution images of the Earth surface [1]. The goal of SAR processing is to transform the SAR raw data, or SAR signal, into an image. The generation of SAR images involves both a great amount of data and a complex focusing algorithm. These reasons make attractive the application of HPCN technology.

The objective of PARSAR project is the porting of a sequential SAR Processor to a parallel architecture. The target configuration is a cluster of UNIX workstations, allowing to exploit the benefits of parallelisation without the need of hardware investment.

This paper describes the parallelisation strategy of the processor, based on a multi-block approach using PVM as interface among processes. Some preliminary performance results are presented, along with the currently on-going activities.

2. DESCRIPTION OF DATA AND ALGORITHM

The SAR data used in the project correspond to the standard full frame scene (100 x 100 km^2) of the European ERS satellite [2]. The raw data is a matrix of 26800 lines each one consisting of 5616 samples or pixels. The pixels are complex numbers, coded in 1 + 1 bytes, and the size of the raw data file is about 300 MB. The calculations are performed in floating point format, resulting in a processing matrix of 1.2 GB. The output SAR image has 25000 lines each one consisting in 4912 samples. The pixels are complex and coded in 2 + 2 bytes, resulting in about 500 MB.

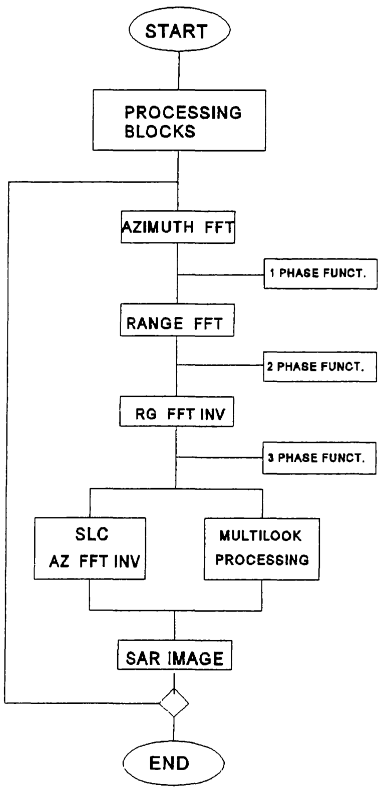

Figure 1. Flow Chart of the sequential CSA Processor

The focusing of SAR raw data is essentially a 2-D correlation of the input signal with the SAR Impulse

Response Function [3]. Classical methods of SAR processing implement the signal compression in the frequency domain. The SAR processor used in the project is called Chirp Scaling Algorithm, CSA [4]. The CSA involves only FFT and multiplications. The structure of the CSA is relatively simple, consisting on a sequence of 1-D FFT and matrix element by element products (see figure 1). The number of operations and loops is constant because the CSA does not use iterative procedures.

3. DESIGN OF THE PARALLELISATION

The parallelisation strategy that has been implemented is called Multi-Block Approach, MBA. It consists on dividing the input data into independent processing blocks; each block is fully processed in one host (child process), resulting in a small piece of the final image. MBA can be seen as a coarse grained parallelisation strategy. PVM, Parallel Virtual Machine, is used to control the whole process. PVM is a software system that permits a network of heterogeneous UNIX workstations to be used as a single large parallel computer [5].

The main advantage of MBA is the full independence between the different tasks split among the available hosts. Each child process works at his own pace and it does not need information from other children. Consequently, fast and slow workstations can coexist in the same cluster without penalizing the whole process. Another interesting point of MBA is the minimization of both the number of I/O operations and the amount of data to be transferred across the cluster.

The implementation of the MBA parallelisation is relatively easy, as the CSA processor has not to be split. Each host in the cluster has a CSA processor, very similar to the sequential one. Consequently, improvements in the sequential code are readily portable to the parallel software. On the other hand, the drawback of MBA is the correlation efficiency, that strongly depends on the size of the processing data blocks and hence, on the available RAM in each host.

The flow char of the parallel SAR processor is shown in figure 2. There is a main process, the parent process, that is is charge of launching different tasks in the computers of the network and managing the execution of them. There are three types of tasks or slaves processes:

- cutter. This task is responsible of reading the raw data file and generating the different data blocks that will be sent to the nodes.
- child. This is a simplified version of the CSA processor, that reads a data block and produces a small piece of the final image (imagette).
- builder. The builder reads the different pieces of the image and assembles the final SAR image file.

Both the raw data blocks and the imagettes generated by the processors are stored in temporary files. At first sight, the use of temporary files can be seen as a drawback, due to the increased number of I/O operations and disk usage. Nevertheless, tests conducted without temporary files showed the importance of disk access conflicts when different child processes read from and write to the same large files. Furthermore, disk operations by the child processes have to be done by direct access implying high inefficiency.

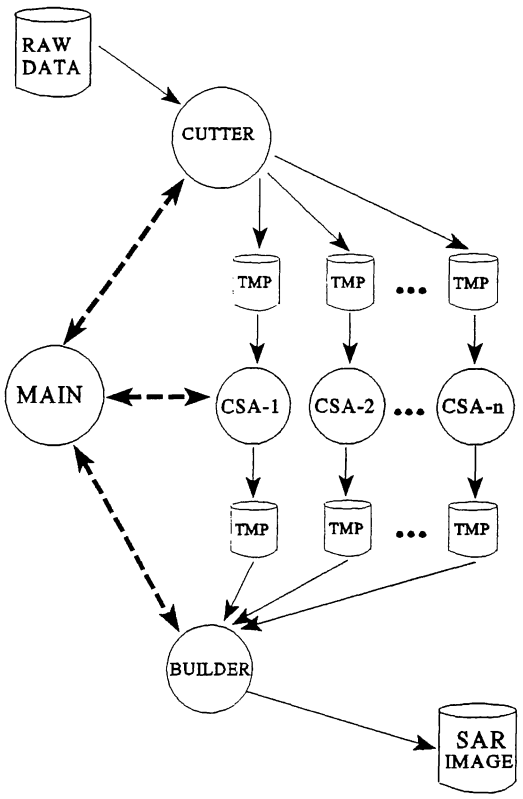

Figure 2. Flow char of the parallel processor

The parallel code starts by generating a set of data blocks that are assigned to the available hosts. The parent process continuously examine the status of the different hosts in the network. When one host is free, the parent assign a new block. The priorities in the parent process are:

- 1. Cutter process. This is needed to ensure that there are data blocks available for the child processes.
- 2. Child processes. Once a host finishes the processing of a block, the waiting time is to be minimized.
- 3. Builder process. When the imagettes corresponding to a image block are available, the corresponding part of the final file is assembled to free disk space.

With this strategy, the disk bottleneck problems are alleviated by imposing that the different slave processes never access to the same file, and most disk operations can be carried out by using the more efficient sequential access.

The number of hosts to be used in the "parallel machine" can be selected, as well as the tasks to be conducted by each node.

4. PRELIMINARY RESULTS

The classical parameters to estimate the performance of the parallel code, speed up factor (ratio of the processing time in one node to the processing time in N nodes) and efficiency (speed up factor over the number of nodes), are not readily portable for a heterogeneous cluster (each node has its processing time). We have used an alternative definition for these parameters that is intuitive and makes some sense for a heterogeneous set of computers:

- efficiency: the ratio of the number of products generated by the sequential code in a given time to the number of products generated by the sequential code running in all the computers of the cluster during the same time.
- speed up: the efficiency times the number of computers in the cluster.

The results of the parallel processor running in different hardware configurations are listed in the next tables, along with the characteristics of the workstations in the clusters and the processing time of the sequential processor in each workstation.

The first test used a block size of 32 MB (note that 2 of the computers in the cluster have 64 MB RAM). The main, cutter and builder processes were run in HP-720, so that this host is fully dedicated to data handling. The remaining two computers were in charge of SAR processing. The efficiency of the parallel code is 0.64, with a speed up factor of 1.93.

MODEL	Clock Rate	RAM	Proc. Time
HP-735	99 MHz	96 MB	150 min
HP-720	50 MHz	64 MB	320 min
HP-715	50 MHz	64 MB	320 min
CLUSTER	-	-	**120 min**

Table 1. Processing time in UPC cluster. Processing block 32 MB. Efficiency is 0.65.

The results of the tests at INDRA are presented in table 2; now, a block size of 64 MB was used, as the computers in the cluster have more available RAM. There are two test cases:

- Case 1: Only the Sun computers, 3, are used. The processing time is 58 minutes, resulting in a efficiency of 0.68, and a speed up factor of 2.03.
- Case 2: The four computers in the cluster are used. The processing time is 46 minutes, resulting in a efficiency of 0.56, and a speed up factor of 2.24.

In the two test cases, the slowest host in the cluster (Sun Sparc 10) was used to execute the main, cutter and builder processes.

MODEL	Clock Rate	RAM	Proc. Time
HP C160-L	160 MHz	128 MB	75 min
Sun Ultra 1/170	167 MHz	128 MB	75 min
Sun Sparc 20/71	75 MHz	128 MB	135 min
Sun Sparc 10/41	40 MHz	128 MB	210 min
Case 1. Sun computers	-	-	**58 min**
Case 2: All hosts	-	-	**46 min**

Table 2. Processing time in INDRA cluster. Processing block 64 MB.

The results of the tests shows that the parallel processor works relatively well in a cluster of three workstations. The efficiency figures obtained in the clusters at UPC and INDRA are equivalent. However, when including an additional host to the cluster, the efficiency decreases. This is due to the fact that the computers performing SAR processing are faster than the cutter, so that they have to wait for data blocks to process.

On-going work is currently being performed to upgrade and fine tune the performance of the parallel code. In particular, we may mention the following points:

- Optimization of the I/O operations, mainly the cutter process. This should allow the use of more workstations in the cluster without loss of efficiency.
- Allowing the processing data block size to be adjusted by each host in the cluster, so that computers with different RAM can be simultaneously used.

5. CONCLUSIONS

The first activities in the porting of a sequential SAR Processor to a parallel architecture has been performed, including the selection of the parallelisation strategy and the implementation of the first parallel prototype. The parallel software is flexible and portable, so that it can be installed in most user sites.

The preliminary results obtained with the parallel code are encouraging, and show a decrease in the processing time of the parallel code with respect to the sequential one. Good results were obtained with a cluster of three workstations.

Additional work is on-going to enhance and fine-tune the code, so that the efficiency of the parallel code can be kept constant when adding more hosts to the cluster.

6. REFERENCES

[1] Elachi C. "Spaceborne Radar Remote Sensing", IEEE Press, 1988.

[2] ESA. "ESA ERS-1 Product Specifications", ESA SP-1149, 1992.

[3] Curlander J.C. & McDonough R.N. "SAR: Systems and Signal Processing", John Wiley & Sons, 1991.

[4] Raney K. et al. "Precision SAR Processing Using Chirp Scaling", IEEE Trans. Geosci. Remote Sensing, vol. 32 pp.786-799, 1994.

[5] Geist A. et al. "PVM 3 User's Guide and Reference Manual", Report ORNL/TM-12187 (1994).

[6] Martínez A. and Marchand J.L."SAR image quality assessment", Rev. Teledetección 2 (1993) pp.12-18.

[7] Martínez A. et al. "Advanced Algorithm Techniques: Enhancement of CSA and Quicklook SAR Algorithms Final Report", ESA ESTEC Contract 3-8616/95/NL/FM (1997).

[8] Geist A. et al. "PVM 3 User's Guide and Reference Manual", report ORNL/TM-12187 (1994).

[9] Sánchez J.I. and Laur H. "ERS-1 SAR Product Validation", Proc. CEOS SAR Calibration Workshop, ESA WPP-048, (1993) pp.295-305.

7. ACKNOWLEDGEMENTS

PARSAR is a project supported by PCI-II, ESPRIT IV.

Autonomy

Chairmen: **F. Pittermann & R. Gerlich**
Dornier & Bodan System and Software Eng., Germany