# Cluster Computing Using MPI and Windows NT to Solve the Processing of Remotely Sensed Imagery

J. A. Gallud[1], J.M. García[2], and J. García-Consuegra[1]

[1] Departamento de Informática,
Universidad de Castilla-La Mancha,
Campus Universitario, 02071 Albacete, Spain
{jgallud, jdgarcia}@info-ab.uclm.es.
[2] Departamento de Ingeniería y Tecnología de Computadores,
Facultad de Informática. Universidad de Murcia, Spain,
Campus de Espinardo, 30080 Murcia
jmgarcia@ditec.um.es

**Abstract.** The design of efficient distributed applications depends on the coordinate use of different API (Application Programming Interface) like MPI and NT API's. In fact, a particular optimized code can be reused in many other applications reducing the cost of its design by means of a set of libraries. Distributed processing is applied in remote sensing in order to reduce spatial or temporal cost using the message passing paradigm. In this paper, we present a workbench called DIPORSI, developed to provide a framework for the distributed processing of Landsat images using a cluster of NT workstations. Our application is based on a NT implementation (WMPI) of the MPI standard. Thus, the large amount of time required by the sequential processes drops when the parallel processing is used. Moreover, we have obtained a reduction of computation time over the 400% for large size images and a moderate number of parallel nodes. Our results confirm that cluster computing is a cost/performance effective solution to the remotely sensed image processing.

## 1 Introduction

The available of high-speed networks and increasingly powerful commodity processors is making the usage of clusters of computers an appealing vehicle for cost effective parallel computing. Cluster computers have several advantages, as they are built using commodity-off-the-shelf hardware components and they can be programmed using the standard parallel tools and utilities. Clearly, the cluster environment is better suited to applications that are not communication-intensive, since a LAN typically has a high message start-up latencies and low bandwidths.

However, several research projects are analyzing the communication subsystem, in order to provide the same quality of message delivery as MPPs. The goal

of these projects is to reduce the message latency to the minimum, usually by eliminating the most of the operating system and device driver overheads [3, 14]

Traditionaly, parallel processing is an area that it has been dominated by Unix-based systems. That is also true in the cluster computing field. However, the computers PC-based market is clearly dominated by the Windows NT operating system. Recently, many research projects have explored the possibilities of cluster computing under Windows NT [2, 4].

Remote sensing involves the manipulation and interpretation of digital images which have been captured from remote sensors on board of satellite or aircraft systems. Such images collect information about the Earth's surface, which allow scientists to perform many environmental studies.

Remotely sensed image processing is an interesting application area for distributed computing techniques [8, 13]. The large data volumes involved and the consequent processing bottleneck may indeed reduce their effective use in many real situations, and hence the need for exploring the possibility of splitting both the data and processing over several storing and computing units [1]. All the procedures involved in the digital image manipulation of such images may be categorized into one or more of the following four broad types of computer assisted operations: image rectification and restoration, image enhancement, image classification and data merging [11].

In this paper, we describe a distributed workbench called DIPORSI, by means the coordinated use of both the MPI API and the Windows NT File System API. DIPORSI stands for *DIstributed Processing Of Remotely Sensed Imagery* and has been designed to run on a cluster of Windows NT workstations. DIPORSI was designed to perform a considerable number of the former tasks by using a cluster of workstations composed by NT platforms which are connected by means of an Ethernet network using the Message-Passing Interface standard (MPI). MPI provides an interface to design distributed applications that run on a parallel system [7].

This paper shows how reducing the long computation time required by remote sensing distributed procedures. All the distributed applications are implemented by splitting the original images into several small ones, which are processed in each node simultaneously. Moreover, remote sensed image is a good application to run in a distributed way, because it is computation-intensive with a small communication between processes. The comparative results have been obtained using a distributed algorithm to georeference a distorted remotely sensed image. This algorithm has been coded in MPI, and we have obtained a very good speedup, and a reduction in the computation time over the 400% for large images and a moderate number of nodes.

The following section explains the structure of the current DIPORSI workbench. In section 3 we present the comparative study based on a particular processing of Landsat-TM images. Finally, in section 4 the conclusions and future work are drawn.

## 2 The DIPORSI structure and its capabilities

DIPORSI workbench was designed to perform in a distributed way, a great variety of the algorithms used in the remotely sensed image processing. These algorithms work to obtain either a new digital image or new features from the original image.

Many remote sensing algorithms work in a parametric way, that is, the requested application is related to the initial parameters. The newer and most used algorithms are the classifiers and their behaviour are governed by a set of initial parameters. To obtain a classified image, the algorithm begins its computations with such initial values as yield the resultant image. This resultant image depends on both the initial parameters as well as the original multidimensional image. In many cases, the process must be repeated with other parameter values because the results are unacceptable. This explains both the need of reducing the spatial and temporal costs by means of a distributed processing, and the parametric nature of the distributed workbench.

DIPORSI appears as a layer between MPI functions and the code of each process. DIPORSI offers a set of functions and a message structure to the user for performing in an easy and distributed manner whatever remote sensing algorithm. Some recent works have proposed similar frameworks for related problems [5, 9].

In figure 1, the functional diagram of DIPORSI can be seen. Note that only one process is executed in each processor.

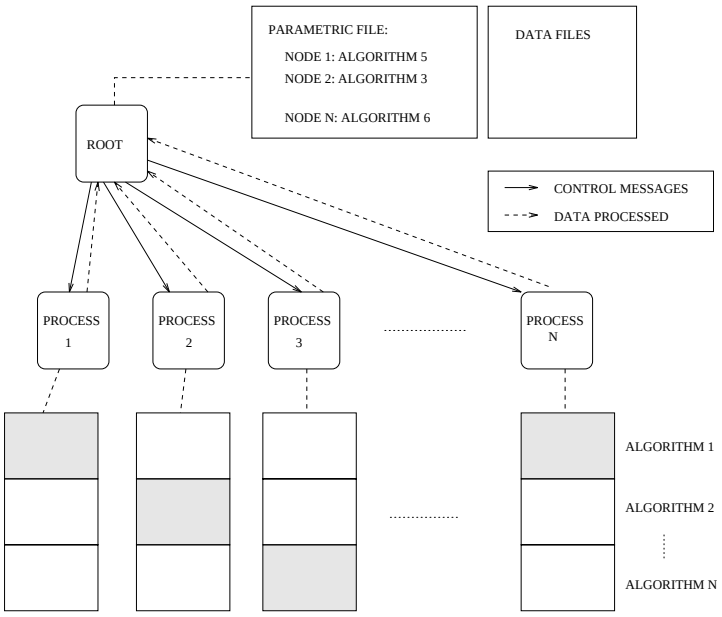


**Fig. 1.** DIPORSI schema

DIPORSI runs in a batch way. Thus, all the user has to do is to generate a parametric file with the following information: The algorithm (or the sequence of algorithms) to manipulate the remote sensed image (i.e. the georeferring algorithm using the bilinear interpolation method), the workload distribution to the computing nodes, and the number and the location of the data images.

DIPORSI provides the user a number of functions for managing files of different kinds such as text files, data files, raster images, etc. Our application allows us to work by distributing both spatially and temporarily either of the remote sensing algorithms. That is, we can make all the nodes perform the same computations on different data or each node does different computations on the same image. Also, DIPORSI generates special messages to the user in response to execution of the distributed algorithm.

Initially, DIPORSI was implemented with a set of functions to manage files to allow that processes manage the data transfer control. These functions were designed by means of MPI functions to send and receive data and control files like images or the parametric files. The idea was based on providing flexibility to individual processes. However the response times were affected by the task of broadcasting the images among the nodes. Currently, DIPORSI uses the file system functions provided by NT operating system instead of calling a function implemented from MPI primitives. Thus, the distributed processes can make use of the network capabilities present in the NT operating system like protection and security.

## 3 The performance of DIPORSI

In this section we present the comparative results between DIPORSI versus the sequential execution of a particular algorithm. For comparison purposes, we run a remote sensing algorithm that is used to compute a corrected image from a distorted one. Next, the algorithm used will be briefly described. In [6] we presented a preliminary results we have obtained with the initial implementation of DIPORSI.

When a image is remotely sensed, a number of different errors appear distorting it in such a way that cannot be used correctly. The process of georeferring a satellite image consists in the application of a set of mathematical operations on the original image to obtain a geometrically corrected image. So, the purpose of geometric correction is to compensate for the distortions introduced by different factors (earth curvature, relief displacement, and so on) so that the corrected image will have the geometric integrity of a map [11, 12].

The usual procedure applies the traditional polynomial correction algorithm with or without using a digital model terrain. This kind of distorsion is corrected by analyzing well-distributed ground control points occurring in an image. These values are then submitted to a least-squares regression analysis to determine coefficients for two coordinate transformation equations that can be used to interrelate the geometrically correct coordinates and the distorted image coordinates. Once the coefficients for these equations are determined, the

distorted image coordinates for any map position can be precisely estimated. However, the process is actually made inversely. An undistorted output matrix of empty map cells is defined, and then each cell is filled in with the gray level of the corresponding pixel (digital number or DN), or pixels depending on the method employed, in the distorted image [10]. That is to say, the process is performed using the following operations:

1. The coordinates of each element in the undistorted image are transformed to determine their corresponding location in the original distorted image.
2. The intensivity value or DN of the undistorted image is determined by using one of these usual methods: nearest neighbour, bilinear interpolation and cubic convolution.

The distributed algorithm works in a similar way as the sequential. The first step is made by the master process, which opens a file that contains the information of the task to be solved. This process reads a parametric file, in which all the activities to be performed in the distributed environment and their parameters are specified in an ordered way. The master process sends such information to all the nodes involved. Thus, each node knows what it must do (correction method) and how much information it must receive (number of bands -the resultant image to a given frequency-, resolution, the transformation functions, etc).

The next step consist of each node runs the algorithm, which acts on the region of the distorted image where the computations must do the corrections. Each node uses the file system functions to access to the data and compute its partial sub image locally.

The last step consist on the task of recovering the resultant image. In the older algorithm this was made by the root process. In this distributed algorithm, each node writes its partial sub image directly on the main node by means of file system functions.

A Landsat image has 7 bands, 6 with 30x30 meters and 1 with 120x120 meters of resolution respectively, with approximately 40Mb each band, which explains the high value of the response time when geometric correction is computed in a single machine. Our parallel algorithm works by splitting the original distorted image into number-of-rows/number-of-nodes blocks in accordance with an uniform workload allocation, then each node computes a small submatrix reducing the computation time and improving the overall performance of the algorithm.

Our hardware environment is the following. The single machine is a Pentium II 333 Mhz with 32 MB of RAM running Windows NT Workstation 4.0. The distributed machine is composed of the 8 PII 333 Mhz with 32 MB of RAM running Windows NT Workstation v4.0. The nodes are linked using a 10 Mbps Ethernet.

The figure 2 shows the comparative results obtained between the sequential and the distributed algorithm. The times of the distributed algorithm were obtained with two nodes. It can be observed that the larger image sizes are used, the better results are achieved with the distributed algorithm.

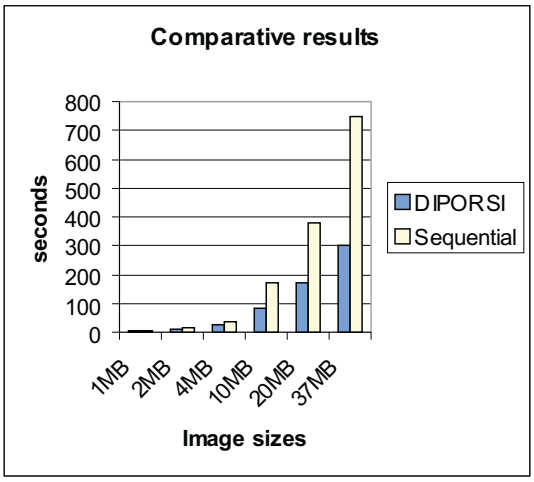


**Fig. 2.** The distributed algorithm vs sequential.

In the figure 3 we can see the speedup obtained with the distributed algorithm for a range from 2 to 8 nodes. Obviously, not all the nodes show the exactly same behaviour though are quite similar.

The figure shows the linear reduction of the time as many as the number of the nodes increase. Moreover, it can be seen that we achieve a near linear speedup for large image sizes.

However, these figures show the computation times, not the total response time of the algorithm. This value is worse due to the communications overhead by transferring the images through a slow interconnection network. Amdahl's law implies that the speedup obtained from faster processors is limited by slowest system component; so, it is necessary to improve the network performance such that it balances with CPU performance. Therefore, we can evaluate DIPORSI with a fast interconnection network like fast Ethernet or Myrinet. Another possibility is to distribute image files in several hard disks. It would be supported by a parallel file system based on software RAID. In this way, we could improve the I/O performance by means of carrying out I/O operations also in parallel.

## 4 Conclusions and future work

This paper describes our distributed workbench called DIPORSI that has been designed to execute the main remote sensing algorithms using a cluster of workstations.

Our distributed algorithm has been coded with MPI. MPI is very useful for implementing distributed applications on low-cost platforms, specially suitable in the remote sensing area.

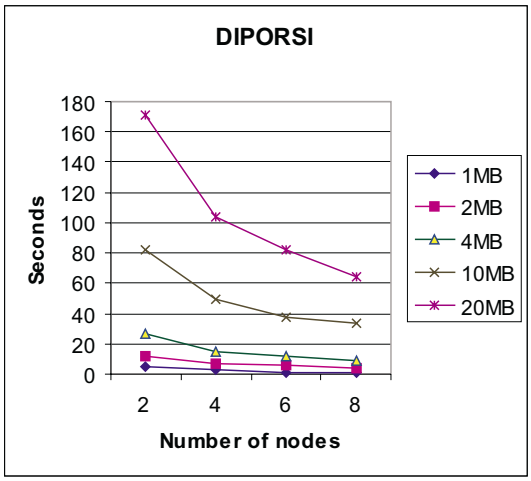


**Fig. 3.** Results obtained with the distributed algorithm (only computations).

The implementation shows the timing results when different image sizes are used. These results show a nearly linear speedup for large image sizes, and a reduction in the execution time over the 400% for a moderate number of nodes.

Future work admits of several possibilities: the evaluation of DIPORSI with fast networks and software RAIDs, the implementation of other algorithms to solve geometric correction, and finally, the integration with Unix plattforms.

## References


1. Andersen, J.D., *Digital Image Processing: A 1996 Review*, Applied Parallel Computing - 3rd International Workshop Para 96. LNCS 1184, Springer-Verlag (1996).
2. Baker, M., *MPI on NT: The Current Status and Performance of Available Environments*, EuroPVM-MPI'98, LNCS No. 1497, Springer-Verlag, (1998), pp 63-75.
3. Culler, D., Liu, L.T., Martin, R.P., Yoshikawa, C.O., *Assessing Fast Network Interfaces*, IEEE Micro, 16(1), (1996), pp 35-43.
4. Dasgupta, P., *Parallel Processing with Windows NT Networks*, Proc. of the USENIX Windows NT Workshop, August (1997).
5. Foster, I., Geisler, J., Gropp, W., Karonis, N., Lusk, E., Thiruvathukal, G., Tuecke, S., *Wide-area Implementation of the Message Passing Interface*, Parallel Computing, No. 24, (1998), pp 1735-1749.
6. Gallud, J.A., García-Consuegra, J.D., Sebastián, G., *Distributed Georeferring of Remotely Sensed LandSat-TM Imagery Using MPI* , Para'98, LNCS No. 1541, Springer-Verlag (1998), pp 161-167.
7. Gropp, W., Lusk, E., Skellum, A., *Using MPI Portable Parallel Programming with the Message Passing Interface*,The MIT Press, (1994).
8. Hoffman, F.M., Hargrove, W.W., *Multivariate Geographic Clustering Using a Beowulf-style Computer*, PDPTA'99, Las Vegas (USA), (1999).

9. Lee, C., Hamdi, M., *Parallel Image Processing Applications on a Network of Workstations*, Parallel Computing, No. 21, (1998), pp 137-160.
10. Lillesand, T.M., Kiefer, R.W., *Remote Sensing and Image Interpretation 2nd Edition*, J.Wiley & Sons.
11. Markham, B.L., *The Landsat Sensors' Spatial Responses*, IEEE Transactions on Geoscience and Remote Sensing, Vol. GE-23 No. 6, (1986).
12. Mather, P.M. *Computer Processing of Remotely-Sensed Images*, John Wiley & Sons.
13. McCormick, J.A., Alter-Gartenberg, R., Huck, F.O., *Image Gathering and Restoration: Information and Visual Quality*, Journal of Optical Society of America, Vol 6 No. 7, (1989), pp 987-1005.
14. Piernas, J., Flores, A., García, J.M., *Analyzing the Performance of MPI in a Cluster of Workstation Based on Fast Ethernet*, EuroPVM-MPI'98, LNCS No. 1497, Springer-Verlag (1998), pp 63-75.