PARAMETER TRANSMISSION PROTOCOL FOR GRID IMAGE PROCESSING

Pescaru Dan[1], Toma Corneliu[2], Chirila Ciprian[3], Gui Vasile[4], Tundrea Emanuel [5]

(1, 3, 5) "Politehnica" University of Timisoara, Department of Computer Science, Bd. V. Parvan no 2, 300223 Timisoara, Romania,
e-mails: dan@cs.utt.ro, chirila@cs.utt.ro, emanuel@emanuel.ro;
(2, 4) "Politehnica" University of Timisoara, Department of Communications, Bd. V. Parvan no 2, Timisoara, 300223 Romania,
e-mails: ctoma@etc.utt.ro, vasile.gui@etc.utt.ro.

Abstract: This paper describes a protocol for parameters transmission used in a framework for parallel implementation of a large class of image processing algorithms. The purpose of the framework is to develop these algorithms with a minimum parallel programming knowledge. This work can be applied in automation of the technological processes witch imply large bitmap images processing. The aim of this research is to provide a simply, portable and efficient programming interface for these algorithms. It is based on Java technology, portable and widely accepted in industry. Efficiency is gained using a message passing distributed model on JPVM platform. Achievement of these goals is proved through examples and experimental results.

Keywords: image processing, grid computing, distributed computing, MIMD, JPVM.

1. INTRODUCTION

The requirements for industrial image processing algorithms [6] can be achieved only by using a distributed environment. These kind of processing environment not only helps in optimum utilization of such resources but also helps in achieving better throughput using multiple processors in parallel.

Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements [1].

Following these facts, usage of a grid as base environment for image processing is a promising option. In a previous work we developed an architecture for this kind of environment,

named PFIP [7]. It is based on Java technology, portable and widely accepted in nowadays industry. Efficiency is gained using a message passing distributed model on JPVM [3, 4] platform. This technology is bases on old PVM [5] architecture that has a large support of researcher community. In this paper we enhance the capabilities of PFIP by adding a protocol for parameter transmission to the image processing algorithms.

2. EXPERIMENTAL RESULTS

The objective of **PFIP** (Parallel Framework for Image Processing) is to provide a framework for implementing parallel processing of image processing algorithms. It consists in a master (derived from **PFIPMaster**) class that create N clients (derived from **PFIPSlave**). Each client could process either a specific function on different image partitions or different functions on the same partition. The general architecture is presented in Fig. 1.

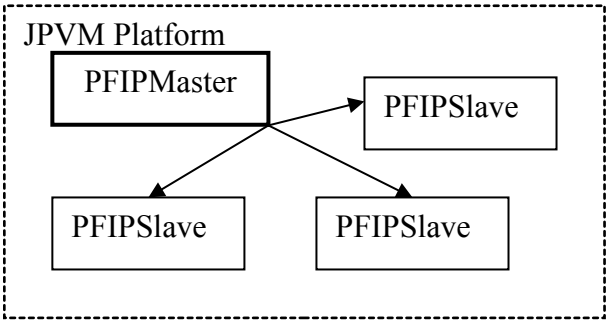

Fig. 1. PFIP general architecture

The PFIPMaster provides a minimal interface for general services, like: setting the number of slaves, splitting and packing an bitmap image, sending data to slaves, receiving results and assembling results.

The PFIPWorker class provides common functionality like: connection to master, receiving image together with all necessary parameters, calling algorithm implementation, sending back the result to master and cleanup worker resources.

The proposed protocol for parameter exchange under the presented architecture is a request/response protocol. A master process sends a request to the slaves in the form of a request method followed by packed algorithms data over a connection with them. The slaves responds with a status information composed of a success or error code, followed by a message containing the computational load information. This information is used by the master in order to balance the tasks distribution over the grid.

The slave processes could also initiate a dialog when local processing is done. The request will contain the results, packed as described below and some statistical information about resource

usage. The response of the server is just an acknowledge that will clear all buffers allocated at the slave level.

Data and control information exchanged through this protocol are packed and prepared using a special package of classes named PFIPUtility. The content of this package is:

- PFIPImage - manage the internal representation of a bitmap image
- PFIPImageCollection - manage a collection of PFIP images, usually all slices that belongs to a decomposed image
- PFIPParameter - manage the internal representation of a basic type parameter like integers, floats or logical
- PFIPParameterCollection - manage a collection of PFIP parameters, usually gather all parameters necessary in algorithm implemented by a PFIP worker
- PFIPControlInformation - pack statistical information about computational resources used for tasks balancing

The defined message types are: *request-processing, acknowledge-processing, request-information, send-status, push-results, acknowledge-results*.

To prove PFIP concept and transmission protocol overhead we choose an image segmentation algorithm based on multi-channel texture-filtering [2]. The resulting speed-up depends of the image size and to the ratio between computers and network speed. All processes was run over a 100 MB LAN consists in 4 HP PII 400 MHz workstations running Microsoft Windows NT 4.0 Workstation, Java v. 1.4.1 and JPVM v. 0.2.1.

Image size	**Numbers of involved PC**	**Time**	**Speed-up**	**Protocol overhead**
16 KB	1	64''		
16 KB	4	60''	1.06	73% (44'')
64 KB	1	183''		
64 KB	4	62''	2.95	50% (16'')
216 KB	1	546''		
216 KB	4	157''	3.47	13% (20'')

Fig. 2. Experimental results

The results are presented in Fig. 2. They demonstrate the speed-up gathered using PFIP considering various image sizes and the transmission protocol overhead. All experiments use the same algorithm, first implemented as standalone non-parallel program on one computer and second as PFIP implementation using four computers.

3. DISCUSSION

This paper describes a protocol based on a framework that allows an image-processing researcher to develop parallel applications very easy, in an almost transparent manner. All software involved in this framework, respectively *jdk* and *jpvm*, is freeware.

4. CONCLUSIONS

The protocol presented here as part of the PFIP framework satisfies the need of transferring parameters through a distributed image processing application. As presented in section 2, despite of that relative simplicity, important performance issues are obtained. As a future work we propose an extension consisting in a standard mechanism to enabling user control of tasks-per-client granularity based on information collected from the grid points.

5. REFERENCES

[1] Berman F., Fox G., Hey A.J.G., 2003, "Grid Computing: Making The Global Infrastructure a Reality", John Wiley & Sons.
[2] Comer L. M., Delp E. J., 1995, "Multiresolution Image Segmentation", Proceedings of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing - May 1995, Detroit, USA, Vol. 4, pp. 2415-2418.
[3] Ferrari A.J., 1997, "JPVM", Technical report CS-97-29, Department of Computer Science, University of Virginia, Charlottesville, VA 22903, USA.
[4] Ferrari, A.J., 1998, "JPVM: Network Parallel Computing in Java", JavaWorld'98
[5] Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., Sunderam S., 1994, "PVM: Parallel Virtual Machine", MIT Press.
[6] Parker J.R., 1997, "Algorithms for Image Processing and Computer Vision", Wiley Computer Publishing.
[7] Pescaru D., Mocofan M., 2004, "Efficient Implementation for Image Processing Algorithms", 6th International Conference on Technical Informatics – CONTI'04, Timisoara, Romania, Vol. 49(63)/2004 No.4, pp. 217-222.