CZASOPISMO TECHNICZNE M
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ

z. 3-M/2008
ISSN 0011-4561
ISSN 1897-6328

RAFAŁ PETRYNIAK*

ANALYSIS OF EFFICIENCY OF PARALLEL COMPUTING IN IMAGE PROCESSING TASK

ANALIZA EFEKTYWNOŚCI RÓWNOLEGŁYCH ALGORYTMÓW PRZETWARZANIA OBRAZÓW

A b s t r a c t

The article deals with parallel computing applied in image processing. An algorithm of edge finding was examined and analysed in tests. Each parallel approach is described in detail and the strengths and weaknesses of each are shown. Different solutions have been implemented to answer the question: "When and how to improve the efficiency of image processing?".
One of the conclusions is that there is a need to build parallel image analysing algorithms to enable running them on new computers with a parallel architecture.

Keywords: *parallel image processing*, *efficiency of parallel computing*, *edge find problem*, *MPI communication standard*

S t r e s z c z e n i e

Artykuł opisuje przebieg badania dotyczącego zrównoleglania procesów przetwarzania obrazów medycznych. Udzielono odpowiedzi na pytania, czy zrównoleglenie jest przydatne i czy uzyskana efektywność satysfakcjonuje nas w każdym przypadku. Przedstawiono, w jakich sytuacjach takie podejście nie jest wskazane i pogorszy wydajność algorytmu. W badaniu zastosowano algorytmy służące do wykrywania krawędzi w obrazie.

Słowa kluczowe: *równoległe przetwarzanie obrazów*, *efektywność obliczeń równoległych*, *wykrywanie krawędzi obrazu, standard komunikacji MPI*

*Rafał Petryniak, MSc, Institute of Applied Informatics, Cracow University of Technology.

1. Introduction

The Gordon Moore's [1] principle referring to doubling the computer efficiency every 18 months can be in a short time not true any more. The author claims [2] that silicon technology in the next 10–20 years will achieve the top efficiency, and from the physical point of view further miniaturization of such devices will not be possible.

A practical solution to this problem can be parallel computing. This has been noticed and applied by both: hardware producers who produce multi-core processors (e.g. Intel Core Duo) and software makers who recommend multi-thread programming, develop distributed software architectures and create tools helping with parallel programs construction.

Following the above an idea has appeared to check the possibility of efficiency improvement in image processing with parallel algorithms.

2. Research objectives

The main goal of the research was to measure the efficiency of parallel medical image processing and evaluation whether this approach is reasonable. It was necessary to check when the parallel approach is valuable and when it is not.

Another objective was to propose a few ideas of paralleling, which under some conditions is recommended and most effective.

3. Analysed domain

The area of interest is medical images. The reason for this is that in image diagnostics – a very important part of medical diagnostics – the information on patients' health is stored in a graphical way. To have such data reliable, information has to be extracted precisely and saved on a data medium. A short time ago, maybe a few or several years ago, image devices recorded low resolution data with small depth of tint. But nowadays, medical devices often store MB of data just from one examination (e.g. series of images from a tomograph).

4. Description of process used in image analysis

In my experiments an edge find problem was analysed for some medical images. This process consisted of a couple of tasks which had to be run in a proper order and with appropriate parameters (see Fig. 1).

The first step was a negative operation (*inversion*) on an original image, followed by *thresholding* in order to get binary data. Then *Sobel filtering* was run to find edges on the image. After another *inversion* the final output image was received [3].

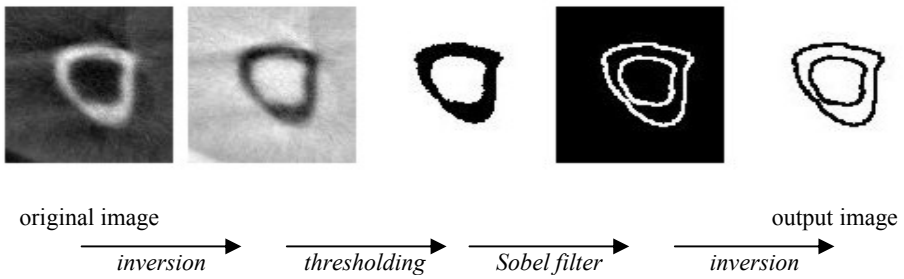

Fig. 1. The process of edge finding on the image
Rys. 1. Kolejność operacji algorytmu wykrywania krawędzi

These tasks do not consume much computer processor unit time. Each operation needs only one scanning of every image (2 loops in a computer program) and modification of the current pixel.

One of the assumptions of this work was to construct algorithms so that running them on more than one computer gives the same output images as for one computer. Several solutions were built that treat the problem in different ways. The results of these solutions and estimation of each approach are presented in what follows.

5. Testing environment

The laboratory environment can be divided into three parts: test images, software, hardware.

A series of 92 images of human bottom limb from a tomograph scanning was analysed. The images were in grey scale, in size 256 x 256 pixels.

The programs for image analysis were implemented in C++ programming language. During software creation and testing a few tools were used: a programming environment Eclipse with a PTP (Parallel Tool Platform) component, Easy BMP graphic library and SSH protocol. All concepts presented forthwith were designed in a message sending model with an OpenMPI library support.

The programs were run on up to 10 computers in a laboratory at the Institute of Applied Informatics at Cracow University of Technology. All computers (called on nodes, processing nodes or computing nodes) had the same hardware configuration: 512 MB RAM, processor Intel Celeron 1.8 GHz, operating system Linux Ubuntu 6.12 with core 2.6. Nodes were connected together by Ethernet 100 Mb.

6. Description of tests

Several algorithm solutions were prepared for the experiments. The first idea regarded sequential image processing on one computer. All other ideas needed more than one processing node and they realised parallel algorithms. Below descriptions of each solution and its results are presented in detail.

7. Sequential solution

In this approach the algorithm runs on one node and processes one data stream. Beforehand, the program reads the data from a disk, then the main procedure starts and calls subroutines which realise inversion, thresholding and Sobel filtering. After processing the image is saved on a disk. The algorithm for this process is written below.

```
m – no_images
FOR i ← 1 TO m DO
input_file = ReadFile (i)
output_file = ProcessImage (input_file)
SaveFile (output_file)
END
```

In this test environment the algorithm finished in 11.68s (average score for 3 tests), which means average 0.1270s for 1 image.

8. Parallel ideas

8.1. The idea of local processing

In the first parallel solution it was assumed that every processing node has all input data (92 images) locally on its disk. One of the nodes, called a coordinator, sends information on proper data to the other computing nodes to process. That is the end of the communication between nodes. After receiving messages the computing nodes read the proper data, process them and save locally on a disc. Fig. 2 shows execution times on a growing number of processing nodes.

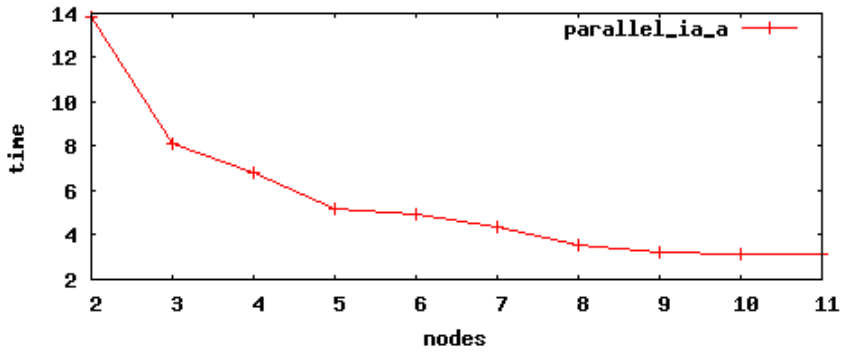

Fig. 2. Algorithm execution times for local processing idea
Rys. 2. Czas wykonania dla koncepcji lokalnego przetwarzania

As we can see, a proper increase of effectiveness was achieved with 5 nodes (1 coordinator node and 4 processing nodes).

Below (see Fig. 3) the communication diagram for the analysed algorithm is presented.

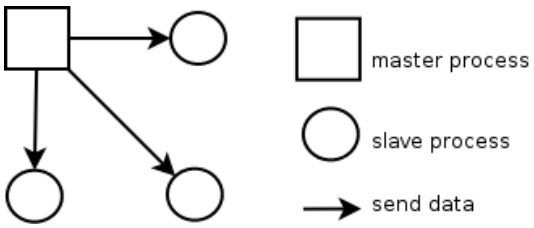

Fig. 3. Communication diagram for local processing idea
Rys. 3. Diagram komunikacji dla koncepcji lokalnego przetwarzania

The basic problem in this approach is a need to gather output data from the processing nodes. An additional effort to support this can make the solution less effective. Installation of a Network File System (NFS) can improve this. Thanks to this solution all nodes may have access to the same data that is kept in one place.

After analysing the above parallel concept and its problems, new ideas appeared to construct algorithms that don't require distributed data. Here are three suggestions:

1) sequential data sending,
2) concurrent data processing,
3) image partitioning.

8.2. The idea of sequential data sending

A computer with all input data (92 images) is a coordinator and it is responsible for communication and processes synchronisation. The coordinator sends an image to the first computing node and waits for the response of processing finish. After the response is received the coordinator contacts with another computing node and sends another image. Although it is a parallel approach, it is very ineffective, because at every moment computations are run only on one node. Figure 4 presents the algorithm execution times on the growing number of processing nodes.

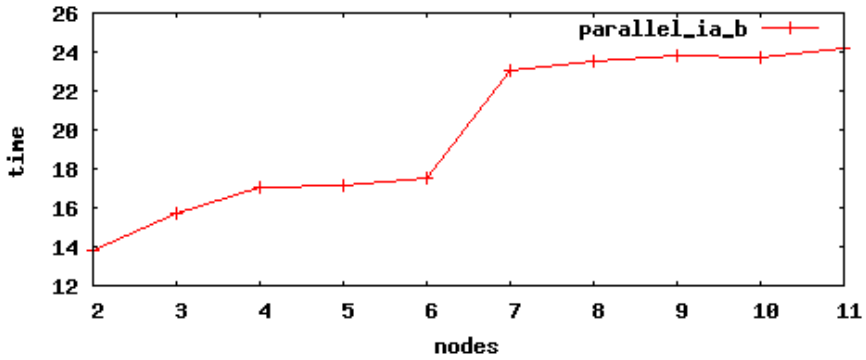

Fig. 4. Algorithm execution times for sequential data sending
Rys. 4. Czas wykonania dla koncepcji sekwencyjnego przesyłania danych

From the diagram we can see that execution times grow with the increasing number of computers. The reason may be a need for the coordinator to keep communication with a growing number of nodes.

The idea of the algorithm is shown in Fig. 5.

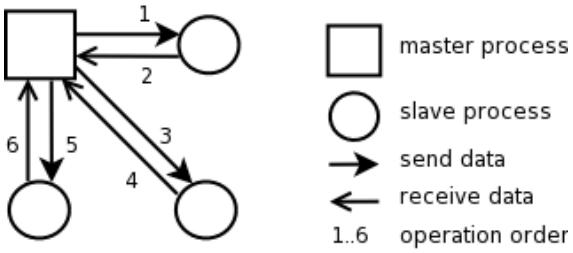

Fig. 5. Communication diagram for sequential data sending
Rys. 5. Diagram komunikacji dla koncepcji sekwencyjnego przesyłania danych

8.3. The idea of concurrent processing

Because of the huge imperfection of the previous approach related to synchronous and blocking process of communication, a new solution was created. To use all nodes in an effective way the coordinator's process (master process) was divided into two subprocesses (two threads: data sending thread and data receiving thread). The threads are a mechanism for concurrent tasks running. This idea is illustrated in Fig. 6.

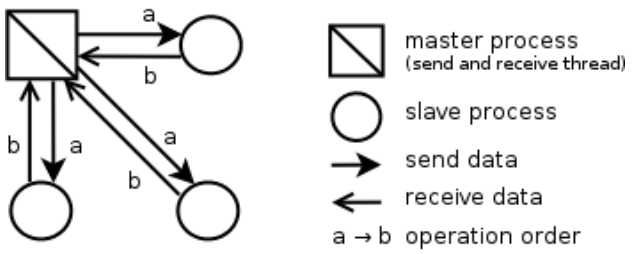

Fig. 6. Communication diagram for concurrent processing
Rys. 6. Diagram komunikacji dla koncepcji przetwarzania współbieżnego

The sending thread sends one image after another to the computing nodes and does not wait for responses (as in the previous approach). The receiving thread waits for messages of the finished image processing from the computing nodes. In this approach every node is processing all the time.

This algorithm can be suspended in two cases: 1) when all nodes are processing data the sending thread has to wait for sending another image; 2) when the receiving thread is waiting for processing finish by any of the computing nodes.

From the implementation point of view this solution requires additional restrictions, which enable work synchronisation for both threads – sending and receiving. One critical clause was to send an image to a node before trying to receive the image from this node. If this condition was not met, there would be a deadlock in the program.

The diagram (see Fig. 7) shows how the efficiency changes for different numbers of computing nodes.

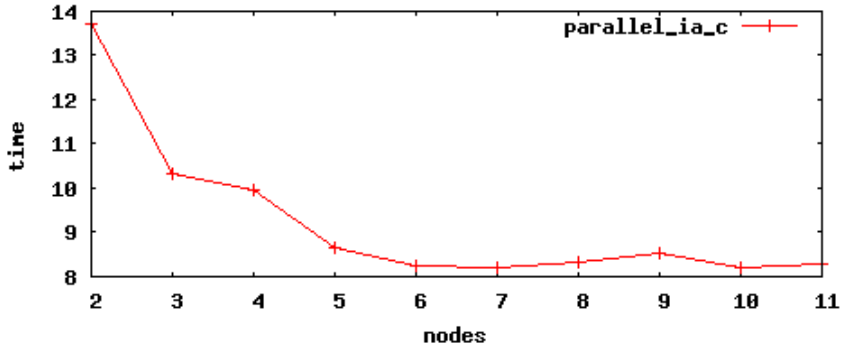

Fig. 7. Algorithm execution times for concurrent processing
Rys. 7. Czas wykonania dla koncepcji przetwarzania współbieżnego

It is easy to notice that the effectiveness increases up to some moment, and then is kept at a similar level. The possible reason may be an increasing cost of administration required by threads synchronisation (in other words, it is an additional time spent on tasks not connected directly with image analysing).

This approach is recommended when hardware configuration of nodes differs, because the nodes that finished processing can get another data to process. Thanks to this the algorithm does not depend on the slowest computer. Unfortunately, in other presented algorithms such dependence does exist.

8.4. The idea of data partitioning

The last idea assumed equal partitioning of every image and processing these parts by all nodes. After reading an image from the disk the coordinator node divides it into portions and sends it to the processing nodes. When processing of the portion is finished, the output of the process is sent to the coordinator. The coordinate node composes the image back from pieces and saves it on the disk.

This solution requires sending additional information on one extra line of image to the neighbouring nodes. It is required by Sobel filter which needs surroundings for the analysed point.

The results of the algorithm are shown in Fig. 8. In the diagram (Fig. 8) we can notice that the acceleration we got for 5 computers is lost for e.g. 12 computers. This can be caused by big granularity of data and, connected with it, time consumed for communication. This approach is recommended for large data. The concept diagram is presented in Fig. 9.

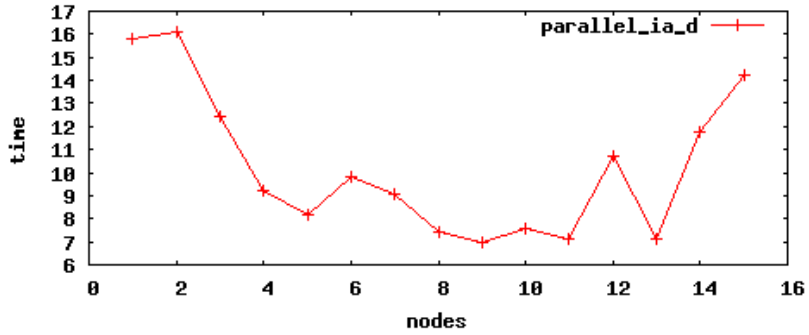

Fig. 8. Algorithm execution times for data partitioning
Rys. 8. Czas wykonania dla koncepcji dzielenia danych

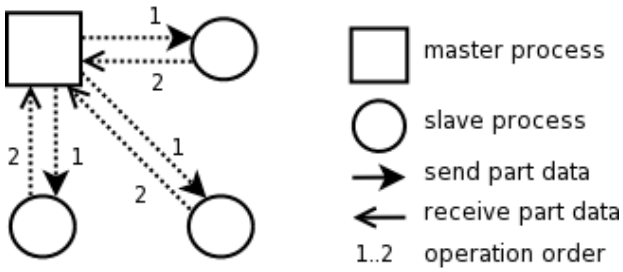

Fig. 9. Communication diagram for data partitioning
Rys. 9. Diagram komunikacji dla koncepcji dzielenia danych

It is important to remember that in this solution the whole processing is limited by the slowest node.

9. Conclusions

From the analysis of the results for each algorithm we may conclude that parallel solutions can improve the efficiency of image processing. This statement is true under some conditions. The first qualification is a large amount of input data (series of images or high resolution images), so the image processing time is longer than communication and synchronisation time. The second qualification is a compound algorithm that after receiving data for computing needs a processor for a long time. Simple tasks, like reading and saving data, should be done on one computer called coordinator, whereas the computations should be decomposed on other nodes.

In these tests the qualification of large data (the series of 92 images) was achieved. But not complicated computations (4-times modification of each pixel in every image) implied effectiveness not proportional to the number of nodes. In this case the cost (hardware) was bigger than the profits (faster processing). However, in critical situations any increase of efficiency is needed.

The algorithm of edge finding applied in this research enabled us to implement the parallel solution easily. But not every algorithm is dedicated to parallel computing. For example, when an algorithm requires global information on an image, not only its part, or when it assumes dependencies between images in a series [4], then we cannot decompose this algorithm.

10. Discussion

Remembering different limitations and requirements for every presented parallel approach, it is worth trying to increase the efficiency of image processing.

We should design and construct algorithms with vision of dividing computations into many nodes. It is essential nowadays, because multi-core processors and grid computing (grids, clusters) are becoming more and more popular.

References

[1] Moore G.E., *Cramming more components onto integrated circuits*, Electronics Magazine, 1965.

[2] *Moore's Law is dead, says Gordon Moore*, Techworld.com, 13 April 2005, http://www.techworld.com/opsys/news/index.cfm?NewsID=3477.

[3] Wojnar L., Kurzydłowski J.K., Szala J., *Praktyka analizy obrazu*, Polskie Towarzystwo Stereologiczne, Kraków 2002.

[4] Latała Z., *Zastosowanie komputerowej analizy obrazu ultrasonograficznego do badania serca*, rozprawa doktorska, Wydział Mechaniczny Politechniki Krakowskiej, Kraków 2002.