ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, January 2015

Parallel computing in digital image processing

Sharanjit Singh[1], Parneet kaur[2], Kamaldeep kaur[3]

Assistant Professor Mtech (CSE), Guru Nanak Dev University Amritsar, India [1]

Student Mtech (CSE), Guru Nanak Dev University Amritsar, India[2,3]

Abstract: Application with sequential algorithm can no longer rely on technology scaling to improve performance. Image processing applications exhibits high degree of parallelism and are excellent source for multi-core platform. Major challenge of parallel processing is not only aim to high performance but is to give solution in less time and better utilization of resources. Medical imaging require more computing power than a traditional sequential computer can do and we also know that for medical imaging, it is necessary that the image is clear and be obtained as quickly as possible. We can achieve through the process of parallelizing. Parallelizing optimizes the speed at which the image is produced.This paper presents the different types of parallelism in image processing i.e., data, task and pipeline parallelism. This paper also discusses three types of operators; point operators, neighborhood operators and global operators used for image processing. Different algorithms used for parallel image processing are discussed and the application of medical imaging is discussed using work flow engine Taverna for scientific processing.

Keywords: RAC, Taverna, threshold, parallelization

INTRODUCTION

Image is the 2-dimensional distributions of small image points called as pixels. It can be considered as a function of two real variables, for example, f(x,y) where f as the amplitude (e.g. brightness) of the image at position (x,y). Image Processing is the process of enhancing the image and extraction of meaningful information from an image. Image processing is gaining larger importance in a variety of application areas. Image processing is widely used in many application areas including the film industry, medical imaging, industrial manufacturing, weather forecasting etc. In some of these areas the size of the images is very large yet the processing time has to be very small and sometimes real-time processing is required.

Parallel processing is a vital part of any high performance computing model. It includes the utilization of large amounts of computing resources to complete a complex task or problem. The resources specific to parallel processing are CPU and memory.

Image Processing with parallel computing is an alternative way to solve image processing problems that require large times of processing or handling large amounts of information in "acceptable time". The main idea of parallel image processing is to divide the problem into simple tasks and solve them concurrently, in such a way the total time can be divided between the total tasks (in the best case). Parallel Image processing cannot be applied to all problems, in other words we can say that not all the problems can be coded in a parallel form.

LITERATURE REVIEW

Sanjay Saxena, Neeraj Sharma, and Shiru Sharma [1], in 2013, presented their work on parallel implementation of different sequential algorithms. The main focus was to improve the performance of segmentation, de-noising and histogram processing. They used multi-core architecture for designing some parallel processing algorithms like noise reduction, features calculation etc. Rafal Petryniak [2], in 2008, examined an algorithm for edge finding which was then analyzed on different tests. He gave the strength and weakness of each parallel approach. Medical images were the main focus in the research. The conclusion drawn was that every algorithm is not dedicated to parallel computing and also parallel solutions can improve the efficiency of image processing. Preeti Kaur [3], in 2013, calculated the various parameters such as fork time, serial time, join time, parallel time, and overheads. Her work deals with reducing the amount of time required to represent the digital images. This work helps to improve the performance of image processing algorithm and allows maximum utilization of the multi-cores. .

PARALLEL COMPUTATION

Features of a parallel program

A parallel program must have some features for a correct and efficient operation otherwise; it is possible that runtime or operation does not have the expected performance. These features include the following –

a) *Granularity:* It is defined as the number of basic units and it is classified as :-

- **Coarse-grained**: Few tasks of more intense computing.
- **Fine grain**: A enormous number of small parts and less intense computing.

b) *Synchronization:* This prevents the overlap of two or more processes.

c) *Latency:* This is the time transition of information from request to receipt.

d) *Scalability:* It is defined as the ability of an algorithm to maintain its efficiency by increasing the number of processors and the size of the problem in the same proportion.

e) *Speedup and efficiency:* These are metrics to assess the quality of parallel implementation.

f) *Overheads:* Extra time needed for the computation.

Requirements for better performance with parallel execution

DOI 10.17148/IJARCCE.2015.4139

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, January 2015

Computer system/servers with integrated multiple processors and better message facilitation among processors
Os effective at controlling the multiple processors.
Clustered nodes with software pc software, such as for example Oracle RAC, which may parallel across the nodes.
The research tasks are involved by several processors the tasks are finished in a lesser amount of time.

- Parallel processing effects in quicker delivery of tasks, increasing throughput. A big amount of tasks could be performed in confirmed system of time.

TYPES OF PARALLEL PROCESSING

Pipeline parallelism

In this sort of processing, long sequences of procedures, or tasks, are parallel, but additionally, there are overlapping successive processes all through which number parallel tasks are possible. The relational model matches in to that model really well. The result of some relational operators becomes the input for other operators; thus, some waiting time is involved. There is a considerable amount of time saved in the completion of an activity through the appropriate use of direction parallelism.

Independent or Natural Parallelism:

In this sort of parallelism, the tasks don't rely on other tasks. As a result, complete delivery time is considerably reduced.

Inter- query and intra-query parallelism:

Transactions are independent. No transaction involves the result of still another transaction to complete. Many CPUs could be kept active by assigning each job or issue to a separate CPU. This kind of parallelism, employing many split up, separate queries at once, is called inter-query parallelism. This can be a organic option for OLTP-type operations. Even some small DSS procedures work that way. Ergo, the higher how many CPUs on a repository host, the greater the performance is likely to be, generally in most situations. To speed up the delivery of an individual large and complex issue, that model decomposes it in to smaller problems. It then executes these smaller tasks concurrently by assigning them split up CPUs. This kind of parallelism is an all-natural option for DSS-type procedures where a single transaction analyzes, computes, and revisions thousands of repository blocks.

Task parallelism

In job parallel strategy, image processing recommendations low level procedures are collected in to tasks and each job is given to another research unit. Picture processing software consists of several various operations. The key concern in job parallel strategy is successful knowledge decomposition and effect composition.

ALGORITHMS FOR PARALLEL IMAGE PROCESSING TASKS

The key stage of these formulas is to determine how many tiles to be generated. How many tiles fit to the total amount of threads? If just a thread exists, the computation is simply successive computation. Otherwise if you can find two or more strings then the image is divided into distinct areas, as revealed in Determine 5, 6. Each thread is in charge of processing the pixels included in its tile and to accomplish various tasks but considering maintaining synchronization between all the processors usually there will be the situation of deadlock between processors. Assume we're getting these images for instance:

Fig. 1. An image

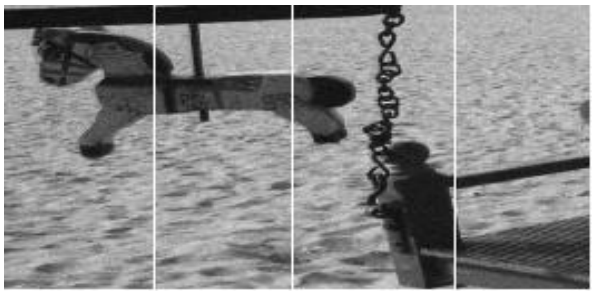

Fig. 2. Division Of Image into Vertical Threads

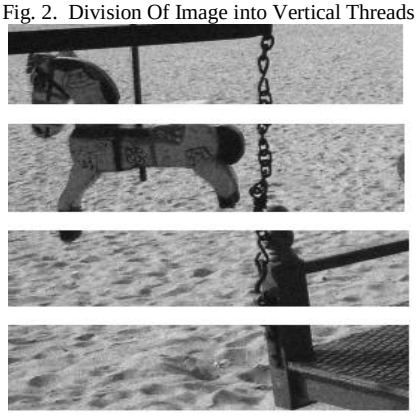

Fig. 3. Division Of Image into Horizontal Threads

According to the requirement we can divide an image and send it different processor.

Pipeline Segmentation by Region Growing Technique and Calculation of different Features of Segmented Regions

Region growing is one of the very popular techniques of segmentation. Main disadvantage of the region growing is that it is time consuming. This algorithm is created to

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, January 2015

reduce timing of this algorithm. These are the steps included in it.

1. Get Input image on client processor.
2. Selection of the Seed pixels (It is based on some user criterion like pixels in a certain gray level range, pixels evenly spaced on a grid, etc.) for different regions on client processor.
3. Count number of seed points and activate same number of kernels.
4. Send copy of image and respective seed pixels on each kernel processor.
5. Regions are then grown from these seed pixels to adjacent depending on pixel intensity, gray level texture, or color, etc on each kernel processor for this image information is also important because region grown is based on the membership criteria.
6. Calculation of different features of different ROI (region of interest) on individual kernel.
7. Send segmented regions and information of ROIs to the client processor and deactivation of worker kernels.
8. Reconstruct the image on client processor and Display of all the information.

Parallel Segmentation by Global Thresholding of the image

This algorithm involved the following steps:-

1. Divide the image into Quad Tree Structure in customer processor.
2. Send a flag from customer processor to different worker processor and activation of four worker processors.
3. Send all the parts of the picture on different kernels or different labs or different worker processors.
4. Chose preliminary Thresholds T0, T1, T2, T3 in different labs individually.
5. Estimate the mean value of each lab μ of the pixel below the threshold and the pixel above the threshold value.
6. Compute a new threshold as:-

$T = (\mu 1 + \mu 2)/2$ in each lab.

7. Repeat steps 5 and 6 until there is no modification in threshold in each lab or worker.
8. Send segmented region to customer from the different labs.
9. Deactivate worker processors.
10. Reconstruct the Segmented Image.

Histogram equalization of an Image by Parallel Computing

1. Divide the image in quad tree format or in row wise or in column wise format and activate required number of kernel processors.
2. Form the cumulative histogram on each part of image on each kernel processor.
3. Normalize the value by dividing it by total number of pixels.
4. Multiply these values by the maximum gray level value and round off the value.
5. Map the original value to the result of step 3 by a one to one correspondence.
6. Send all equalized histogram to the client processor.

DESIGN APPROACH FOR TAVERNA-BASED PARALLELIZATION OF MEDICAL IMAGING

The essential methods of Taverna–based application development are decomposition of computer software techniques into components. Two types of decomposition need to be differentiated: element decomposition (also called task decomposition, regarding the code parts) and knowledge decomposition. Portion decomposition decomposes the application into elementary units, which may be viewed as a dark package between identified feedback knowledge and production data. How you can decompose the application usually follows these steps:

- Decomposition of the application into theoretically different components for easier reusability of these building blocks;
- Decomposition of large but independent components into smaller components, wherever possible, creating check pointing data that can be used if the task needs to be restarted;
- Decomposition of each component by differentiating the input and output parts to facilitate the data structure modification;
- Finally, a decomposition of each acquired component into objects with a log strategy to enable debugging and error tracking procedures.

Data decomposition is easy to understand and is related to the decomposed components. A component executed with a part of the data forms a task.

AN EXAMPLE OF PARALLELIZATION USING TAVERNA

To test our method with a concrete request, a content–centered picture access (CBIR) platform, named Parallelized Medical Picture Retrieval (ParaMed IR) is produced centered with this setup. The data and computationally many rigorous part of CBIR are to catalog sources of images applying visual features. From every picture, crucial visual information called functions is extracted. These details could be simple colour or structure features. In this process, functions are split into teams called visual crucial words.

To begin with, picture parts are identified (through a fixed grid, randomly or through fascination level detectors), and in neighbourhoods of the parts visual functions are extracted. The extracted visual functions are then clustered in to a confined amount of similar visual crucial words. Mathematical info on the appearance of these visual key words in images (normally in the forms of histograms) is analyzed for every single picture to measure the likeness sets of images. It may conceptually be decomposed in to three main components: extraction of functions from the images, clustering of the functions in to visual key words, and generation of the histograms which are then stored in a database and other indexing structure.

DOI 10.17148/IJARCCE.2015.4139

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, January 2015

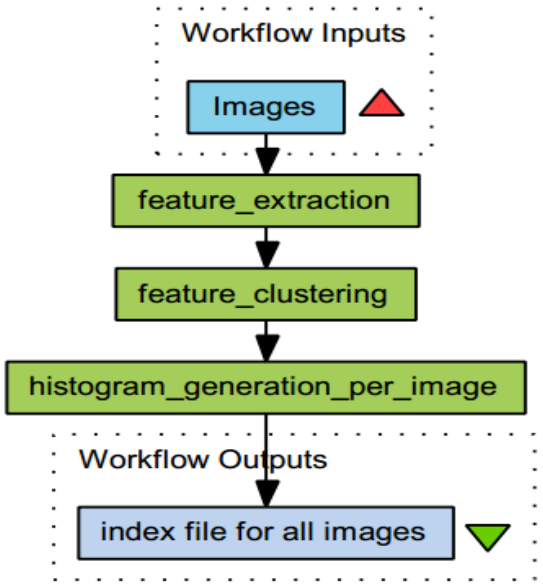

Fig. 4. Conceptual Decomposition of a Medical Imaging Workflow for Feature Extraction into Basic Components.

The performance of the application form is beginning via Taverna and can use a number of accessible resources know to Taverna. Following workflow design, the experts need to produce sequential executables for every single workflow factor and link them to the finished workflow map. Then, the parallel performance needs to be configured for every single workflow factor.

CONCLUSION

This paper centers on parallel computing in digital image processing tasks. Different options that come with parallel program and the requirements for better performance are given. The forms of parallel processing in image processing receive i.e., data, task and pipeline parallelism. We've also presented the algorithms for parallel image processing tasks.

Parallel Segmentation by Region Growing Technique and Calculation of different Top features of Segmented Regions, Parallel Segmentation by Global Thresholding of the image and Histogram Equalization of an Image by Parallel Computing would be the three algorithms given in detail.An application of parallel computing in medical imaging is discussed in detail. For medical imaging development, a workflow engine called Taverna is discussed and its design approach is also given.

REFERENCES

1) Sanjay Saxena, Neeraj Sharma, Shiru Sharma (2013), "Image processing tasks using parallel computing in multi core architecture and its applications in medical imaging", International Journal of Advanced Research in Computer and Communication Engineering, Volume 2, Issue 4, ISSN: 2278-1021.
2) Rafal Petryniak (2008), "Analysis of Efficiency of Parallel Computing in Image Processing Task", Cracow University of Technology.
3) Preeti Kaur (2013), "Implementation of Image Processing Algorithms on the Parallel Platform Using MATLAB", International Journal of Computer Science and Engineering Technology, Volume 2, No. 6, ISSN: 2229-3345.
4) Thomas Braunl (2001), "Tutorial in Data Parallel Image Processing", Australian Journal of Intelligent Information Processing System (AJIIPS), Volume 6, No. 3, pp. 164-174.
5) C. Nicolescu and P. Jonker (2001), "A data and task parallel image processing environment," Lecture Notes in Computer Science, Vol. 2131, pp. 393–408.
6) Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R.,Wipat, A., Li, P.: Taverna (2004), "A tool for the Composition and Enactment of Bioinformatics Workflows", Bioinformatics 3045.3054.
7) Eric Olmedo, Jorge de la Calleja, Antonio Benitez, and Ma. Auxilio Medina (2012), "Point to Point Processing of Digital Images using Parallel Computing", International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, 1694-0814.
8) Müller, H., Michoux, N., Bandon, D., Geissbuhler, A. (2004), "A review of content based image retrieval systems in medicine- clinical benefits and future directions", International Journal of Medical Informatics 73.1.23.
9) Cristina Nicolescu and Pieter Jonker, "A data and task parallel image processing environment", Faculty of Applied Physics, Pattern Recognition Group, Delft University of Technology, Netherlands.
10) W. Burger and M. J. Burge (2009), "Principles of digital image processing: core