Data Parallel Programming Model for an Operation Based Parallel Image Filtering System

Myat Su Hlaing[1*], Soe Mya Mya Aye[2]

Abstract

In computing, a parallel programming model is an abstraction of parallel computer architecture, which it is convenient to express algorithm and their composition in programs. Data parallelism is parallelization across multiple processors in parallel computing environments. Parallel image filter is implemented on distributing and collecting the updated image where communication among processes is necessary for exchanging the updated values of pixels. This research focuses on distributing the image data across different nodes, which operate on the image data in parallel. The parallel approaches for image filtering is taken by processing with the distributed memory implementation using Message Passing Interface (MPI) and hybrid implementation using shared memory implementation using Open Multi-Processing (OpenMP) and MPI.

Keywords: Data Parallelism, MPI, OpenMP, parallel programming model

Introduction

Image Processing is the process of enhancing the image and extraction of meaningful information from an image. Smoothing filters are used for blurring and noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image prior to object extraction. Noise reduction can be accomplished by blurring. A very large size digital image on a single machine can take a lot of time and the operation is computationally intensive. A possible solution is to reduce the processing time, to parallelize the processing on Central Processing Unit (CPUs) and to take advantage of distributed computing infrastructures such as computational grids [5]. The main objectives of this paper is to research a multithreading method for computing the image registration computing, to parallelize the image registration using multiprocessing method, to reduce the processing time and to implement image filtering using multithreading and multiprocessing methods. The MPI and OpenMP programming models can be combined into a hybrid paradigm to exploit parallelism beyond a single level.

Cluster Computing

Cluster computing is best characterized as the integration of a number of off-the-shelf commodity computers and resources integrated through hardware, networks, and software to behave as a single computer. Clusters may also be deployed to address load balancing, parallel processing, systems management, and scalability.

Windows Compute Cluster Server (CCS) has been installed on a 32-node cluster headed by Dell® PowerEdge 2900 server computer at University of Yangon. Each computing node is configured with eight 2.33-gigahertz (GHz) Intel® quad-core processors. 3Com® Switch 5500G offers 1G connectivity to the cluster's network [3], [4]. The cluster is accessed by a user's workstation in the university's local area network.

1 Lecturer, Department of Computer Studies, University of Yangon.

2 Professor (Head), Department of Computer Studies, University of Yangon.

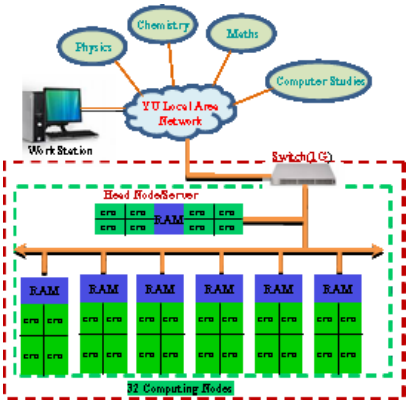

Figure 1: Cluster at University of Yangon

CCS-based HPC cluster configuration at University of Yangon is shown in Figure1. This configuration uses both head node of Network Interface Cards (NIC), with Gigabit NIC_1 connecting to the compute nodes and Ethernet NIC_2 connecting to the university's local area network; the compute nodes use also Gigabit NIC_1.

Image Filtering

An image f(x, y) *is* sampled so that the resulting digital image has M rows and N columns. The values of the coordinates (x,y) now become discrete quantities. The notation introduced in the preceding paragraph allows to write the complete MxN digital image in the following compact matrix form in Eq. (1).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad (1)$$

The spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image [1]. Spatial domain methods are procedures that operate directly on the pixels. Spatial domain processes will be denoted by Eq. (2).

$$g(x,y) = T[f(x,y)] \quad (2)$$

Where f(x, y) is the input image, g(x, y) is the processed image, and *T* is an operator on f, defined over some neighbourhood of (x, y). The operator T is applied at each location (x, y) to yield the output, g, at that location. A filtered image is generated as the center of the mask visits every pixel in the input image. Figure 2 shows area or mask processing methods.

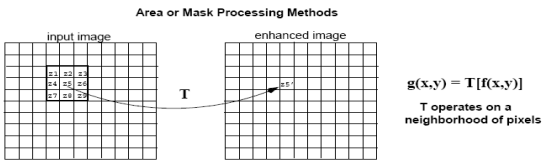

Figure 2: Area or Mask Processing Methods

In Spatial Domain filtering technique, median filtering is researched in this paper. The median filter is used to reduce noise in an image like the mean filter. The median filter operation is given by Eq. (3). In median filtering at a point in an image, the median is calculated by first sorting all the pixel values from the surrounding neighbourhood into

numerical order and then replacing the pixel being considered with the middle pixel value as shown in Figure 3.

$$\hat{f}(x,y) = \underset{(s,t)\in S_{x,y}}{\text{median}} \{g(s,t)\} \qquad (3)$$

124	126	127
123	150	125
115	119	120

Neighborhood values:
115,119,120,123,124,125,126,127,150
Median value: 124

Figure 3: Taking a Median Value in a 3x3 neighbourhood

Parallel Programming Model

A parallel programming model is a set of software technologies to express parallel algorithms and match applications with the underlying parallel systems. The most common forms of interaction are shared memory and message passing interface.

Shared Memory Implementation using OpenMP

Open Multi-Processing (OpenMP) is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C/C++ and FORTRAN, including UNIX and Microsoft Windows platforms [2]. For a share memory implementation the multi-core technology is used in a multi-core computer. Multithreading method becomes the most efficient programming tool for improving the computing capacity. A task queue is created when the master thread derives a thread. Each thread gets a task from the queue and does the computation for filtering after one block image has registered. The idle thread continues to get an unfinished task from the task queue and do it, until there are no more tasks needed to compute. The processes involved in OpenMP can be seen in Figure 4.

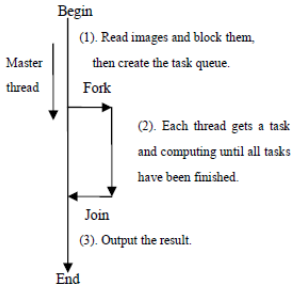

Figure 4: Blocking and merging the image

Distributed Memory Implementation using MPI

For registering the image on the cluster, the image is blocked into “n” smaller matrices and then each pixel value is replaced by the value of filtering them on different nodes “N”. Each node produces the computation and then transfers the result back to the head node. The process of blocked image registration on cluster. Parallel processing with MPI is done step by step as follows:

Step 1: On the head node, the image is blocked into “n” smaller blocks.

Step 2: The head node retrieves block data of source image and then sends other blocks data of each image to a compute node.

Step 3: Each node receives block of sub image and the sub image block is replaced by new pixel value according to filter method.

Step 4: The blocked result data from the compute nodes are gathered into the head node.

Step 5: The blocked data are merged and output as the resultant filtered image.

Hybrid Implementation with both OpenMP and MPI

Hybridization using MPI and OpenMP facilitates cooperative shared memory (OpenMP) programming across clustered compute nodes because MPI facilitates communication among compute nodes and OpenMP manages the workload on each compute node. MPI and OpenMP are used in tandem to manage the overall concurrency of the application. Parallel processing with Hybrid MPI and OpenMP is done step by step as follows:

Step 1: On the head node, the image is blocked into "n" smaller blocks.

Step 2: The head node retrieves block data of source image and then sends other blocks data of each image to a compute node.

Step 3: In each node, the master thread of process creates a task queue which contains "n' elements depending on a multi-core computer and derives a new thread, then each thread running on different cores gets a task from the task queue to compute with traditional sequential algorithm.

Step 4: The threads are joined into the master thread of each process after all blocks are replaced by new pixel value according to filter method in each node.

Step 5: The blocked result data from the compute nodes are gathered into the head node.

Step 6: The blocked data are merged and output as the resultant filtered image.

MPI and Hybrid based parallel version up to 24 nodes are used in cluster consisting of the head node of HSERVER1 and the compute nodes of HSERVER2, HSERVER3 and HSERVER4. The relative speedups computed as SP(N)=T(1)/T(N), SP(N) is speedup using N compute nodes, T(1) is execution time on only one node and T(N) is execution time on N nodes.

Results and Discussion

The sequential implementation programs and parallel implementation programs are compiled and executed on the Dell® PowerEdge 2900 server computer with Microsoft Visual Studio 2008 .Net. Parallel based median filter on different processors using MPI and Hybrid version with 2048 x 2048 of image have been implemented on up to 24 processors. The execution time using using MPI and Hybrid are shown Table 1.

Table 1 Execution Time (Seconds) using MPI and Hybrid

No of Processors	MPI	Hybrid
1	5.505	5.505
8	1.109	0.989
12	0.738	0.738
16	0.491	0.491
20	0.356	0.349
24	0.308	0.301

The relative speedups are shown in Figure 5. The highest speedup of Parallelized filter is achieved by Hybrid for 2048 x 2048 of pixels of image. The best performance on cluster server is achieved using the Hybrid of OpenMP and MPI as shown in Table 1.

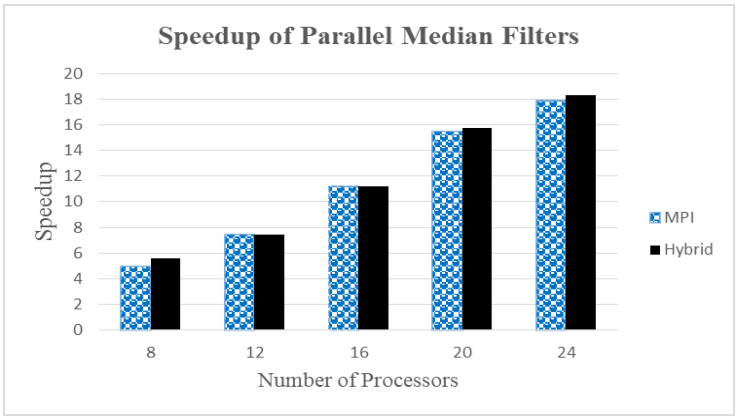

Figure 5: Speedup of Parallelized Median Filers

Conclusions

The parallel approaches are taken by processing with distributed memory implementation using MPI and then hybrid implementation with both OpenMP and MPI. Programming for parallel processing highly depends on the computer's architecture, especially memory architecture and parallel interface. By running the proposed filters on up to 24 PCs (Processors), the speedup is about eighteen times faster than on 1PC (Processor) on median filtering using MPI. The speedups of the two parallel median filters go from about five on 8 processors to about eighteen on 24 processors of the compute cluster server. Linear speedups are obtained using all parallelization approaches on the cluster. The highest speedup is achieved by hybrid median filter using OpenMP and MPI on 24 processors. Parallel programming model of Hybrid method is the best method for cluster computing.

Reference

Gonzalez,R.C. and Woods,R.E. , (2002) **Digital Image Processing, 2nd Edition**, Prentice Hall, USA, New Jersey, ISBN: 0-130-94650-8.

Grama,A., Gupta,A., Karypis,G. and Kumar,V. , (2003) **Introduction to Parallel Computing**, 2nd Edition, Pearson Education, The Benjamin/Cummings, ISBN: 7-111-12512-6.

Pho Kaung and Ye Chan, (2009) **Performance Test of Fast Fourier Transform on the Computing Cluster at Yangon University**, Regional Conference on the 1st AUN/Seed-Net Electrical and Electronics Engineering, International Symposium on Multimedia and Communication Technology, Bangkok.

Pho Kaung, (2009) **High Performance Computing on the Compute Cluster at Yangon University**, Asia Research Network, Vol.6, No.1.

Vecchiola,C., Nadiminti,K. and Buyya,R. , (2007) **Image Filtering on .NET-based Desktop Grids**, 6th International Conference on Grid and Cooperative Computing.