Project Name : **Edge Detection by applying Stencil code on Image**
Course Name: Scalable
Computing

Dozent: Prof. Dr. Peter Luksch
Tutor: Meisam
Booshehri
Ripan Kumar Kundu 217205751

Abstract

In this project our task was to implement Serial Version program that takes an input Image and then Produces an output image after convolving Stencil matrix with the input image, for reading and writing an image we t write our code using library:

- **Libpng**

First of all, I wrote a serial version code in C++ to detect an edge.For obtained the the output image of edge dectected I apply the stencil matrix to the input image. Then, we parallelized our sequential code using OpenMP and MPI.After the execution I calculate the execution time both for serial version,OpenMP and MPI.I found that openmp is faster than serial version.and then further parallelize the code by using both OpenMP and MPI and found the significant execution time.

Introduction:

Edge Detection:

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods.

2.2 Implementation approach for Edge detection

step 1: Extract all pixel values of Input image and store in 2D array
step 2: Convolve each pixel of an Input image with stencil kernel
. During convolution avoid the convolution for boundary pixel of an
image.
step 3: check if the new pixel value is greater than 255, set new pixel value
equal to 255.
step 4: if the new pixel value is less than 0, set new pixel value equal to 0.
step 5: store all new pixel value in 2D array
step 6: write a new image with new pixels.

Stencil Operation:

We will use at nine point stencil that performs edge detection. *Qx,y*

$$Q_{x,y} = \begin{array}{llllll} c_{00}P_{x-1,y-1} & + & c_{01}P_{x,y-1} & + & c_{02}P_{x+1,y-1} & + \\ c_{10}P_{x-1,y} & + & c_{11}P_{x,y} & + & c_{12}P_{x+1,y} & + \\ c_{20}P_{x-1,y+1} & + & c_{21}P_{x,y+1} & + & c_{22}P_{x+1,y+1} & \end{array}$$

Consider P(x,y) be a pixel value of an input image and Q(x,y) is a pixel of the output Image which lies at the intersection of the row x and the column y. I(i,j) are the elements of the 3x3 stencil matrix.

Portable Network Graphic(libpng):

It is a platform independent library that contains C/C++ functions for handling PNG Images. So we are using this library for reading and writing our Input image.

Serial Version of Program:

In our Serial Version program, we first read our image and then used 2D Array pointer for reading the rows and columns of an image and then convolved it with Stencil Matrix and then we wrote our output image.

Parallel Version using OpenMP:

OpenMP is an application programming interface(API) that supports multi-platform shared memory multiprocessingprogramming in C, C++, and Fortran. A portable standard for shared memorymultiprocessing. It is used to reduce the execution time of the program byusing Fork-Join Model in which master thread creates a team of parallelthreads. SPMD(Single Program Multiple Data) technique was used for edgedetection. We parallelized our program using OpenMP and we have seen that it supports multi threaded programming and that's why we have seen a moderate difference in timing afterusing OpenMP, the timing comparison is shown below in the histogram.

Parallel Version using MPI:

The Message Passing Interface (MPI) standard is an approach which allows the exchange of data between different processes in a cooperative manner.It defines a collection of operation and their semantics but not their concrete protocol or implementation. It is a programming paradigm in which mainly send and receive operations are performed within nodes and processes. Communication occurs when a portion of one process's address space is copied into another one. This operation is cooperative and occurs only when the first process executes a send operation and the second process executes a receive operation When we parallelized our program using

Message Parsing Interface (MPI) we have seen high performance of our program, We have seen how master node assigns jobs to slave nodes and then when the slave nodes finish their job they all give back the results to master node and then master node writes the output Image we have checked it with nodes 4,,8,16.

Histogram:

Computation time taken for our task is different according to the approach we used. Time taken for the execution of the program by distributing the task within several processes and executed in parallelized manner is faster than the sequential program. Furthermore, optimized code by using both OpenMP and MPI approach gives the significant execution time in comparison to previous one.

The execution time of the sequential program, OpenMP and both OpenMP & MPI program using a different number of threads are shown in below diagram.

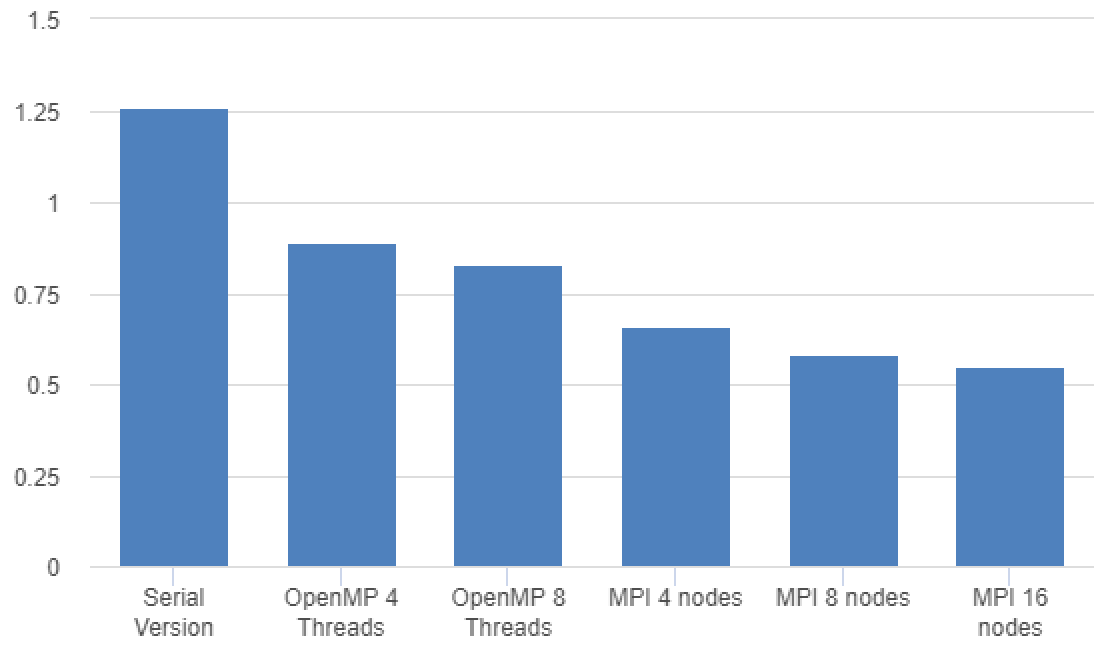

Figure: Different program execution time difference

Conclusion:

As according to the Histogram generated the performance of MPI is much higher than OpenMP and Sequential program.

References:

1. MPI. Retrieved (2018 Jan 21) from https://computing.llnl.gov/tutorials/mpi/
2. OpenMP. Retrieved (2018 Jan 21) from https://computing.llnl.gov/tutorials/openMP/
3. OpenMP. Retrieved (2018 Jan 21) form http://www.openmp.org/

4. https://www.mathworks.com/discovery/edge-detection.html

5. https://www.coursera.org/lecture/parallelism-ia/stencil-introduction-JIKh6