|epcc|

Message Passing Programming Coursework Assignment

Exam number B136013

November 15, 2018

Contents

1 Introduction

The project solves an image processing problem. It uses a two-dimensional domain decomposition in order to split the workload to the active processes. To achieve this we use MPI communication protocol for process communication. This approach arises a variety of challenges that need to be addressed, such as communication, decomposition, and boundary conditions.

2 Project Specification

2.1 Description

There are a variety of project requirements in order to produce a correct output. On the one hand, there are fixed "sawtooth" boundary conditions in the horizontal direction. On the other hand, there are periodic boundary conditions in the vertical direction. This means that when a top process performs halo swap to fill the upper edge of the local table it receives it from the according to bottom process.

Another specification is the terminate condition. The main loop of image reconstruction should finish when the maximum difference of a pixel in the image between the old and the value it's insignificant. This means that after some iterations when the produced image has not drastic variations from the previous the loop should be terminated.

3 Analysis

3.1 Tools

For the development of this project the used programming language is C due to its performance and for low-level calculations. In addition, GNU Make was used for the build phase and python to compare the output for testing reasons.

3.2 MPP API

The implementation of the project uses mainly the basic functions of the MPI API. Some of these features create a virtual topology, perform non-blocking communication and use derived types.

3.2.1 Communication

The main functions used for the non-blocking communication between the processes are MPI_Isend and MPI_Irecv. There are called in scatter and gather part that communication needs to be established between the master process and all of the slaves. Also, in each iteration of the calculation loop, we use them in order to send and receive the halo swaps.

3.2.2 Topology

In terms of the produced virtual topology, the main function was MPI_Cart_create that creates the new 2 dimension topology, in addition, methods like MPI_Cart_coords and MPI Cart rank are used identify the neighbor's ranks or coordinates.

3.2.3 Derived Types

In order to reduce the code volume and avoid unnecessary memory allocations derived types are used extensively. Derived types such as row, column, and table are declared once in the main function. They can be found in the communication phase like the non-blocking functions. Their main goal is to avoid memory copies for the send and receive buffer. What we managed to is to read and write directly from the old buffer.

3.3 Design

The design and control flow of the project is basically the same as the case study with a different implementation. In the beginning, the program creates the virtual topology, does the decomposition of the problem and fill the necessary data structures for the rest of the execution. Then the master process reads the image and stores it to its master buffer. At the same time, all of the processes allocate the necessary buffers for the calculations.

At this point the data exchange takes place. The master scatters the image which is stored in the master buffer directly to the edge buffers of the workers. When this part is done each process the old buffer filling it with white and fixes the horizontal borders if the worker has a part of the image which belongs to the left or right side.

After the initialization is completed the main loop is ready to start. The calculation phase is decomposed as followed:

1. Halo swaps are sent to the neighbors if they exist, otherwise to MPI_PROC_NULL
2. The middle calculations are computed (excluding those that require the halo swaps)

3. The program waits to receive the halo swaps and then performs their part of calculations
4. At specific intervals, the average pixel is logged and the program checks if the loop can be terminated
5. If not the new buffer is overwritten to the old one
6. Step 1 is executed

In the end, the master gathers all of the old buffers and reconstructs the master buffer which will be written to the new output image.

3.4 Input/Output

On the one hand, the input of the program is an edge image. On the other hand, the output is a new reconstructed image and a log that contains the average pixel between the fixed interval of the main calculation loop.

4 Evaluation

Evaluation has been done in order to find out if the program performs as it should. All of the experiment has been run on the backend of Cirrus supercomputer and the code is compiled using -O3 flag for serial optimization in the machine code.

4.1 Correctness

First and foremost, before performance analysis, we have to ensure that the produced outputs are valid regardless of the number of processes that are used underneath. The approach is very simple. We run the serial program from all of the given input images and stored the outputs. Then we run the experiment using a different number of processes.

4.1.1 Testing

Each time our program produces a new image we compare it with the stored output of the serial code. The comparison is made automatically through a Python script that uses the filecmp function. If the files are the same then our experiment is correct. It worth mentioning that the termination condition has to remain the same in all of the executions because different conditions create different results.

4.2 Performance

Graphs: demonstrate that the performance of the code improves as you increase the number of processes. processes Input variety: performance change input(problem) size and see how it scales Speedup of strong scaling: performance metrics speed up vs ideal (linear) speedup average time per iteration how performance is affected by the problem size (small problem becomes worse with many processes)

4.2.1 Timing

The timing of the experiment excludes Input and Output. The timing starts before the scatter and ends after the gather of the buffers. This decision has been made in order to take into consideration only the communication and calculation procedures.

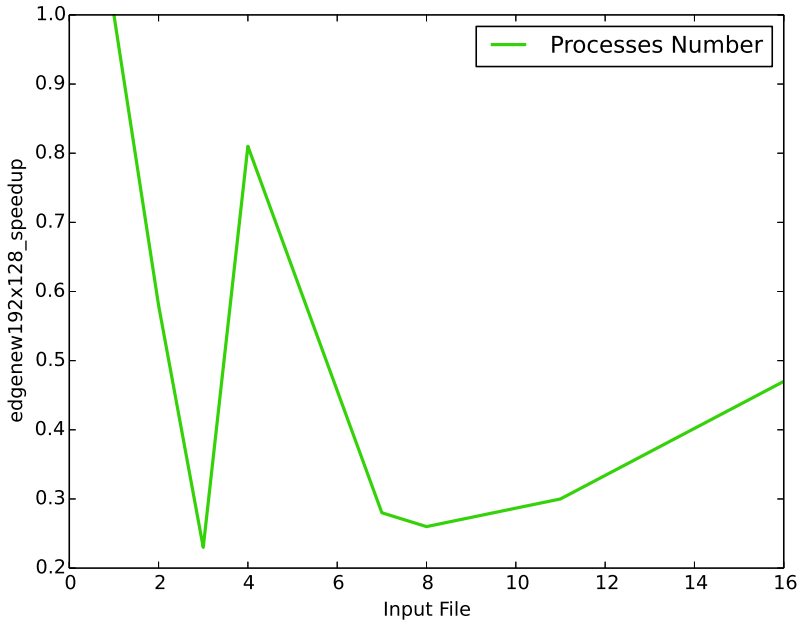

Figure 1: Running Time Loop 2 with 4 Threads

4.2.2 Average Pixel

Performance analysis for the average pixel results.

5 Conclusion

In conclusion.