ASSIGNMENT 2A - PROGRAMMING WITH MPI REPORT

CMT106 – High Performance Computing

1.Introduction

MPI is a cross-language communication protocol for writing parallel computing, which support peer-to-peer and broadcast. MPI is an information delivery application interface that includes protocol and semantic descriptions that indicate how it can perform its functions in various implementations. The purpose of MPI is high performance, large scale, and portability. MPI is still the main model for high performance computing today. It's different from OpenMP parallel programs. MPI is a parallel programming technique based on information transfer. The messaging interface is a programming interface standard, not a specific programming language. In short, the MPI standard defines a set of programming interfaces that are portable.

In this assignment, I focus on the basic use of MPI, and discuss the performance about using 2D application topology of MPI with 1,2,4,6,8,10,12 processes. In detail, understand how a parallel program running by MPI, generate the test cases, and learning how to code the picture blurring algorithm of into an MPI 2D application topology program, also build performance model for parallel code that expresses the speed-up on architecture to see how the performance improved.

2.Experiments environment

The experiments have been conducted on the school provided Linux laptop (latitude 5490). This machine has one CPU(Intel(R) Core CPU i5-8250U @ 1.6Ghz) with 4 cores. And each core has 2 threads therefore totally 8 threads. Each core is equipped with 32KB L1 data cache and 256KB L2 cache. And the operation system is Ubuntu 18.04.3 LTS Linux. The compiler is Open MPI 2.1.1

3. Timing experiment

3.1 description

In this experiment I had done some implementation on the code then executed the parallel program, record output data, draw graph and finally developed a simple performance model for the parallel code and got some feature from it. Firstly I insert two parts code to a MPI 2D application topology parallel program， the first part is declaring and initializing some variables to make sure the topology had been set up and the second part is coding shift communication operation in the left right, up and down directions on the different color arrays which mostly fit the parallel requirement. When I run the program, I set the number of processes to 1,2,4,6,8,10,12 to observe the program running under different processes. Then, I recorded their running time and combined the results of the running with only one process to calculate their speedup under different conditions and made a graph. Finally, I developed a performance model for the parallel code and got some feature and conclusions from performance model and the figure.

3.2 performance model

Generally speaking ,for calculate the ideal parallel speedup, we suppose the serial program process time is t(serial), and the same parallel program process time is t(parallel), then the speedup is t(serial)/ t(parallel).But if we want to get the relationship between speedup and number of processers, we need to think about how the MPI 2D application topology works, The update formula requires 4 floating-point operations per grid point. The number of grid points per processor shifted in the left/right direction is n/P, where n x n is the size of the grid and P is the number of processors in one column of the processor mesh. The number of grid points per processor shifted in the up/down direction is n/Q, where Q is the number of processors in one row of the processor mesh.[1]

Therefore, the speedup for this MPI 2D application topology parallel program is:

$$
\begin{aligned}
S(N) &= \frac{4n^2 t_{calc}}{(4n^2/N)t_{calc} + (2n/Q)t_{shift} + (2n/P)t_{shift}} \\
&= \frac{N}{1 + (P+Q)\,\tau/(2n)} \\
&= \frac{N}{1 + (Q/n)(1+\alpha)\tau/2}
\end{aligned}
$$

[2]

We can clearly see that speedup is directly proportional to the number of processors：

$S(N) = kN$

Here k is a constant.

3.3 speedup graph (the detailed speedup data can get from timingData.xlsx)

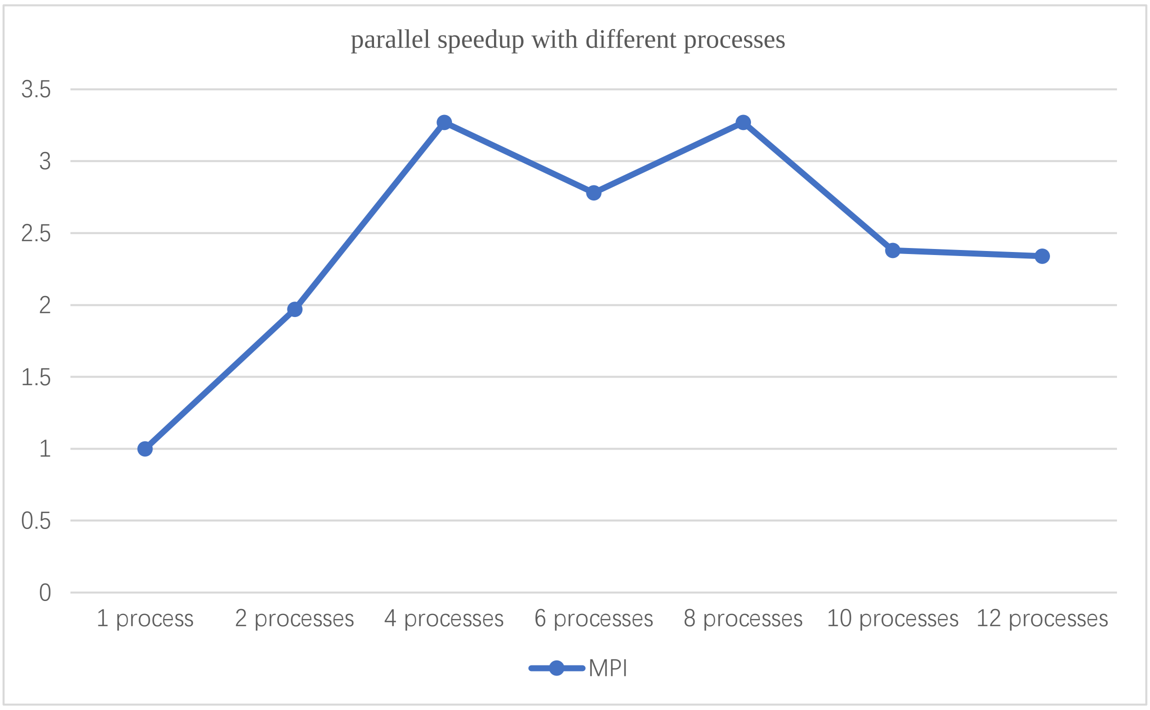

Figure 1 parallel speedup with different processes

Comparing the above figure with the performance model, we can clearly see that the results can meet the speedup requirements especially when using 1,2,4 processes but not totally fit the performance especially when using more than 4 processes. The different happens maybe the reason that in this experiment, we just using the machine with only 4 cores which couldn't give more speedup when we using more than 4 processes. Therefore, it's important for us to choose multi-core computers to doing HPC research or executing a multi-thread program.

4.Conclusion

This is a very good assignment which helps us totally understand the basic knowledge about MPI. Thorough following the step by step guidance from code, I can finally implement a MPI parallel program and can have ability to do some basic analysis of a parallel program .But at the same time I also consider that there still be a lot of points need me to master, especially the ability to build an complex performance model. To this end, my next step is to read more article about parallel computing to learn more about performance modeling of parallel programs in future.

Reference

[1] [2] David W. Walker, Day 4: High Performance Computing CMT106 ,19 November 2019 pp. 61-62.