CSC4005

Distributed and Parallel Computing

Assignment #4

Heat Simulation

Hajun Lee

117010437

Introduction

In assignment4, we should implement 'Heat simulation' by using sequential, pthread, MPI and openmp. Those are to simulate heat simulation to compare the efficiency of all those methods what have mentioned.

Method

Basically, we need to simulate and calculate each block about heat.

1. the place where fire comes out is fixed with 100 * 0.5 * width in every size of output.
2. I managed to process when output size square.
3. I should consider the happens when height of the output can divide by the number of processes which are threads.

In the assignment4 pdf file, the temperature of the wall is 20 degree, and the temperature of the fireplace should be 100 degree. So, after simulating process, it will go to stable state.

Design

Different with other assignments, this time I need to customize header file of paint.h.

1. paint.h

to make result clear, xlib are for draw the heat distribution.

So before, I defined variables needed for draw and set the function of setColor().

16 colors for heat range of 20 to 100 and each color shows up to 5 range.

For temperature_to_color_pixel function, input degree of position and return of corresponding color pixel.

2. sequential.cpp

Before main function:

1. Include Header files
2. Modified variables.

Main function:

1. Initialized drawing configurations and the heat distribution.
2. Draw.
3. startTime
4. draw
5. endTime
6. fin.

3. MPI.cpp

Before main function:

1. Include Header files
2. Modified variables.

Main function:

1. Initialized drawing configurations and the heat distribution.
2. Draw.
3. startTime
4. initial MPI
5. initialized local variables
6. except wall, calculate once base on last output.
7. barrier
8. MPI gather to global
9. Barrier
10. endTime
11. free variables, total, fin.

4. pthread.cpp

Before main function:

1. Include Header files and defined namespace.
2. Modified constant variables.

Main function:

1. Initialized drawing configurations and the heat distribution.
2. Draw.
3. startTime
4. pthread create
5. define local variables
6. except wall, calculate once base on last output.
7. barrier local
8. update the output base on local output
9. barrier local
10. endTime
11. pthread join
12. fin.

5. openmp.cpp

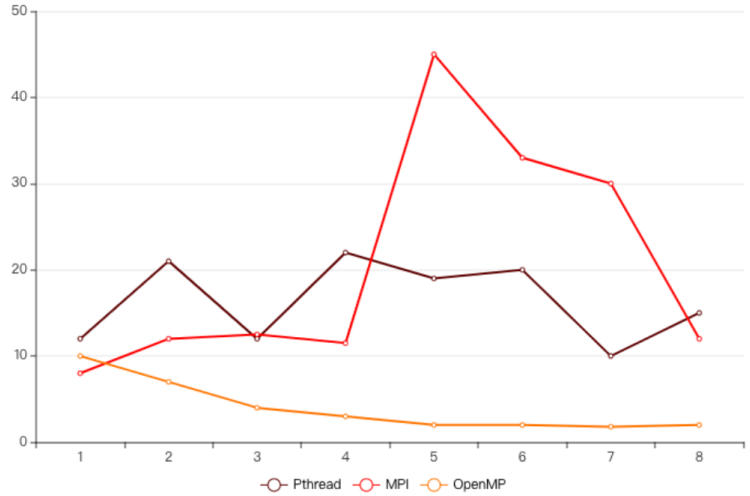
Before main function:

1. Include Header files
2. Modified variables.

Main function:

1. Initialized drawing configurations and the heat distribution.
2. Draw.
3. startTime
4. openmp, start parallel
5. initialize local variable
6. except wall, calculate once base on last output.
7. Omp barrier
8. Update
9. Omp barrier
10. Refresh the graph when master multiple iteration of 10
11. Omp barrier
12. Openmp end parallel
13. endTime
14. print output, fin.

Performance Analysis:

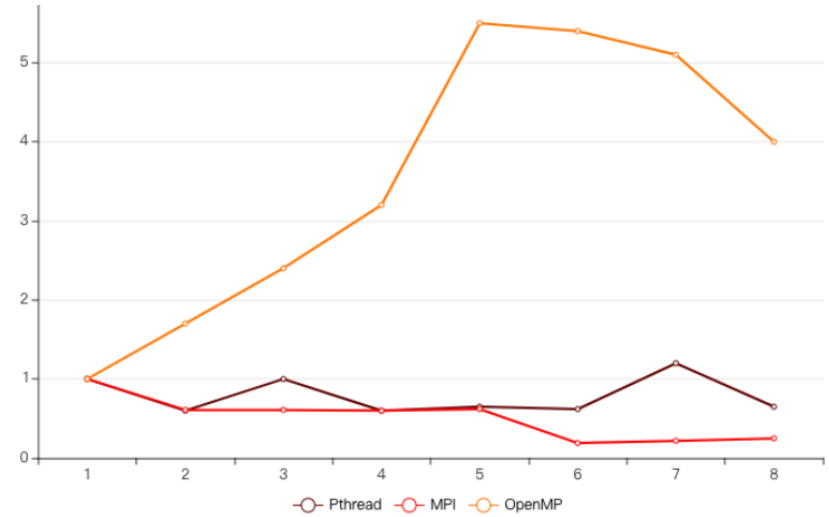


First, I should talk about the running time of each version.

For mpi, when the number of processes increase, the time of scheduling gain. On the other side, it went maximum and went lower again.

For pthread, running time is pretty stable than other version.

For openmp, when processes increase, running time is decrease.

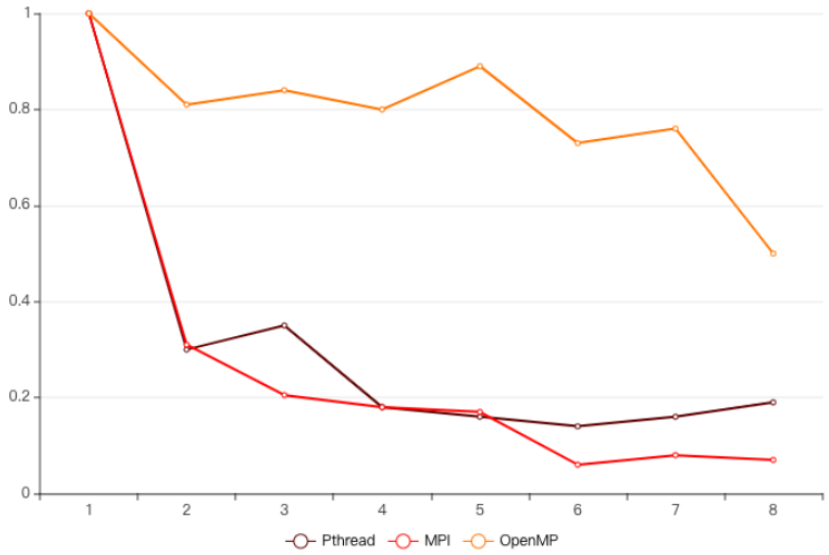


Second, should talk about the speedup feature.

For mpi, when processes increase, speedup is decrease.

For pthread, I don't know much detail about this but it's pretty stable about speedup as well.

For openmp, when the number of processes increase, speedup will be increase as well.



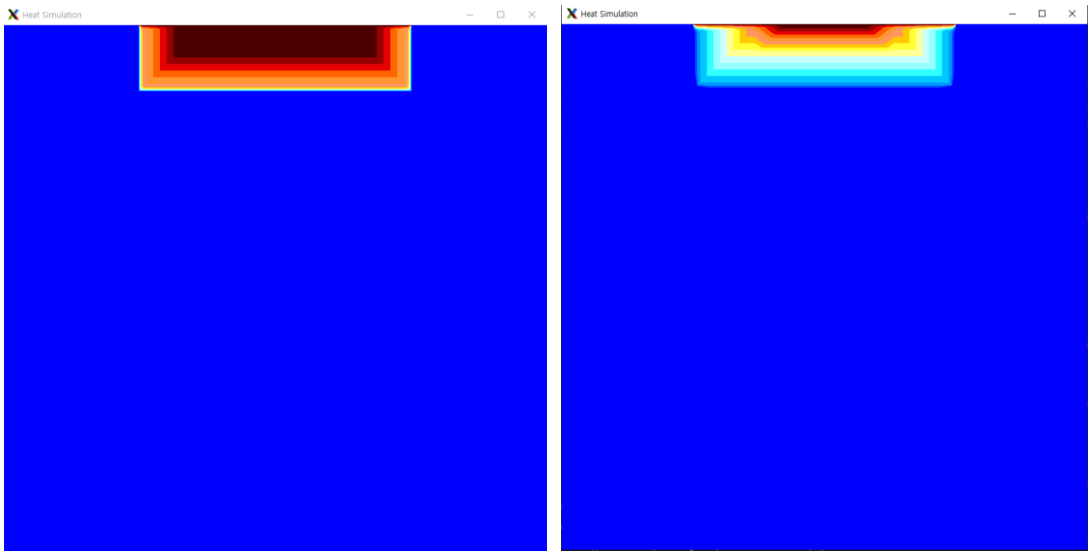
Finally, I should talk about the efficiency.

For mpi, when the number of processes increase, efficiency will go lower.

For pthread, it's mostly the same with mpi and when the number of threads is 4 or higher, it will be stable as well.

For openmp, when the number of threads increase, efficiency will go lower (but not that stable).

Output:



How to run a code (sequential.cpp):

To run the sequential version,

1. Compile it with below.
   - g++ -std=c++11 sequential.cpp -o sequential.out -lX11
2. Run it with below.
   - ./sequential.out 800 800 1000

How to run a code (MPI.cpp):

To run the MPI version,

1. Compile it with below.
   - mpic++ mpi.cpp -o mpi -lX11
2. Run it with below.
   - mpirun -n 4 mpi 800 800 1000

How to run a code (pthread.cpp):

To run the pthread version,

1. Compile it with below.
   - g++ -std=c++11 pthread.cpp -o pthread -lX11 -lpthread
2. Run it with below.
   - ./pthread 4 800 800 1000

How to run a code (openmp.cpp):

To run the openmp version,

1. Compile it with below.
   - g++ -std=c++11 openmp.cpp -o openmp -fopenmp -lX11
2. Run it with below.
   - ./openmp 4 800 800 1000