**Assignmen2 Report**

CSC4005 **Yu Chen**
Prof. Yeh-Ching Chung 115010124
TA: Hongliang Zhu & Peipei Zhu 8 November 2018

## Instruction

This assignment implemented three version of Mendelbrot Set figure drawing program. One is sequential program provided by instruction. The other two are parallel program using multiprocess by MPI library and multithread by pthread library. The complied executable file is in the cluster at direction of my student number, named as `serial`, `mpiMend`, and `pthreadMend`. For mpiMend, use command `mpiexec -n 4 ./mpiMend` to run the program with 4 process. The number of process can be changed by the parameter `-n`. The Then re-build by `mpicc`. The threads number used in `pthreadMend` can be changed by change the parameter `nthread`. (I am sorry for to hurry to encapsulate programs to pass the parameter by terminal.)

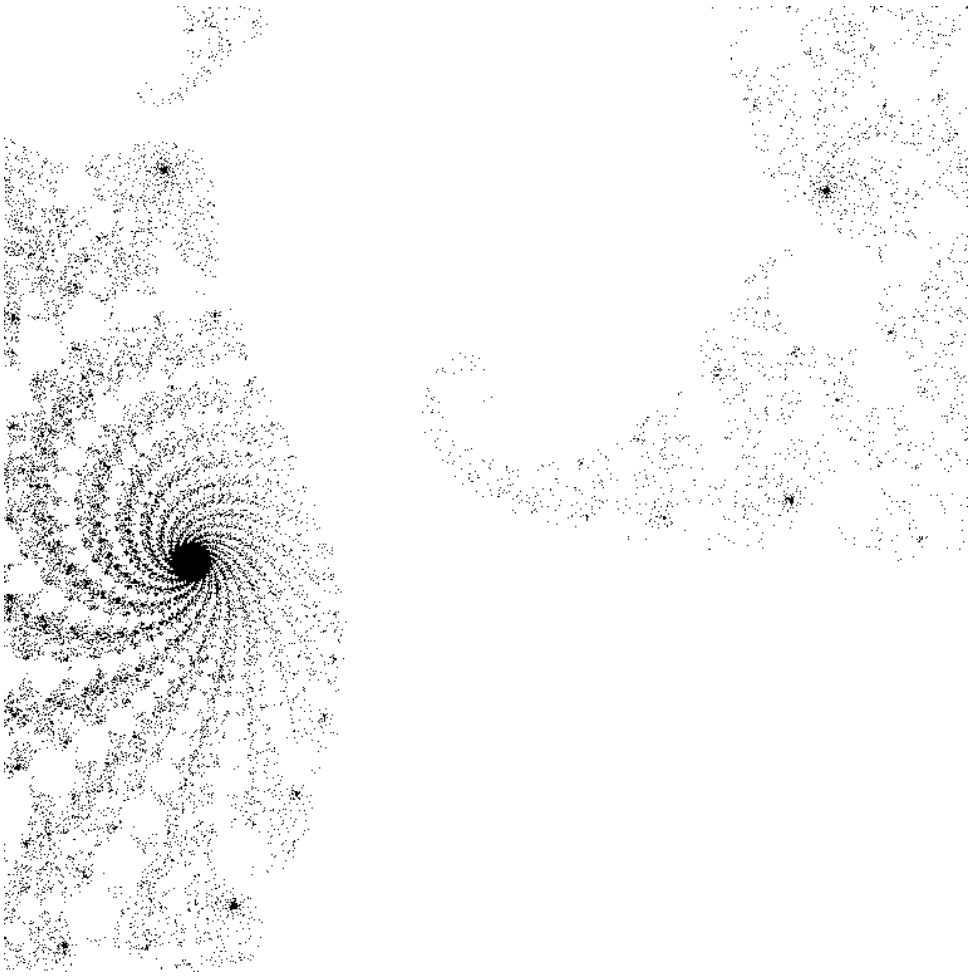

Figure 1: .

Fig. 1 is a sample output with a offset of (0.286932, 0.014287), scale factor of 800/160000= 0.0005 and 500 iteration upper limit. These parameters can be changed by constant value defined at beginning of each source code c file. The scale factor here is the "pixel" factor, so the

origin factor is 200. The running time of the program will show in terminal when the calculation of the picture finished.

## Design

The structure of sequential Mendelbrot program mainly consist by X11 part and point calculation part. The X11 part can be used in all three version. It generated a $800x800$ graphical window to show the result image. It sometimes refresh slower than calculation.

### MPI design

An array named `rowData` with length `X_RESN + 1`, the first `X_RESN` integers save the rowdata of whether the pixel reaches the upper bound. The last integer is the row number sending back to master to locate which row to print. Three types of tag is set to distinguish different data sending from master to slave. Tag 1 means a integer of the row number. Tag 2 is an array of the content in the row. Tag 3 is termination flag.

First, the master process sends n row number to n slave process to initialize the task assigning. After receiving the row number, the slave calculate the value of this row. Then save it into `rowData`. If the `rowData` is completed, slave sends it back and receives next data. If the receiving tag is 3 means all the tasks have done.

The master process has a counter to count whether all the rows have been collect. The row data received from slaves is printed to X11 immediately.

### Pthread design

First, initialize the pthread. In this task `pthread_mutex` is used to insure data correctness. (Actually, if no `pthread_mutex` there is segmentation fault.) A variable `startx` is used to save the row number of the next task.

Previous paragraph describe the main stream codes design. The mainly running codes is in the function invoked in each thread. The function `mandelbrot` is the threads function. It firstly check whether all the tasks are finished. If not, get the next task's column number.(I wanted to calculate the row at beginning, but mistake it as column. Thanks to the graph is a square). The thread calculate the column and draw on the figure. Then, update the `startx` when all other threads is locked.

## Experiment & Analysis

The performance analysis mainly includes different iteration under difference processes or threads. The offset and factor number effect the performance. The difference mainly cause by the number of "black" points. The best case is the whole figure is white and worse case is whole black. However, it is need further codes to count the actual imaginary number operation.

In this assignment, I haven't test for multiprocess for MPI program because some problem occurred when use processes more than 4. The test uses 4 process or threads, which is representative number of "multi", for MPI and pthread program.

The table. 1 is the performance test for three versions program. The process/thread used in MPI and pthread is 4. From table we can see that the MPI do not take advantage even at the end of table with a 10k iteration. Howevner, the speed up factor is increasing with iteration number, which is 0.84 for 10k iterations. Pthread program take advantage quickly when iteration reaches 5k. It has a great advantage to sequential and MPI program when

| | 100 | 500 | 1000 | 2000 | 5000 | 10000 |
|---|---|---|---|---|---|---|
| sequential | 0.09 | 0.21 | 0.36 | 1.16 | 2.25 | 6.01 |
| MPI | 1.38 | 1.59 | 1.85 | 2.03 | 5.75 | 7.08 |
| pthread | 0.07 | 0.34 | 0.62 | 1.04 | 1.55 | 2.73 |

Table 1: .

iteration is 10k, which actually is not a large number of iteration.The speed up factor is 2.20 for 10k iterations. The performance should further increase for the number really large.

## Experience

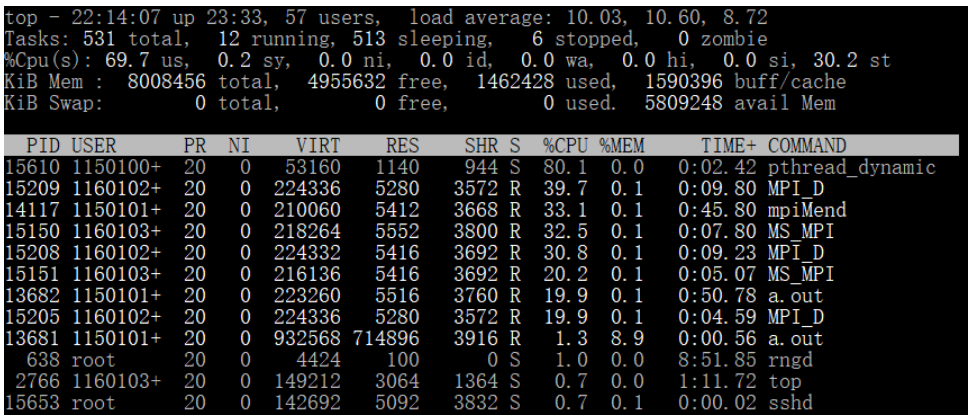


Figure 2: .

Due to load of the server. The program cannot always run with a full physical core. For example, in Fig. 2, my task `mpiMend` just has 33% core to run. However, some users can use more than one CPU of the server like Fig. 3. (I just really want to know how to do this with a normal user privilege.)

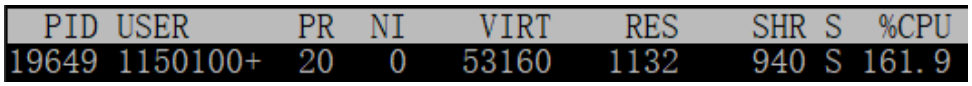


Figure 3: .

It is always that the figure printing time is much longer than running time in pthread program of this assignment. I cannot find out the reason for this. The time stamp of pthread is right. Compare with the running time of sequential and MPI program, the running time for pthread is also reasonable. Maybe there is some problem when X11 using multi-threads or I miss some sets in X11.

I do not implemented a version that can send more than 1 row each time. It may affect the performance due to large time of communication between processes for MPI. Theoretically, the performance should be better in this way for a small number of iteration.

## Source Code

The assignment is demo in a server. The source code for the submission time is compressed into a zip file and submitted to the Blackboard.

## References


[1] The Latex Template used for assignment is cite from overleaf. *https* : *//www.overleaf.com/latex/templates/ece* − 100 − *template/pjrrfybfggqt*

[2] The source code for sequential program is spread by the instructor from the site *http* : *//www.cs.nthu.edu.tw/ ychung/homework/para*$_p$*rogramming/mandelbrot.htm*