CSC4005: Distributed and Parallel Computing

Assignment2: Mandelbrot Set Computation

1 description

Set of points in a complex plane that are quasi-stable (will increase and decrease, but not exceed some limt) when computed by iterating the function:

$$z_{k+1} = z_k^2 + c \tag{1}$$

when z_{k+1} is the $(k+1)$th iteration of the complex number $z = a + bi$ and c is a complex number giving the position of the point in the complex plan. (For example, in a image of height H and width W, $c = \frac{x-height/2}{height/4} + \frac{y-width/2}{width/4} \times i$. You can also scale the c to obtain a different output).

The initial value for z_0 is zero. The iterations continued until the magnitude of z_k is greater than a threshold or the maximum number of iterations have been achieved.
For $z_k = a + bi$. The magnitude of z_k is defined below:

$$z_k = \sqrt{a^2 + b^2} \tag{2}$$

Computing the complex function $z_{k+1} = z_k^2 + c$ is simplfied by recognizing that:

$$z^2 = a^2 + 2abi + bi^2 = a^2 - b^2 + 2abi \tag{3}$$

Therefore, real part is the $a^2 - b^2$ while the imaginary part is $2abi$. The next iteration values can be produced by computing:

$$\begin{aligned} z_{real} &= z_{real}^2 - z_{imag}^2 + c_{real} \\ z_{imag} &= 2z_{real}z_{imag} + c_{imag} \end{aligned} \tag{4}$$

You need to design your own method to partition the image and assign pixels to different threads or processes.
For visualizing the Mandelbrot Set on an image, compute z_k for each pixel, if z_k is quasi-stable, draw this pixel on the display using Xlib.

2 requirement

- You need to implement two versions of the tasks, which are **MPI version** and a **Pthread version**. And hand in the codes for these two versions in two seperate code files. You need to print the following information for your codes including your **name, student id, assignment id, implementation version, running time of the program**.(see the folloing figure.)

- In your submit code, it should display an image with size of 200 × 200.
- Include the results in your report.
- You need to specify the **command line** about how to **compile** and **run** your program.
- You need to **compare the performance** of different implementation and configurations in your report.
  - The number of processes or threads used in the program (up to 16 processes and threads)
  - MPI vs Sequential vs Pthread
  - Size of the output images (three different sizes ranging from small, medium to large)
  - More if you have
- You need to include **two figures** describing the structure of your MPI program and Pthread program.
- The report should be written in appropriate format which you could refer to the report template.

3 Tips

- When measuring the running time of the program, make sure only measure the computation time. (Because Xlib's drawing is time consuming.)
- You should start your homework as soon as possible, do not try to finish it in the last two days before the deadline.
- Debug your program on the VM that is built on the image ubuntu.vdi. Make sure the program is OK, only after then measure the running time of the program on the server.
- Make sure use 'qsub' command to submit your jobs onto the master machine, do not directly run your program on the master machine.
- Try to limit your program running time within 60 seconds. If your program is running slow, try to improve your code or reduce the image's size.
- The 8 VMs on the server is now split into two cluster (each cluster has 20 cores), you can choose one of them to submit your job. See job_submission.pdf for more details.

4 Where and What to Turn in Your Homework

- Please turn in a paper includes
  - Report
  - Performance analysis
- zip your source codes and paper in a zip file, and name it studentID.zip, then submit it on Blackboard.
- Late submission penalty, 5 points deduction for each 12 hours after the deadline.

5 Due:

23:59,10/22/2019