Parallel Programming HW3
Manderlbrot Set

TA: Tony Wei

Due on 2014/12/14

1 Goal

This assignment is decided to help you get familiar with OpenMP library API. Besides, it also combines all the different memory architecture we have learned so far. In this assignment, you need to parallelize the sequential program of Mandelbrot Set by using

1. distributed memory – MPI
2. shared memory – OpenMP
3. hybrid (distributed-shared) memory – MPI + OpenMP

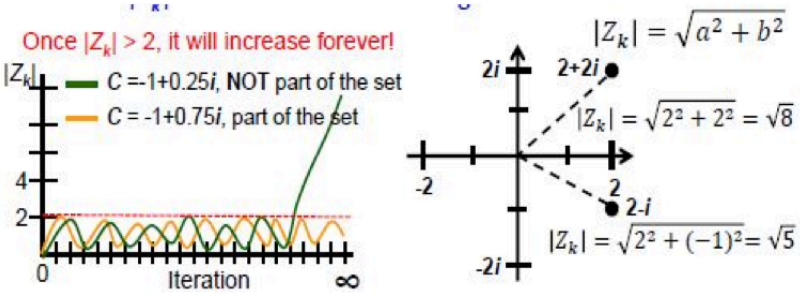
Another goal of this assignment is to help you to know the importance of load balance in parallel programming. So for, each different implementation, you need to write both the **static** and **dynamic** scheduling versions.
Finally, after you finish all your **6** implementations, you need to design a lot of experiments to analyze the performance of your program and write the report.

2 Problem Description

The Mandelbrot Set is a set of complex numbers that are quasi-stable when computed by iterating the function:

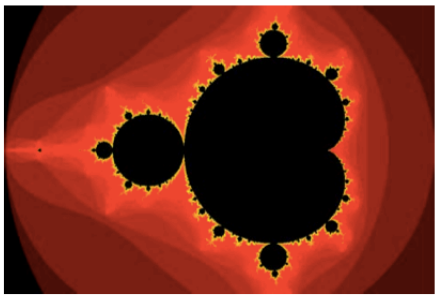
$$Z_0 = C, Z_{k+1} = Z_k^2 + C$$

- C is some complex number: $C = a + b_i$
- Z_{k+1} is the $(k+1)_{th}$ iteration of the complex number
- if $|Z_k| \leq 2$ for any k, C belongs to Mandelbrot Set



What exact is Mandelbrot Set?

- It is fractal: An object that display self-similarity at various scale; Magnifying a fractal reveals small-scale details similar to the large-scale characteristics
- After plotting the Mandelbrot Set determined by thousands of iteration:



For more information, please refer to the lecture notes.

3 Input Format

Input Parameters

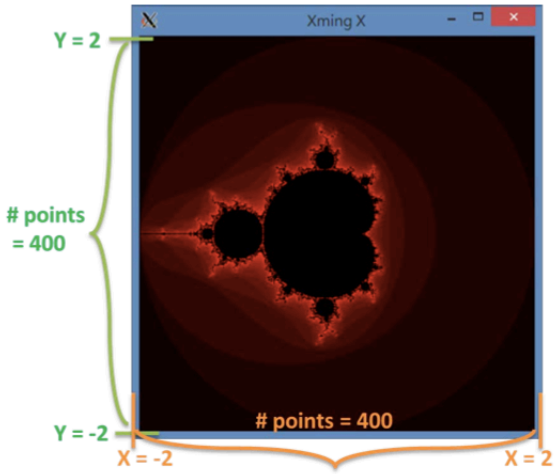
./executable ${1} ${2} ${3} ${4} ${5} ${6} ${7} ${8}

- ${1}: number of threads
  - P.S. For pure MPI version, this parameter will not be used, but please just give it a number for maintaining the same format
- ${2}: left range for x-axis: [double]
- ${3}: right range for x-axis: [double]
- ${4}: lower range for y-axis: [double]
- ${5}: upper range for y-axis: [double]
- ${6}: number of points in x-axis: [int]

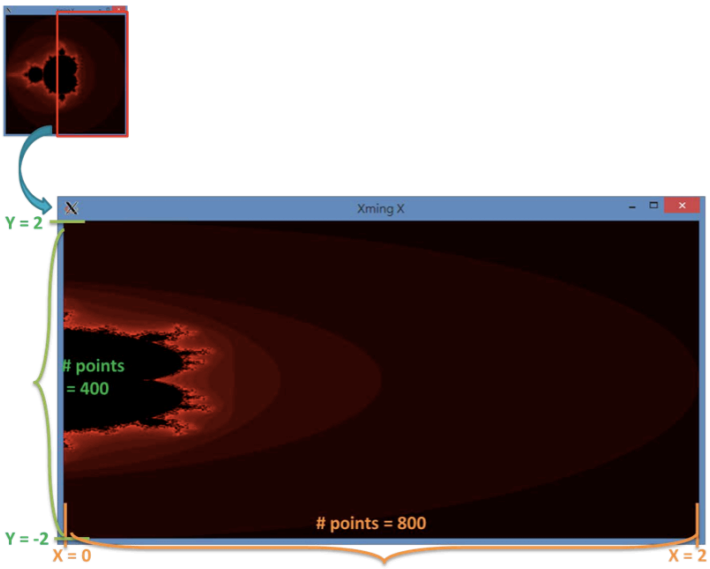
- ${7}: number of points in y-axis: [int]
- ${8}: display flag for Xwindow: [enable / disable]

4 Output Format

Example 1: **mpirun -n 2 ./MS_Hybrid_static 4 -2 2 -2 2 400 400 enable**



Example 2: **mpirun -n 2 ./MS_OpenMP_static 4 0 2 -2 2 800 400 enable**



5 Grading

You are required to implement **6** different version of Mandelbrot Set. Moreover, your grade will be judged by correctness, report, and demonstration as described in below:

5.1 Correctness (50%)

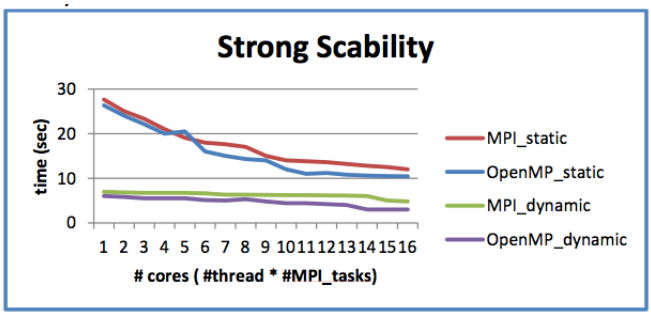
During the demo time, TAs will use a lot of different combinations of parameters to test your program, and check whether the output graphs are correct. Also, please make sure you follow the different memory architectures to implement your programs as described before.

The detailed score distribution of this part is:

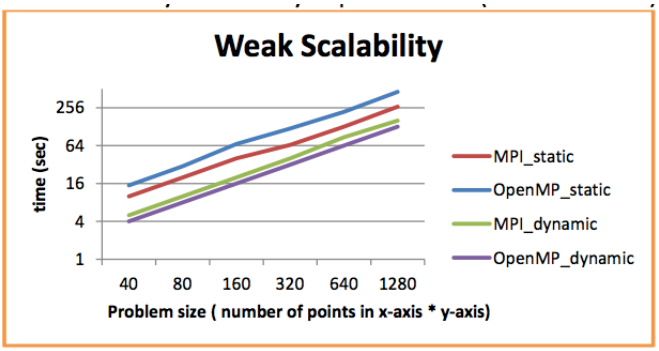
- MPI version: 15%
- OpenMP version: 15%
- Hybrid version: 20%

5.2 Report (30%)

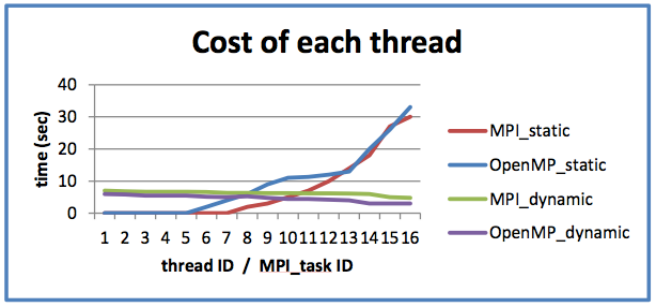
1. Design
   Explain your implementations, especially in the following aspects:
   (a) What the differences between the implementations of six versions
   (b) How do you partition the task
   (c) What technique do you use to reduce execution time and increase scalability
2. Performance analysis
   (a) Scalability chart – strong & weak scalability
   Please explain your graph. Do not just put the graph without any explanations!
   - Strong scalability – scalability to number of cores (Problem size is fixed.)

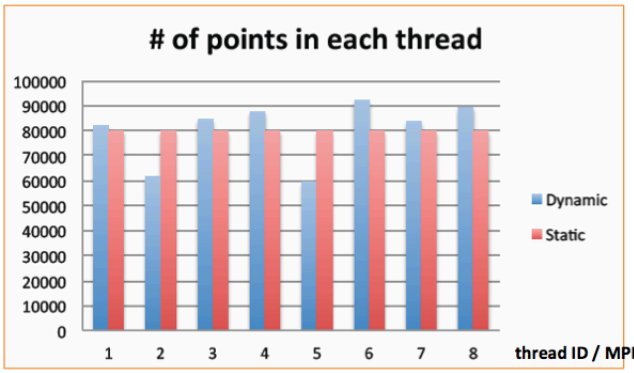


- Weak scalability – scalability to problem size (#cores is fixed.)



(b) Load balance chart





(c) Think as many interesting experiments as you can. For example,
- Whats the best distribution between MPI tasks & threads, why
- Whats the best distribution of cores between machine, why
- ...(The more, the best!)

3. Experience
   (a) What have you learned from this assignment?
   (b) What difficulty did you encounter when implementing this assignment?
   (c) If you have any feedback, you can also write it here.

5.3 Demo (%20)

6 Reminder

1. Late submission penalty policy please refer to syllabus.
2. Please compress all of your codes and report in the file named **${ID}_${name}_HW3.zip** and upload to iLMS before **12/14(Sun.) 23:59 pm**.
3. Because we have limited machines for you guys for tuning. Please start your work ASAP and do not leave it until the last day!
4. Please **disable** the display of Xwindow functions when submitting the job to computation nodes