CS542200 Parallel Programming
Homework 3: Mandelbrot Set

Due: December 13, 2015

1 GOAL

This assignment helps you getting familiar with OpenMP library API and knowing the differences between different memory architecture we have learned so far. Besides, this assignment also helps you knowing the importance of load balance. 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

For three different memory architecture mentioned above (MPI/OpenMP/Hybrid), you need to implement two different scheduling policies for each version:

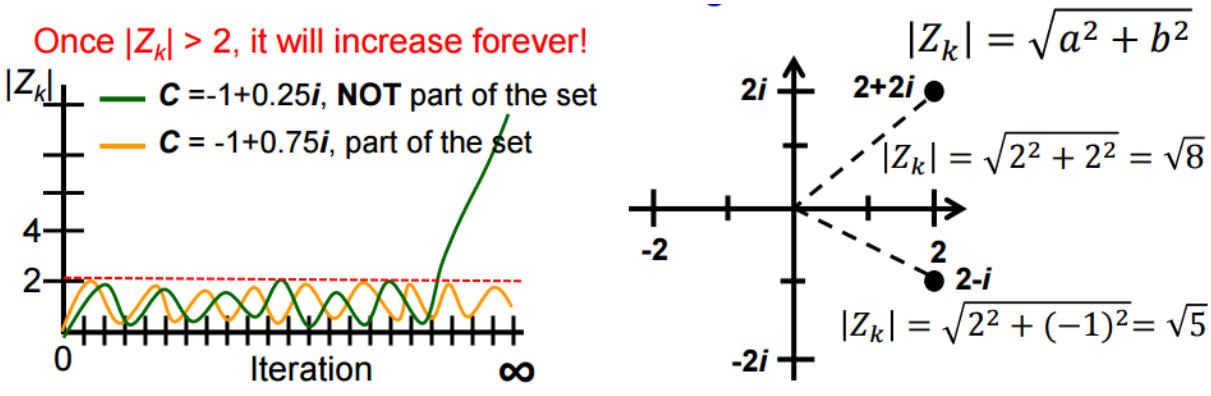
1. Static scheduling
2. Dynamic scheduling

2 PROBLEM DESCRIPTION

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

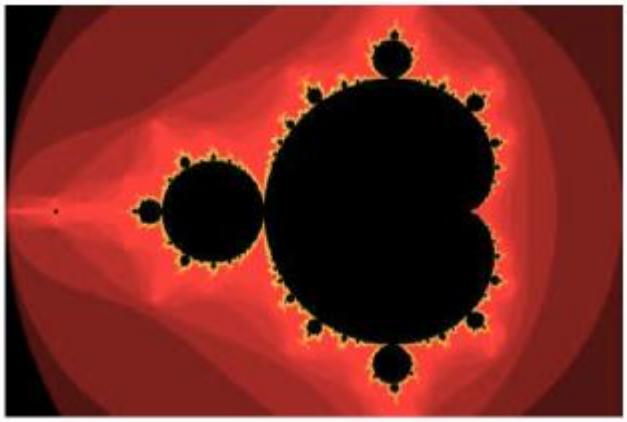
$$Z_0 = C, Z_{k+1} = Z_k^2 + C$$

- C is some complex number: $C = a + bi$
- 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 larger-scale characteristics
- After plotting the Mandelbrot Set determined by thousands of iterations:



For more information, please refer to lecture notes.

3 INPUT FORMAT

Input Parameters:

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

- ${1}: number of threads

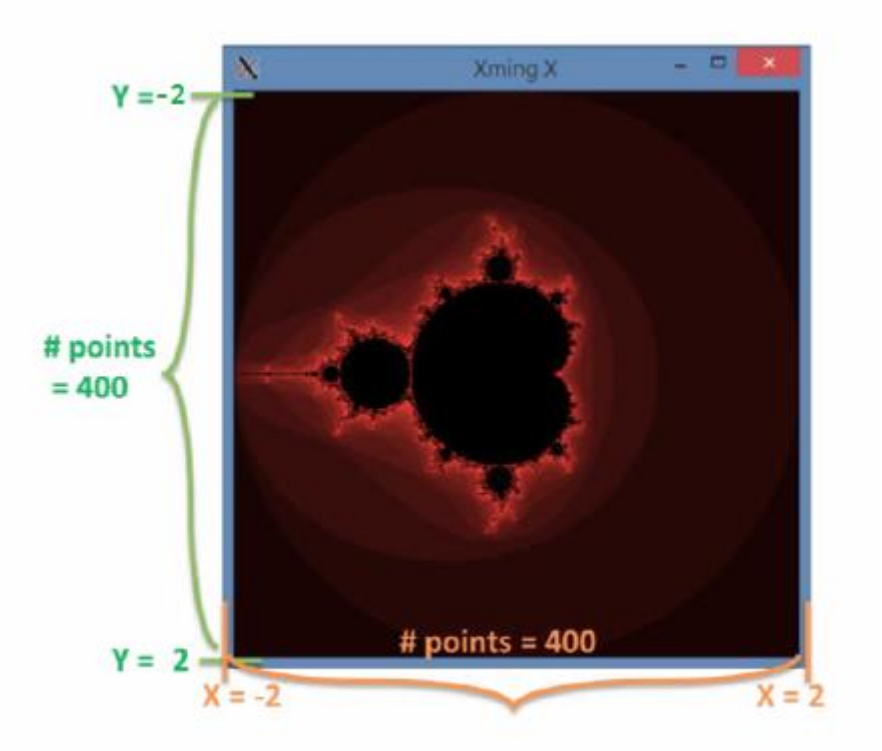
  (P.S For pure MPI version, this parameter will not be used, but please give it a number for maintaining the same input format)

- ${2}: left range of real-axis [double]
- ${3}: right range of real-axis [double]
- ${4}: lower range of imag-axis [double]
- ${5}: upper range of imag-axis [double]
- ${6}: number of points in x-axis [int]
- ${7}: number of points in y-axis [int]

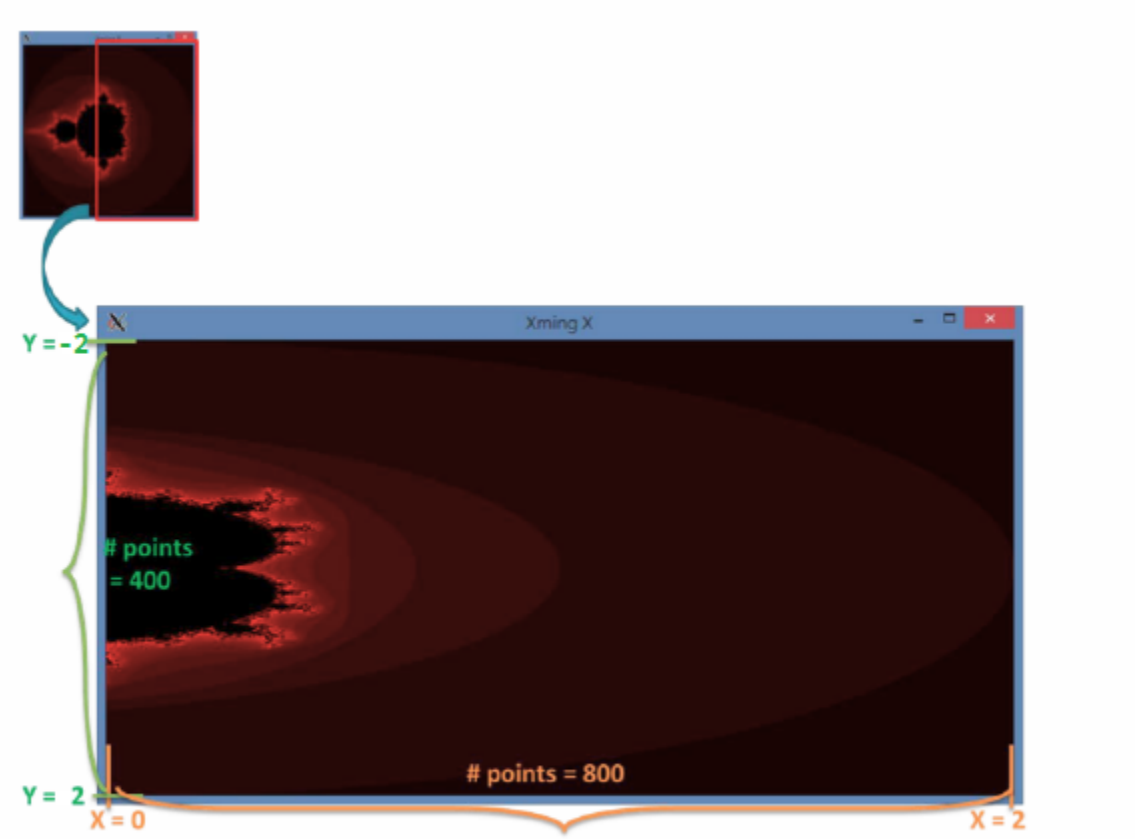
- ${8}: enable/disable Xwindow

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 **six** different versions of Mandelbrot Set. Moreover, your grade will be judged by correctness, report, and demonstration as described below:

5.1 CORRECTNESS (50%)

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

The detail score distribution of each 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 are the differences between the implementation of six versions?

   (b) How do you partition the task?

   (c) What technique do you use to reduce execution time and increase scalability?

   (d) Other efforts you've made in your program

2. Performance analysis

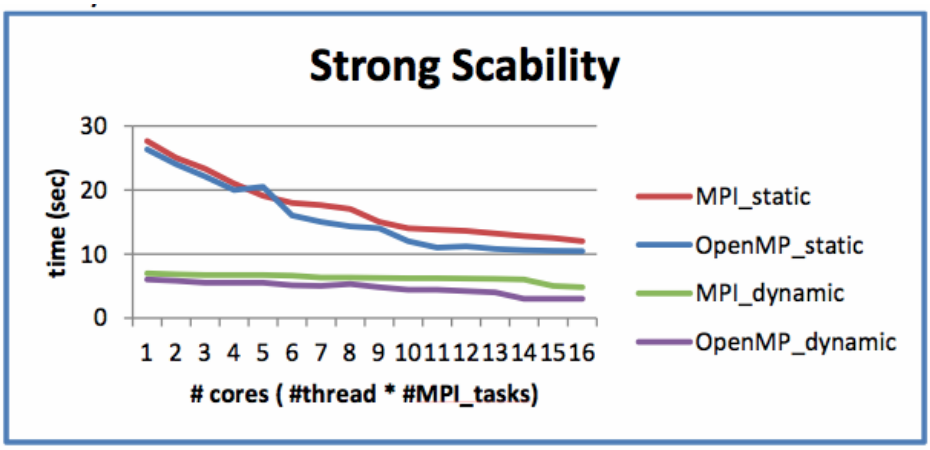
   Please fix your coordinate's range from (-2, -2) to (2, 2).

   (a) Scalability chart – strong & weak scalability

   Please explain your graph. Do not just put the graphs without any explanations!

   i. Strong scalability – scalability to number of cores (Problem size is fixed)

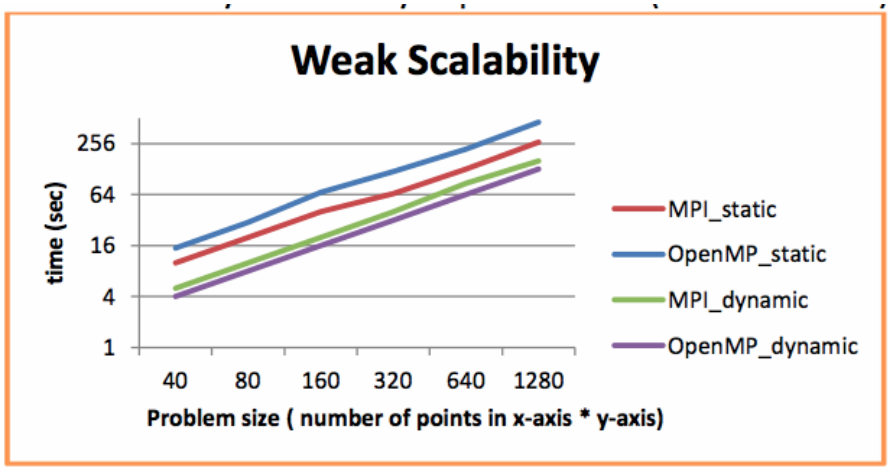
   {MPI, OpenMP}x{static, dynamic}



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

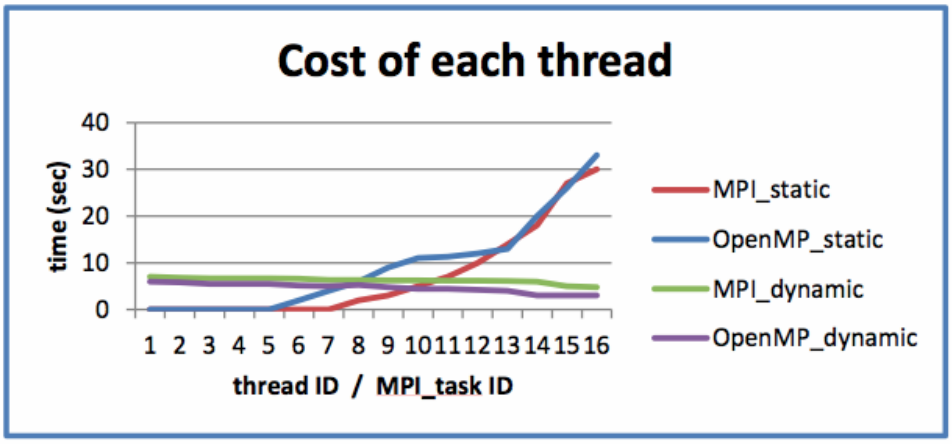
   {MPI, OpenMP, Hybrid}x{static, dynamic}

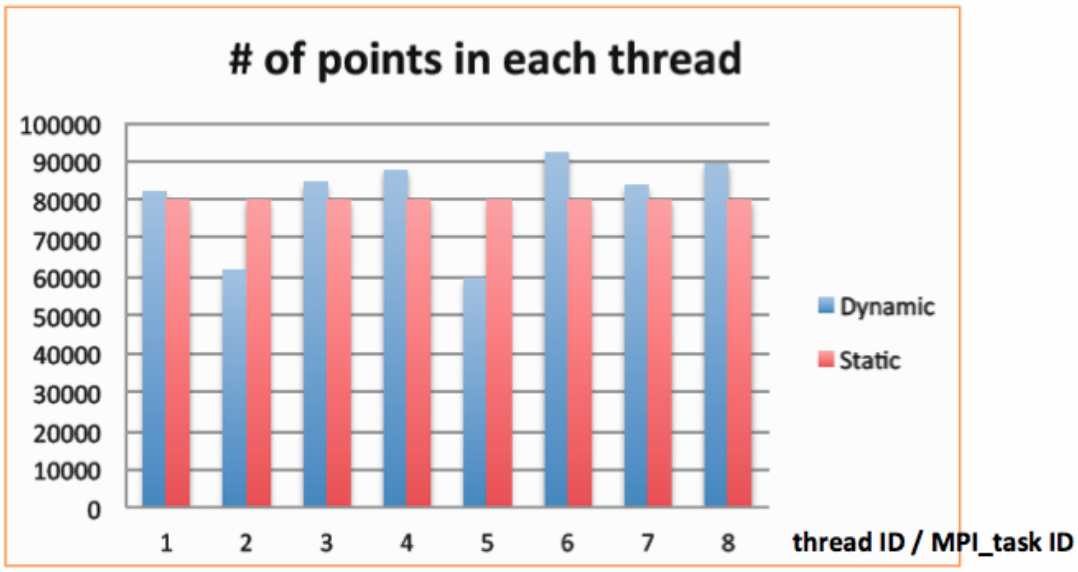
   You can use different number of points in x-axis and y-axis to change the problem size.



(b) Load balance chart

{MPI, OpenMP, Hybrid}x{static, dynamic}





(c) Conduct as many interesting experiments as you can. For example,

- What's the best distribution between MPI tasks & threads, why?
- What's the best distribution of cores between nodes, why?
- The more, the better!

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, please write it here.

5.3 DEMO (20%)

- Test the correctness of your codes.
- Go through your codes. Make sure your implementation follows the requirements.
- We will ask some questions about your codes and report.

6 REMINDER

1. Compile

   Example:

   ```
   mpicc -o MS_MPI_static -lX11 MS_MPI_static.c

   gcc -o MS_OpenMP_static -fopenmp -lX11 MS_OpenMP_static.c

   mpicc -o MS_Hybrid_static -fopenmp -lX11 MS_Hybrid_static.c
   ```

2. While running Hybrid versions, please modify the **job.sh** script we provide in **/home/pp2015/shared/hw3** to get better performance.

3. Please package your codes and report in a file named **HW3_{student-ID}.zip** and name your codes/report as follows:

   (a) Source code: **MS_${API}_${method}.c**

   - ${API} = MPI, OpenMP, Hybrid
   - ${method} = static, dynamic
   - Example: MS_MPI_static.c, MS_Hybrid_dynamic.c, ……

(b) Report: **HW3_${student_id}_report.pdf**

(c) Please pack your compile script or Makefile with your source code and report.

Make sure your compile script can execute correctly and your code has no compile error before you upload your homework, and upload to iLMS before **12/13(Sun) 23:59**

4. Please **disable** Xwindow while running experiments for the report.
5. Since we have limited resources for you guys to use, please start your work ASAP. Do not leave it until the last day!
6. Late submission penalty policy please refer to the course syllabus.
7. **Do NOT try to abuse the computing nodes by ssh to them directly**. If we ever find you doing that, you will get 0 point for the homework!
8. Asking questions through iLMS or email are welcomed!