Homework 1 – SF2568 Parallel Computations for Large-Scale Problems

Cuong Duc Dao, Donggyun Park
cuongdd@kth.se, donggyun@kth.se

9th February 2021

1. Describe the difference between a *process* and a *processor*

Solution:

- **A process**: is a computational activity assigned to a processor. It is defined by the program code and the data it works on.
- **A processor**: is a physical hardware device that can access memory and compute new data values.

2. A multiprocess consists of 100 processors, each capable of a peak execution rate of 2 Gflops (i.e. 2×10^9 floating operations per second). What is the performance of the system as measured in Gflops when 10% of the code is sequential and 90% is parallelizable?

Solution:
Gustafson's Law:

$$T_1 = f' T_P + (1 - f')\, P T_P \tag{1}$$

where T_P is the parallel execution time, P the number of processors and f' is the fraction of the code that is sequential. This results in the following:

$$T_1 = 0.1 \times 2 \times 10^9 + 100\,(1 - 0.1) \times 2 \times 10^9 = 180.2 \quad \text{Gflops} \tag{2}$$

3. Is it possible for a system to have a system efficiency η_P of greater than 100%?

Solution:
According to Gustafson's Law, the scaled speedup of P processors is:

$$S'_P = f' + (1 - f')P \tag{3}$$

Supposed that our parallelization is perfect(a given task can be fully parallelized), i.e., $f' = 0$, then $S'_P = P$. The system efficiency is

$$\eta_P = \frac{S'_P}{P} = \frac{P}{P} = 1 \tag{4}$$

This is the upper bound of η_P and therefore it cannot be greater than 100%

4. In the Parallel Rank Sort method presented in the lecture, the number of processors must be the same as the number of elements in the list. Assume that the number of elements in the list is actually ten times larger than the number of processors in the computer. Describe a modification to handle this situation

Solution:

We can divide the list into the number of processors, which means 10 sorting tasks are assigned to each processor. Then the speed up would be P/10.

5. You are given some time on a new parallel computer. You run a program which parallelizes perfectly during certain phases, but which must run serially during others.

   (a) If the best serial algorithm runs in 64s on one processor, and your parallel algorithm runs in 22s on 8 processors, what fraction of the serial running time is spent in the serial section?

   (b) Determine the parallel speedup.

   (c) You have another program that has three distinct phases. The first phase runs optimally on 5 processors; this is, performance remains constant on 6 or more processors. The second phase runs optimally on 10 processors, and the third on 15. The phases consume 20%, 20%, and 60% of the serial running time, respectively. What is the speedup on 15 processors?

Solution:

(a) he speedup from Gustafson's Law:

$$S'_P = f' + (1 - f')P \tag{5}$$

From here,

$$\frac{64s}{22s} = f' + (1 - f') * 8 \tag{6}$$

Therefore, we get

$$f' = \frac{8}{11} \approx 0.73 \tag{7}$$

(b) The parallel speedup is:

$$S_P = \frac{T_S}{T_P} = \frac{64}{22} \approx 2.91 \tag{8}$$

(c) Let's say a given fraction f for the serial part of the program. Then we can express S_P as following:

$$S_P = \frac{T_S}{T_P} = \frac{T_S}{T_S * f + T_S * (1 - f)(0.2 * \frac{1}{5} + 0.2 * \frac{1}{10} + 0.6 * \frac{1}{15})} = \frac{1}{f + \frac{1}{10} * (1 - f)} = \frac{10}{1 + 9f} \tag{9}$$

6. For the following problem, you need to have access to MPI on one of the platforms (preferably `tegner`). Implement the Mandelbrot algorithm in a parallel environment!

   (a) Implement the Mandelbrot program using MPI. You can assume that the number of processors divides the number of columns evenly.

   (b) Reproduce the figure from the lecture notes.

   (c) Magnify some interesting parts of the figure. Do this by computing only parts of the complete picture using higher resolutions.

Solution:

(a)

We have decided to use 'MPI_Gather' function to merge each partition from the slave processor since it does not make any difference from using 'MPI_Send' and 'MPI_Recv' functions when you produce an image.

As the number of pixels is 2048 by 2048, 8 processors are used in our implementation in order to ensure that the number of columns are divided to be evenly distributed to each slave processor.
The source code is not included in this report. *.c* file for computations using MPI has been submitted.

(b)

The figure from the lecture note can be reproduced when there are no offsets in the xy-coordinate. The pixel values of the black part are all 255 and the red part is 1 or 2 in the form of a circle(relatively small numbers, it is not observable with naked human eyes).

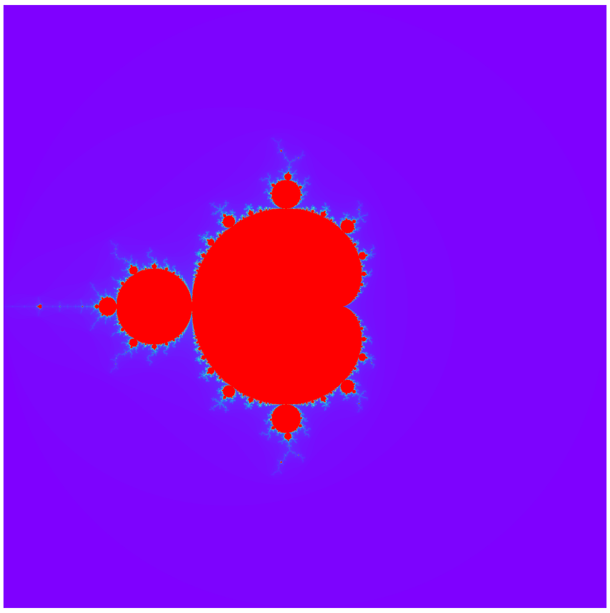

Figure 1: Original Mandelbrot set visualization, $N = 256; b = 2.0; w = 2048; h = 2048$

The second figure is produced with the offset added to each coordinate. The pattern becomes shifted as expected. This factor is going be used in part(c) to observed magnified complex boundaries.

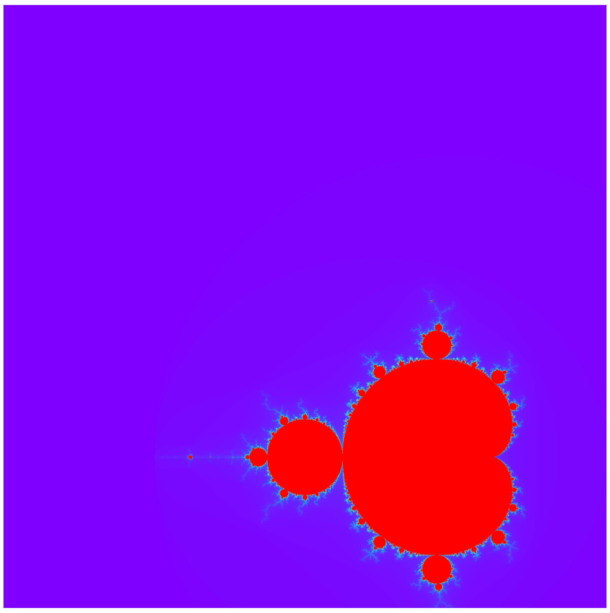

Figure 2: Shifted version $x_{\text{offset}} = -1.0; \quad y_{\text{offset}} = -1.0$

(c)

We have reproduced the image with the same number of pixels but smaller increments and magnified it 40 times. It shows the details of the complex boundary of the Mandelbrot set.

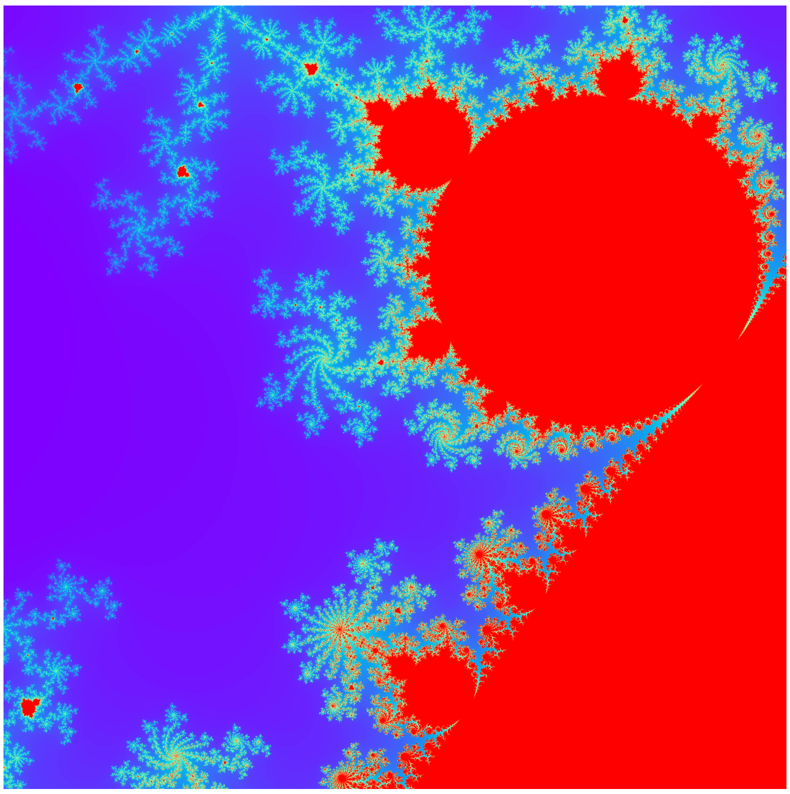

Figure 3: Magnified version: magnified by 40 times; $x_{\text{offset}} = 1.302$; $y_{\text{offset}} = 1.542$