Efficient Generation of Mandelbrot Set using Message Passing Interface

28993373
Samarasekara Vitharana Gamage, Bhanuka Manesha
School of Information Technology
Monash University
Malaysia
bsam0002@student.monash.edu

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The Mandelbrot set, named after Benoit Mandelbrot is the set of points c in the complex plane which produces a bounded sequence with the application of equation 1 repeatedly to the point $z = 0$ [1].

$$f(z) = z^2 + c \tag{1}$$

So in order to generate the Mandelbrot set for a $m \times n$ image I, we need to execute equation 1 on each pixel. So we derive the following equation.

$$M_{i,j} = z_{i,j}^2 + c_{i,j} \tag{2}$$

where $i = 1...m, j = 1...n$

II. Preliminary Analysis

A. Parallelizability of the Generation of Mandelbrot Set

In order to ensure that the process of generating the Mandelbrot Set is data parallalizable, we can use Bernstein's conditions [2]. Bernstein's condition is a simple verification for deciding if operations and statements can work simultaneously without changing the program output and allow for data parallelism [3]. According to Bernstein, if two processes satisfy the following equations, then they are parallalizable.

$$I_1 \cap O_2 = \emptyset \tag{3}$$

$$I_2 \cap O_1 = \emptyset \tag{4}$$

$$O_1 \cap O_2 = \emptyset \tag{5}$$

I_0 and I_1 represents the inputs for first and second process while O_0 and O_1 represents the outputs from first and second process. Equation 3 also known as anti in-dependency, states that the input of the first process should not have any dependency with the output of the second process, while equation 4, also known as flow in-dependency, states that the input of the second process should not have any dependency with the output of the first process. And finally equation 5 also known as output in-dependency, states that both the outputs of the two processes should not be equal.

Using equation 2 we can derive the input and output equations for any two neighboring pixels which will be computed by two processes P_1 and P_2.

$$I_1 = z_{i,j}^2 + c_{i,j} \tag{6}$$

$$O_1 = I_{i,j} \tag{7}$$

$$I_2 = z_{i,j+1}^2 + c_{i,j+1} \tag{8}$$

$$O_2 = I_{i,j+1} \tag{9}$$

Applying Bernstein conditions for equations 6, 8, 7 and 9, we get,

$$z_{i,j}^2 + c_{i,j} \quad \cap \quad I_{i,j+1} = \emptyset \tag{10}$$

$$z_{i,j+1}^2 + c_{i,j+1} \quad \cap \quad I_{i,j} = \emptyset \tag{11}$$

$$I_{i,j} \quad \cap \quad I_{i,j+1} = \emptyset \tag{12}$$

Since all the above conditions are satisfied, we can state that the generation of Mandelbrot Set satisfies the Bernstein conditions, thereby the data can be parallelized and computed concurrently.

Next we calculate the theoretical speed up to determine whether there is a benefit of running the code in parallel.

B. Theoretical Speed Up of the Generation of Mandelbrot Set

To determine the theoretical speed up of the Mandelbrot Set Generation, we first determine the pararalizable portion of the serial implementation of the code. This can be achieved by calculating the total time for generating the Mandelbrot set (t_p) and the total time to execute the whole serial code (t_t). Then using equation 13 we can get the r_p value.

$$r_p = \frac{\text{time taken for parallelizable portion of the code}}{\text{total time for serial execution}} \tag{13}$$

Then after calculating the r_p value we can use Amdahl's Law () to calculate the theoretical speed up value to generate the Mandelbrot set. Amdah stated that if p is the number of processes, r_s is the time spent on the serial part of the code and r_p is the amount of time spent on the part of the program that can be done in parallel, () then the speedup can be stated as

$$S_p = \frac{1}{r_s + \frac{r_p}{p}} \tag{14}$$

So using Amdah's law, we are able to generate the following chart with the theoretical speed up values.

TABLE I
NUMBER OF PROCESSES VS SPEED UP FACTOR

Number of Processes	Speed Up Factor
4	3.99968
8	7.9985
16	15.99378
32	31.9743
64	63.89566
∞	38580.2469

So having calculated the theoretical speed up, next we look into the design of the partition scheme for parallelizing data.

III. DESIGN OF PARTITION SCHEMES

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Naive Row Segmentation-based Partition Scheme

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. First Come First Serve Row Segmentation-based Partition Scheme

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

C. Improving III-B with MPI PACK

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

D. Optimizing III-B with Mandelbrot Symmetry Property

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

E. Implementation of Partition Schemes

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{15}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(15)", not "Eq. (15)" or "equation (15)", except at the beginning of a sentence: "Equation (15) is . . ."

F. Results and Discussions

G. Conclusion

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

REFERENCES

[1] C. Clapham and J. Nicholson, "Mandelbrot set," 2014.
[2] A. J. Bernstein, "Analysis of programs for parallel processing," *IEEE Trans. Elec. Comp. EC*, vol. 15, pp. 746–757, 1996.
[3] P. Feautrier, *Bernstein's Conditions*, pp. 130–134. Boston, MA: Springer US, 2011.

—c—l—l—l—l—l—l—l— height

Computer Specification
Value of iYmax
Value of iXmax
Value of IterationMax
2*   2***Serial Program**   **Parallel Program**
2   **4**   **6**   **8**   **16**   **32**
Run #1   300000   300000   300000   300000   300000   300000   300000
Run #2   300000   300000   300000   300000   300000   300000   300000
Run #3   300000   300000   300000   300000   300000   300000   300000
Run #4   300000   300000   300000   300000   300000   300000   300000
Run #5   300000   300000   300000   300000   300000   300000   300000
Average   300000   300000   300000   300000   300000   300000   300000