Simulation and Analysis of Wireless Sensor Network using Message Passing Interface

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

***Abstract*—With the increasing need for safer and reliable systems, Mandelbrot Set's use in the encryption world is evident to everyone. This document aims to provide an efficient method to generate this set using data parallelism. First Bernstein's conditions [?] are used to ensure that the Data is parallelizable when generating the Mandelbrot Set. Then Amdhal's Law is used [?] to calculate the theoretical speed up, to be used to compare three partition schemes. The three partition schemes discussed in this document are the Naïve Row Segmentation, the First Come First Served Row Segmentation and the Alternating Row Segmentation. The Message Parsing Interface (MPI) library in C [?] is used for all of the communication. After testing all the implementation on MonARCH, the results demonstrate that the Naïve Row Segmentation approach did not perform as par. But the Alternating Row Segmentation approach performs better when the number of tasks are < 16, where as the First Come First Served approach performs better when the number of tasks is ≥ 16.**

***Index Terms*—mandelbrot set, data parallelism, row segmentation, alternative, naive, first come first served, MonARCH, MPI, distributed computing**

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 $z = 0$ [?]. Aside from the inherent beauty of the Mandelbrot Set, some of the mysteries of this set is still unknown to Mathematicians [?]. So having a way to generate this set efficiently can increase the research in this area and improve our understanding of this set.

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

In order to generate the Mandelbrot set for a $m \times n$ image I, we need to execute equation 1 on each pixel of the image. The complex part of the equation is determined using the m and n values of the image. Then the Iteration at which either the magnitude (z^2) exceeding the escape radius or the *iterationMax*, is recorded. Both the *iterationMax* and Escape Radius are pre-determined and can affect the time taken to generate the set. The iteration is then used to determine the pixel color in the image I. Therefore each pixel of the image can be represented using equation 2.

This method is known as Mandelbrot generation with Boolean escape time. Figure **??** shows the image of a Mandelbrot Set generated using this method.

$$I_{i,j} = \mathit{Iteration}_{i,j} \tag{2}$$

$$\text{where } i = 1...m, j = 1...n$$

There are many uses of the Mandelbrot Set in the field of encryption such as Image Encryption [?], Key Exchange Protocols [?] and Key Encryption [?]. With the increasing need for cyber-security in the modern era, safer and reliable encryption techniques will play a major role in the future.

This report aims to provide an efficient partition scheme to generate the Mandelbrot Set. For this, three partition schemes are compared and contrasted against each other and then ranked based on their performance. First, Bernstein's conditions are used to determine whether the generation of Mandelbrot Set is data parallelizable [?]. Then the theoretical speed up is calculated using Amdhal's law [?], which is used as the objective for each partition scheme. The workload is divided among N number of tasks, in different segmentation configurations for each of the partition schemes. Then the actual speed up is obtained for all of the partition schemes with different task configurations. Using the theoretical speed up and the actual speed up, the percentage difference of each of the partition schemes are then obtained and used for the comparison. All of the inter process communication is done using the Message Parsing Library (MPI) in C [?].

II. Theoretical Analysis and Inter Process Communication Design

III. Results and Discussion

IV. Conclusion

Now that we have analyzed all the cases, we can rank each of the partition schemes based on their performance.

Referring to Figure **??**, the Naïve Row Based Segmentation performed the worst with increasing percentage difference at each doubling of the number of tasks except when the number of tasks were two. Next, the First Come First Served partition scheme performed on average for most of the test cases but performed the best when the number of tasks were

> 16. Finally, the Alternating Row Based partition scheme performed the best when the number of tasks > 16.

The speed up differences between each partition scheme can be explained due to the bottle necks and communication overhead of each approach. The partition scheme that is able to reduce each of the factors is able to perform better than the rest.

Conclusively, we can state that First Come First Served partition scheme is effective when the number of tasks are high while the Alternative partition scheme is effective when the number of tasks are low.

V. Future Work

This report opens up future research paths for each of the three methods. The effect of having the number of tasks greater than 32 can be a one of areas that can be experimented in the future. Another path is removing the assumption of having tasks as multiples of two and having an odd number of tasks. This will allow us to observe how the remainder affects each of the methods, especially Naïve and Alternative since the workload will be imbalanced for some nodes. This reports paves the way to exploring the efficient ways of generating the Mandelbrot Set in parallel architectures.

Appendix