COMP90025 Project 1C: MPI and Mandelbrot Set

San Kho Lin (829463) — sanl1@student.unimelb.edu.au

1 Introduction

In this project, I am going to experiment MPI parallel programming on a given sequential program which is counting the total sum of complex points that fall under a Mandelbrot Set.

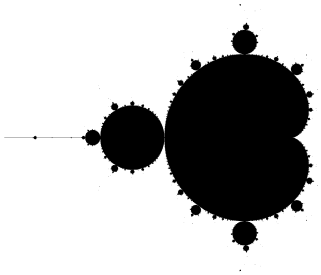
2 Sequential Optimization

In Project 1A[4], I discussed that the substantial amount of Mandelbrot Set computation time fall on to the sequential portion of it – of which, it can not be parallelized. In this project, I also focus my study on the *Escape Time Algorithm*[7] that given in the sequential program. There are many heuristics as well as formal methods and approaches [6] [7] [2] available for optimizing Mandelbrot Set computation. I realize that this lead me to further reading on understanding the problem domain – Fractal Maths in broader aspect. Since the main objective of the project is MPI parallel programming, I only try the following approaches.

2.1 Cardioid Bulb Checking

This technique check whether the given coordinate fall within the cardioid or in the period-2 bulb before passing to the ETA[1]. Typically it takes maximum iteration to detect for those points that are in the "black areas" (Figure 1). In these areas, the iterations never escape to infinity.

Figure 1: Mandelbrot Set



The following equation determine if the point lies inside the main cardioid in advance[7].

[1]Escape Time Algorithm

$$q = (x - \tfrac{1}{4})^2 + y^2, q(q + (x - \tfrac{1}{4})) < \tfrac{1}{4}y^2.$$

Similarly, the following equation is used for the period-2 bulb. x and y are the real and imaginary coordinate of the complex plane, respectively.

$$(x + 1)^2 + y^2 < \tfrac{1}{16}$$

2.2 Periodicity Checking

In this online article[3], the author states that many points within the Mandelbrot Set eventually reach periodic orbits, i.e. they converge to a sequence of values that repeats as the *square-and-add operation* is done. This repetition can be detected and the calculation can be stopped early. Therefore, the sequential Escape Time Algorithm is altered to add the following periodic checking.

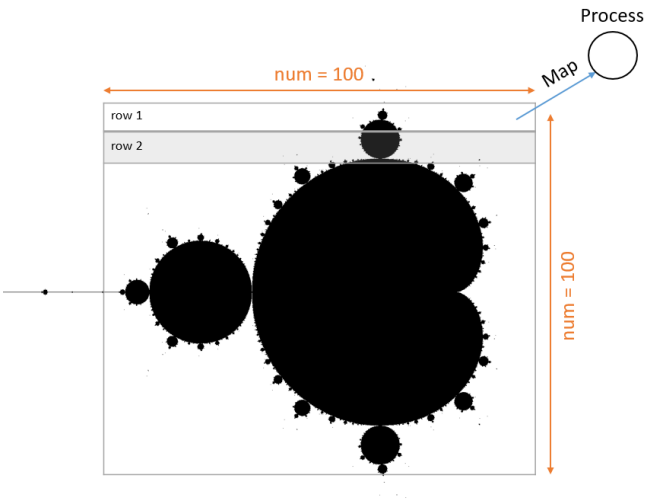
- Record a value to test against (trade-off memory for computation)
- Test the next n iterations against the recorded number
- If they become equal, it is in a periodic loop
- Otherwise record a new number and double the number of iterations to test
- If it ever enters into a loop, it will be found

I use a combination of both approaches[1] for highly optimized sequential time. The runtime improvement is quite apparent (Listing 1) before parallelize the computation.

3 Parallelizing the Load

As discussed in Project 1A, the main challenge for parallel programming the Mandelbrot Set is load balancing. For MPI, I experiment the techniques from the text book *Parallel Programming, Wilkinson and Allen*[8]. It is similar in concept such that **Static Task Assignment** and **Dynamic Task Assignment**. Either approaches, I partition the task into row wise decomposition (Figure 2). The number of problem size is known and it is proportion to the number of processes/tasks.

Figure 2: Row Wise Decomposition



3.1 Static Task Assignment

Contrary to the text book[8], I do not need to use **Master** and **Slave** process for static task assignment. Each processor find out its own group of rows based on rank. Then the processes compute their range independently. At the end, one process, for example the processor with rank 0, performs the `MPI_Reduce()` reduction with `MPI_SUM` operation to add up the total count. This approach is better in such that it saves the communication cost. Since the Set is not uniform growth by nature, some processes will complete the task faster and prone to become idle. However, this issue can mitigate to sudden extent by randomizing the group of rows[5] that assign to each processes.

3.2 Dynamic Task Assignment

It is also known as **Work Pool** or **Processor Farms**. It has the **Master** process who control the task assignment and global operations such as broadcasting parameters, adding up the count and so on. The **Slave** or **Worker** processes will just wait the message passing from the Master and perform heavy lifting computation task on the receiving parameter. After completed, it send the result back and ask for more task/load. This approach is more communication intensive than the static task assignment. In order to save communication start up cost, it has to be careful with partition and the amount of task that send to worker processes.

4 Conclusions

Run time breakdown table in Figure 3 summarise the total run time on number of nodes against number of tasks for the problem size of 10000 for given 2 regions. In summary the use of multiple cluster nodes have to consider for the communication cost when programming parallel implementation. This communication cost is not consider in the theoretical PRAM model. It also has to consider optimization of sequential part of the problem domain in order to achieve speed up.

5 References

[1] Simpsons contributor. *Highly optimized Mandelbrot set processor*. URL: `https://en.wikipedia.org/wiki/User:Simpsons_contributor/periodicity_checking` (visited on 10/06/2016).

[2] V. Drakopoulos. *Comparing Rendering Methods for Julia Sets*. Department of Informatics and Telecommunications, Theoretical Informatics, University of Athens. URL: `http://cgi.di.uoa.gr/~vasilios/DRA02.pdf` (visited on 10/06/2016).

[3] Lockless Inc. *The Mandelbrot Set*. URL: `http://locklessinc.com/articles/mandelbrot/` (visited on 10/06/2016).

[4] San Kho Lin. *COMP90025 Project 1A: OpenMP and Mandelbrot Set*. COMP90025 2016 SM2: Parallel and Multicore Computing, Continuous Assessment. 2016.

[5] Norm Matloff. *Programming on Parallel Machines, University of California, Davis. GPU, Multicore, Clusters and More*. eBook. URL: `http://heather.cs.ucdavis.edu/parprocbook` (visited on 10/06/2016).

[6] Robert P. Munafo. *Mandelbrot Set - Speed Improvements*. 1999. URL: `http://www.mrob.com/pub/muency/speedimprovements.html` (visited on 10/06/2016).

[7] Wikipedia. *Mandelbrot Set Optimizations*. URL: `https://en.wikipedia.org/wiki/Mandelbrot_set#Optimizations` (visited on 10/06/2016).

[8] Barry Wilkinson and Michael Allen. *Parallel Programming, Techniques and Applications Using Networked Workstations and Parallel Computers 2nd Edition*. Prentice Hall, 2005. ISBN: 0-13-140563-2.

6 Appendix

Listing 1: Sequential Non-optimized Vs Optimized

```
$ time ./non_optimized.exe -2.0 1.0 -1.0 1.0 100 10000 -1 1.0 0.0 1.0 100 10000
2538
3509

real    0m0.200s
user    0m0.000s
sys     0m0.000s

$ time ./optimized.exe -2.0 1.0 -1.0 1.0 100 10000 -1 1.0 0.0 1.0 100 10000
2538
3509

real    0m0.045s
user    0m0.000s
sys     0m0.000s
```

Figure 3: Run Table

Problem Size = 10000, Regions = 2

1-node

task	second
2	37.91188
8	4.936066
16	2.439032

2-nodes

task	second
8	5.506166
16	2.402512
32	1.502287
48	1.04373

4-nodes

task	second
8	8.8874
16	2.443049
32	1.460142
48	1.375807
64	1.056015

8-nodes

tasks	second
8	5.089587
16	2.563241
32	1.473531
48	1.377355
64	1.290683
96	1.312298
128	1.508471

```
srun mandelbrot.exe -2.0 1.0 -1.0 1.0 10000 10000 -1 1.0 0.0 1.0 10000 10000
```