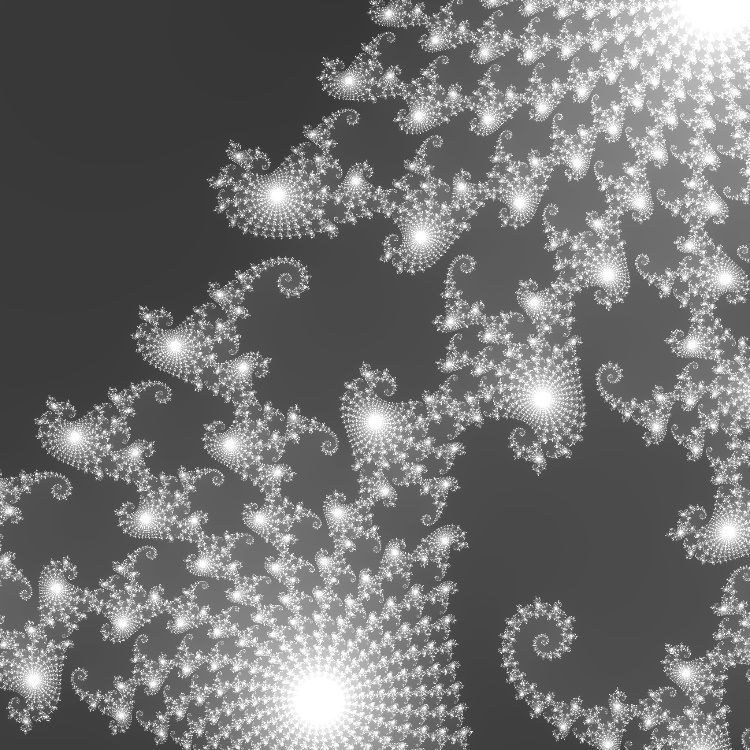
Zixiu Su

Professor Trevor Bakker

CSE 3320-002

10/6/2019

Threads Assignment Report

 This assignment is to implement a multi-thread program and observe the change of the run time with respect to the number of threads. The testing model is generating fractal image (Mandelbrot set) with Escape Time Algorithm. The single thread version source code for that task is given. It iterates through each pixel’s coordinates in the complex plane, recursively mapping it by the relation Pc: z -> z^2 + c until it’s out of bound or the number of steps reaches an arbitrary maximum value. Then we map the number of steps to a grey scale to generate an image. A test image with command “./mandel -x -0.7468 -y 0.117 -H 2048 -W 2048 -s 0.00001 -m 400 -o m1.bmp” looks like this:

To parallelize this task, we can slice the image into multiple bands and assign each of them to a single thread. For the library, I choose <pthread.h> in C because it’s covered in class and the only one I know. I changed the compute\_image function to take an extra parameter int t to distinguish the working section by the outer loop (j value). And added a timer before and after the image computation, the result is saved in “log.csv”.

I use “gcc -std=c99 -Wall t\_mandel.c bitmap.c -o t\_mandel -lpthread” to compile on Omega. And testing these two settings with different thread number (-n 1, 2, 3, 4, 5, 10, 50, repeat 10 times each, find the fastest):

•A: -x -.5 -y .5 -s 1 -m 2000 -W 2048 -H 2048

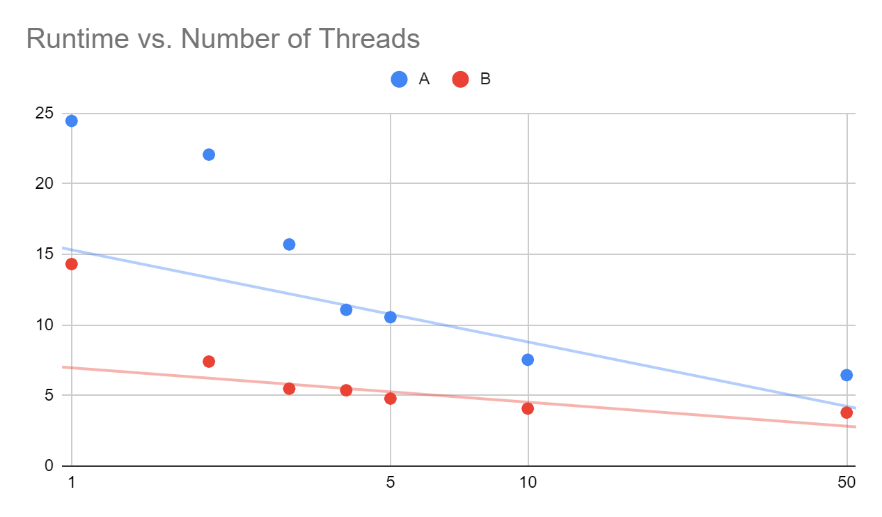
•B: -x 0.2869325 -y 0.0142905 -s .000001 -W 2048 -H 2048 -m 1000

Presumably, the runtime will decrease as the thread count increases until a threshold (context switching cancels the benefit of parallelism).

Results:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 10 | 50 |
| A | 24.4571 | 22.0678 | 15.7053 | 11.0726 | 10.5497 | 7.5326 | 6.4435 |
| B | 14.3148 | 7.4048 | 5.4857 | 5.3705 | 4.7831 | 4.0747 | 3.7871 |

Plotting:



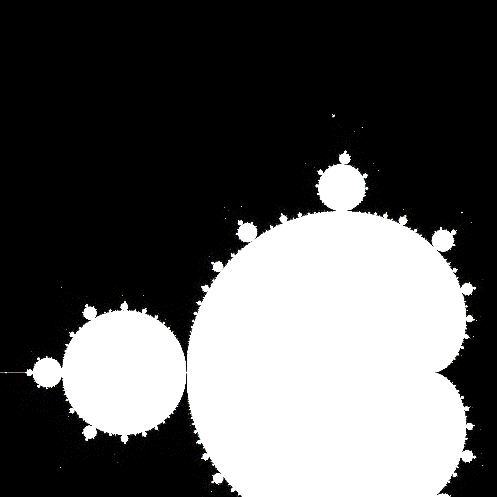
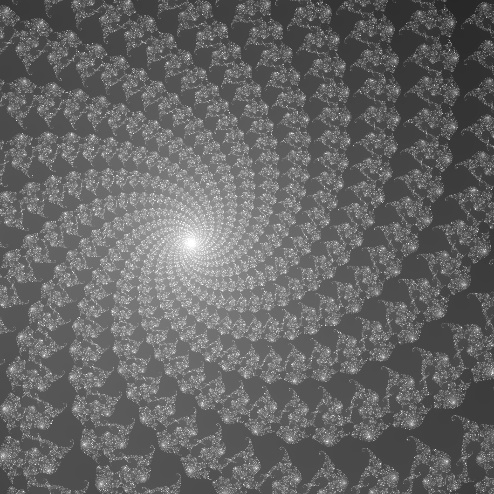
The trend is generally as predicted. With an anomaly at A(n=2), it can be explained by the characteristic of A’s image. The dark area means the recursion ends early due to out of bound, while the bottom section is more computation-heavy.

Figure 2. B

Figure . A