# Assignment 2

Tuesday, March 22, 2022 1:57 AM

• In your own words, briefly explain the purpose of the experiments and the experimental setup. Be sure to clearly state what your command line arguments were, so that your results can be reproduced to review your report.

This experiment essentially shows the advantages and disadvantages of multithreading and how threads work. In this case, when executing compute_image with only one thread, that one thread would have to go through the whole image pixel by pixel, which is very inefficient and time-consuming, hence the amount of time it takes to render the bmp file. But when multithreading was added, as more and more threads are added in through the use of -n, in the command line, the amount of time it takes to render becomes less and less. Of course, we can kind of see that as we add more and more threads, the difference in time becomes less and less. Moreover, if images are detailed enough, or on the flipside, not as detailed, a greater amount of threads may become detrimental to the performance of the program.

The configuration used is written like so (can also be found in configuration.txt):
./mandel -x 0.286932 -y 0.014287 -s .00001 -m 1500 -H 3080 -W 2000 -o mandel5.bmp

• For the following two configurations, measure and graph the execution time of multithreaded mandel using 1, 2, 3, 4, 5, 10, and 50 threads. The execution time of these experiments may be upset by other things going on in the machine. So, repeat each measurement ten times, and use the fastest time achieved.
• A: mandel -x -.5 -y .5 -s 1 -m 2000
n = {
    1 = 1109482 microseconds or 1109 milliseconds
    2 = 1003786 microseconds or 1004 milliseconds
    3 = 709968 microseconds or 710 milliseconds
    4 = 503683 microseconds or 504 milliseconds
    5 = 483600 microseconds or 484 milliseconds
    10 = 314129 microseconds or 314 milliseconds
    50 = 295307 microseconds or 295 milliseconds
}
• B: mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000
n = {
    1= 2725526 microseconds or 2726 milliseconds
    2= 1417731 microseconds or 1418 milliseconds
    3= 1055895 microseconds or 1056 milliseconds
    4= 798226 microseconds or 798 milliseconds
    5= 939726 microseconds or 940 milliseconds
    10= 939495 microseconds or 939 milliseconds
    50= 857868 microseconds or 858 milliseconds
}
• Explain the shape of the two curves. What is the optimal number of threads? Why do curves A and B have a different shape?
Shape A had a decreasing curve the whole way from n = 1 to n = 50, although this is not the case for Shape B. In fact, shape B even rises in time taken to render from n = 4 and over. It seems that the optimal number for Shape B is 4 threads, whereas for Shape A the number of threads that resulted in the best runtime is 50 threads. The difference in shapes may be because of the difference in their image. With this in mind, it seems that Shape B is much more taxing because of how much detail it puts the program through: referring to the scale value as well as its W and H values which make it that much more detailed. As more threads are used, the more expensive the context switches become as the program would have to go through context switches quite a lot.