Mandelbrot Report: Parallelization Comparision

Gardner, Mariah

The University of Texas at Arlington

CSE3320: Operating Systems

Purpose of the Experiment

The purpose of this experiment is to evaluate the performance of a parallelized Mandelbrot set computation algorithm. Specifically, we measure how the execution time of the program scales with an increasing number of threads, comparing different configurations of the Mandelbrot set computation. By using multithreading, we expect the execution time to decrease as more threads are used, up to a point where performance may plateau or even degrade due to overhead or contention.

Experimental Setup

To conduct the experiments, the Mandelbrot set computation program was modified to support multithreading. The program computes the Mandelbrot set for a given region of the complex plane, and the computation is divided among multiple threads. The -n flag was used to specify the number of threads, and the execution time was measured for each configuration.

For the experiments, the following command line arguments were used:

Experiment A: mandel -x -.5 -y .5 -s 1 -m 2000

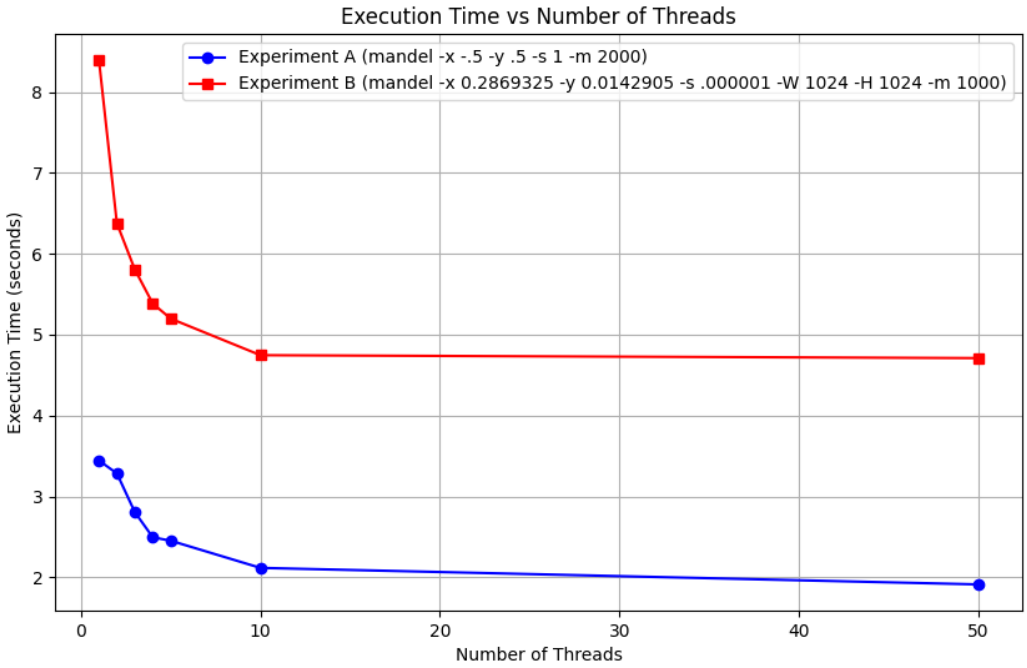
This experiment uses a large region of the Mandelbrot set with a moderate number of iterations (2000). This configuration was tested with 1, 2, 3, 4, 5, 10, and 50 threads.

Experiment B: mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000

This experiment uses a much smaller region with a very high level of detail, and a higher image resolution of 1024x1024 pixels. The number of iterations is set to 1000. This configuration was also tested with 1, 2, 3, 4, 5, 10, and 50 threads.

Results

The execution times for both experiments were measured and graphed. The following observations were made:



Explanation of the Shape of the Curves:

Experiment A: The Mandelbrot set calculation for this experiment involves a relatively large but not very detailed region. With a lower number of iterations (2000), the computation is less complex. As a result, threading provides good parallelism for the first few threads. However, after 10 threads, the execution time flattens out, indicating that the system has reached its optimal level of thread utilization. The curve for **Experiment A** is less defined because of overhead or other system limitations that cause slight

fluctuations in performance. Despite this, there is still a small gain in performance when going from 10 to 50 threads, though it's marginal.

Experiment B: The Mandelbrot set calculation here involves a very small region and high resolution (1024x1024 pixels). The number of iterations is 1000, and the high level of detail benefits greatly from parallelism. The curve for **Experiment B** is much smoother and cleaner, with a steeper slope indicating better gains for each additional thread. The performance continues to improve with each added thread, but after 10 threads, the rate of improvement starts to diminish. Even so, Experiment B has a much higher execution time due to the increased complexity of the task.

Optimal Number of Threads:

Experiment A: The optimal number of threads for **Experiment A** is around 10. Beyond 10 threads, the performance gain becomes marginal. Although there is a slight performance improvement when going from 10 to 50 threads, the gains are relatively small, and the system's ability to efficiently use more threads begins to plateau. The curve flattens after 10 threads, and additional threads contribute less to the overall performance.

Experiment B: For **Experiment B**, the optimal number of threads is also around 10 or slightly higher, as performance continues to improve steadily with each additional thread. However, the gains from adding more threads are more pronounced in **Experiment B** due to the higher complexity of the task. The curve for **Experiment B** is much cleaner and steeper, reflecting the system's ability to scale more effectively as threads increase. Despite this, after around 10 threads, the execution time starts to level off, though it still remains significantly higher than in **Experiment A**.

Resources

Original Code: The Mandelbrot set computation, and multithreading modifications were based on the original code provided in the course repository. You can refer to the original version for a single-threaded implementation and background on the algorithm here.

GitHub Repository: The modified version of the Mandelbrot set computation, with parallelization support using threads, is available on my GitHub. This repository contains the code for both Experiment A and Experiment B, along with a script to plot execution times for varying thread counts.