11/25/2024
Leigh Goetsch
CPE 2600- Systems Programming Dr. Turney

Lab 11: Multiprocessing

Overview

I implemented the movie program and benchmarking program by modifying the `mandel.c` program to have a method called `fly_in()`. This takes the same arguments as `mandel()`, but updates the `center` and `zoom` values to fly into the fractal. The `fly_in()` method is called in a loop in the `main()` method to create a movie of the fractal.

The `mandelmovie` program takes most of the same arguments as `mandel`, but also takes a `num_children` argument (`-n` default to 1) to specify the number of children to fork. The parent process will call `fly_in()`, and the children will call `mandel()` to render the fractal. The parent process will wait for all children to finish before updating the fractal and rendering the next frame.

The `benchmark` program goes through the list [1, 2, 5, 10 20] for the number of children to fork, and runs the `fly_in()` method with the specified number of children. The program records the time it takes to render the fractal for each number of children, and outputs the results to a CSV file, then graphs the results using `gnuplot`.

Results

Here is the table of results from the benchmark program:

num_children	runtime
1	397.798254
2	212.495177
5	101.662532
10	61.160311
20	59.847350

Table 1: Benchmark results

The graph of the results:

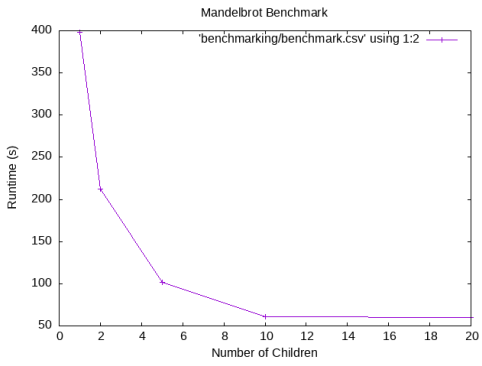

Figure 1: Benchmark results

The graph shows that runtime decreases as the number of children increases, which is expected since children render the fractal in parallel. The runtime drops significantly from 1 to

2 children and then decreases more gradually with additional children.

The graph flattens out after 10 because the CPU has 12 cores.