Sequential Program

Homework 1

By

John Robertson

CS 432 Parallel Computing

February 4, 2021

Problem Specification:

An implementation of [Conway's Game of Life](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life) in C. A cell dies or lives depending on the number of living cells neighboring it. The naïve implementation is O(n3) and the goal is to optimize this as a sequential program (aka, without multithreading).

Program Design:

My program is designed around a struct of int\*\*s, of which only one is created and then passed around. The struct consists of a present and next state to keep track of the states of the game as well as their sizes; as I was writing this, I realized that their sizes were unnecessary and made the boards struct consist of only the arrays. The effect this has on memory and speed, given that the difference made is that of two ints, is marginal. The struct is passed to a triple-nested loop which on the inside calls a function to apply the rules of the game on each element in the cell without changing them, then copying the new board to the first board so the next stage is now in existence. *Because the triple-nested loop is the bottleneck of the program, only it is timed.*

Testing Plan:

My testing plan involved executing the minimum viable version of the program at each turn and altering things until there were no errors, then testing the final version with various inputs to test time taken, as seen below. The printarray function from the example was used to verify the correctness of the game, and srand was given a seed so it would always give the same output.

Test Cases:

|  |  |  |
| --- | --- | --- |
| **Board Size** | **Generations** | **Time Taken (in seconds)** |
| 0 | 0 | 0 |
| -1 | 1 | Throws error |
| 1 | -1 | 0.0000003 |
| 3 | 3 | 0.000002 |
| 1000 | 1000 | 135.274725 |
| 5000 | 1000 | 1381.02425 |
| 5000 | 5000 | Too long to feasibly find out |

Analysis and Conclusions:

I couldn't think of any manual improvements to the algorithm. My algorithm involves a loop with two nested for-loops (which call a function of a series of O(1) instructions) and then a call to a function that consists primarily of two more nested for-loops before returning to the original loop, so an improvement would have to be both to the inner two loops and the transpose function. I used -O3 with compilation on input 1000 1000 to see what would happen and the average time was 34.07 seconds, four thirds of a minute faster than my manual work.

References:

<https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/> [How to make a Makefile]