CSC2002S Parallel and Concurrent Computing Assignment 1 Report

By: Noah Gonsenhauser

My parallelization approach:

My parallelization approach to the assignment was to divide up the searching for the minimum point of a function to different CPU threads. The amount of work a CPU thread would do was tested to find the best sequential cutoff point by running a wide range of x and y minimums and maximums, a number of different amounts of columns and rows, and of course different search densities. All of this raw data was interpreted and the best cutoff point was chosen before running the same set of tests on the Rosenbrock function of terrain, to determine accuracy. This was all done by a function within a separate java testing class for both the serial and parallelized versions on different machines in order to have plenty of data for more conclusive results

The sets used for testing were:

Rows and Columns: 10,100,1000 and 5000

Grid sizes: 10, 100, 1000 and 5000, which had the maximums as positive numbers and minimums as negative numbers.

Search densities: 0.1,0.2,0.3 … 0.9,1

And sequential cutoffs (which in this case, was the number of points allocated to a thread):

10,100,1000,10000,100000,1000000

Every possible combination of the above sets was tested for the parallelized version of the program, it was tested once with the Rosenbrock function with the best cutoff on all 3 machines used for testing, and all machines also were tested with the serial version too.

Validation:

In order to validate my data, I compared my results with that of the Serial Algorithm for all cutoffs on all machines and the general consensus was that it was as accurate as the serial algorithm. The larger the dataset got the less accurate the algorithm became, even with the higher search densities. (10000 rosenbrock issue) (How did I benchmark the program)

Systems tested on:

I tested my algorithm on my own personal desktop computer in an ubuntu VM running x86 architecture, I tested it on my MacBook air which is ARM based, and I also tested it on the university’s nightmare server. There were no issues in testing the algorithm in batches, however when testing different sequential cut-offs, there was an issue where the program would run out of memory on my laptop. (not sure why this is) (Any other difficulties) It was also found that 10,000 was the best sequential cutoff for all machines

Speedup Graphs:

(Grid Size vs Num Searches vs Search Algorithm speed)

Discussion:

Conclusion:

GIT LOG:

LINK RAW DATA HERE: