CSC2002S Assignment PCP1 2023
Parallel Programming with Java:
Parallelizing Monte Carlo Function Optimisation

NKLASA001

Asandile Nkala

Aim:

The aim of this assignment is to implement the Monte Carlo Minimization Parallel which uses Monte Carlo Method to locate the minimum in a function and Search to determine the next point to be visited.

This assignment investigates the performance of sequential vs parallel programs using the Java Fork/Join Framework. This assignment will investigate and conclude as to which conditions is it best to use parallelization over sequential programs vice versa.

Methodoligy:

1. Parallelization:

Multithreading was achieved by implementing a method Divide & Conquer through Java Fork/Join Framework to compute results Parallelly.

The Parallel program reads an Array of Searches and creates subtasks, the array is then divided into subarrays until the the length of the subarrays meets the requirement of THRESHOLD, the the subarrays are computed Sequentially.

The returned arrays from subtasks then get joined together using a method definedwithin the classes to join the results of the subtasks.

2. Timing

The Speedup of this experiment depends on the Row, Column, xmin, xmax, ymin, yman & the Search Density. After executing the program, the programs output the Speedup which is always higher in Sequential Program and lower in Parallel Program. For both the

Sequential and Parallel Programs The time was recorded just before Searching and completing the timing as soon as it they were done Searching.

3. Thresholds:

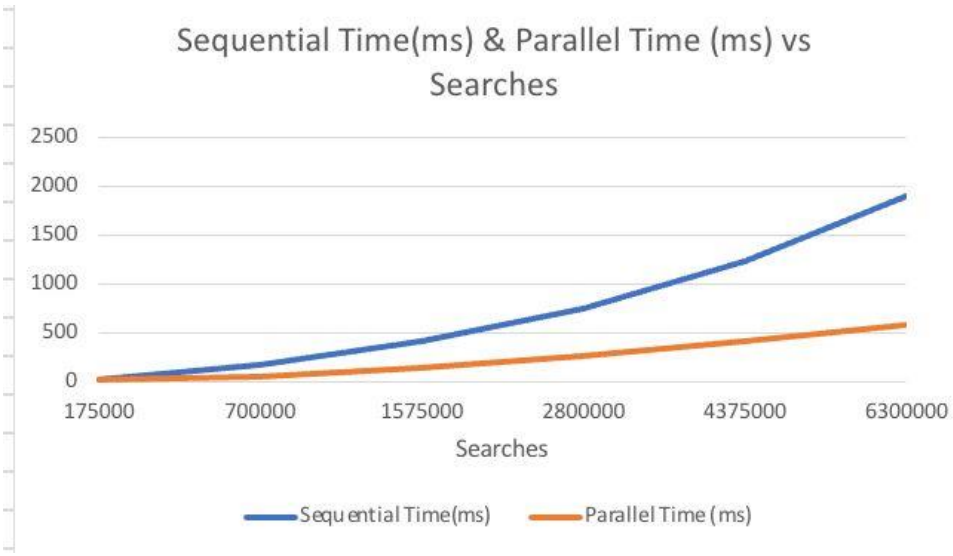
The treshold was initially set to the size of the data set and then recursively dividing the size of the array by 2 until the performance of the program begins to slow down. I used this method since the Java Fork/Join Framework uses the concept of Divide & Conquer to recursively divide a task into smaller sub tasks to process independently then combining the results of all sub tasks into a single results.

4. Testing:

The programs were tested on my PC Intel core 15 machine with 4 cores and the departmental server that runs on a 4 core 8 thread CPU. I also run my program on a computer in CSC Building Senior Lab and The programs were runnin faster in Senior Lab.

Table and Graph:

Searches	Sequential Time(ms)	Parallel Time (ms)
175000	30,75	28,03
700000	174,64	61,75
1575000	417,25	147,5
2800000	753	267,5
4375000	1243,75	428,35
6300000	1908,75	592,25



Discussion:

The results above shows that Parallel programs have better performance compared to Sequential programs but depending on the Threshold of the Parallel program.

It is very hard to accurately time long a parallel program performs especially when we consider that parallel tasks create overhead because of joining the results of subtasks, which in this case was to merge arrays, with the use of optimal thresholds, the overhead is significant hence why I am confident in the timing of the programs

Conclusion:

Given the threshold, parallel programs prove to be superior over sequential programs. To produce the best results from parallel programming, we need to use the threshols as to minimie the computational overheah that is a result of using a threshold value that is either too low or splitting as task that is already small enough to compute sequentially.