OPERATING SYSTEMS II

OpenMP vs Pthread

-Jupally Sriram

The underlying idea in implementing this project is to know how many checkings are required to be made. For a typical 9*9 sudoku, there are 9 rows, 9 columns and 9 grids that require checking.

For an N*N sudoku, there will be N rows, N columns and N grids that require checking. Individual functions have been implemented to check a given column or a given row number or a given grid number. There are a total of 3*N checks to be made. Starting from the first one, the checkings are distributed to threads first as in which thread needs to verify which row or column or grid.

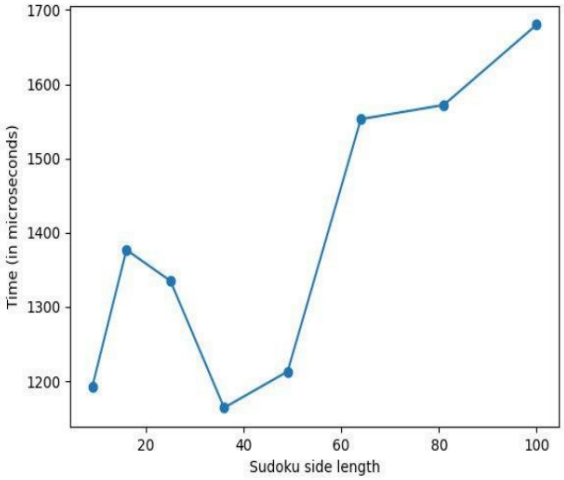
This is done using - There are 3*N verifications in total to be made. They are put into an array and kth thread is made to verify ‘k-1’th element, ‘2*k-1’th element and so on. This makes the work distribution as even as it can get.

In Pthread and OpenMP, this is achieved in respective styles of multithreading, and can be understood.
Following are the graphical comparisons between the OpenMP library and Pthread library.

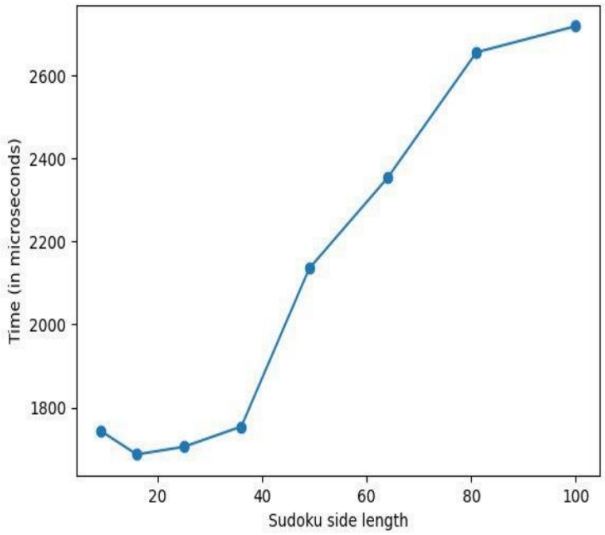
Graphs:

PART A:

Graphs with X axis as the sudoku side length and the time taken to compute validity as Y axis.



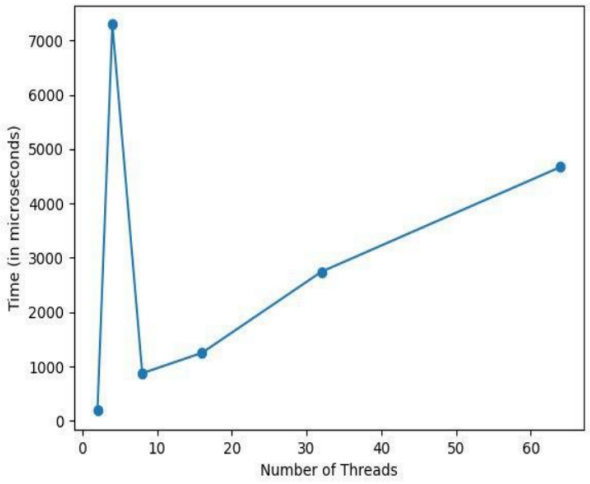
Using OpenMP Library



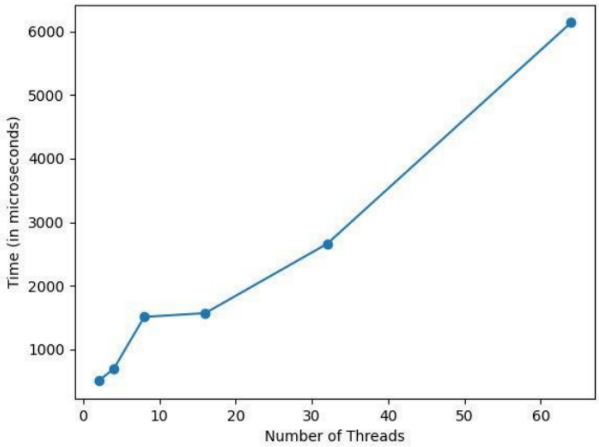
Using PThread Library

PART B:

Graphs with X axis as the number of threads and the time taken to compute validity as Y axis.



Using OpenMP Library



Using PThread Library

Observation and Analysis:

- Overall, on an average OpenMP library performs better than the Pthreads library while computing the validity of a sudoku puzzle.
- In part A, we can see that in the graph using Pthreads, there is a dip and then a gradual increase, which is due to higher threads allocated for less sized sudoku at first, which increases the computation time and wastes resources. As we increase the sudoku's size, we can distribute the load evenly to the threads.
- In part A, in the graph using OpenMP, a similar trend is followed, but there is a maximum at the stage where sudoku's size is 16 and then a dip and a gradual increase follows.
- This might be because the threads used are 16 and the size of sudoku being used is also 16. So, some kind of maximum/minimum might be occurring in the computation.
- While in part B, the graph using Pthreads library shows gradual increase with the number of threads used. This might be because for a small computation, providing more threads might lead to resource wastage and increase in computation time.
- In part B, the graph using OpenMP library has a peculiar peak when the number of threads equals 4, and the rest is a gradual increase just like in the case of Pthreads. This might have to do with the number of cores of the device and when the number of threads equals this number, some kind of maximum/minimum might be occurring in the computation process