Programming Assignment 2: Validating Sudoku Solution
CO21BTCEH11004

Global variables are created:-

- arr :- 1d (n*n) array to store sudoku
- chk :- 1d (3*n) array to store is row, col and grid are valid or not.
- n,k :- to store size of sudoku and no. of threads

Functions defined:-

- chkRow(int x) :- Return 1 if row x is valid else 0.
- chkCol(int x) :- Return 1 if col x is valid else 0.
- chkGrid(int x) :- Return 1 if grid x is valid else 0.

Pthreads:-

main() function ➖

- Input n,k, and arr (sudoku) from the input file.
- K threads are generated and each calls ThreadRunnner(int) func passing thread number as parameter (0-k-1).
    - ThreadRunner(int):- Assign threads to different rows, cols, and grids to check and update the chk matrix.
- K threads are joined using thread join.
- Parent thread checks whether sudoku is valid or not by consolidating the data generated by threads.
- Output file is printed.

Distribution of work ➖

- Each thread is assign a number mod from 0 to k and we have to check n rows,n col and n grid. For (i=0 to n) when i%k == mod thread check that row or col or grid.

OpenMp :-

main() function ➖

- Input n,k, and arr (sudoku) from the input file.
- Parallelizing for loop running for range(0:3*n) for k threads and task is divided to check row, col and grid.

- Parent thread checks whether sudoku is valid or not by consolidating the data generated by threads.
- Output file is printed.

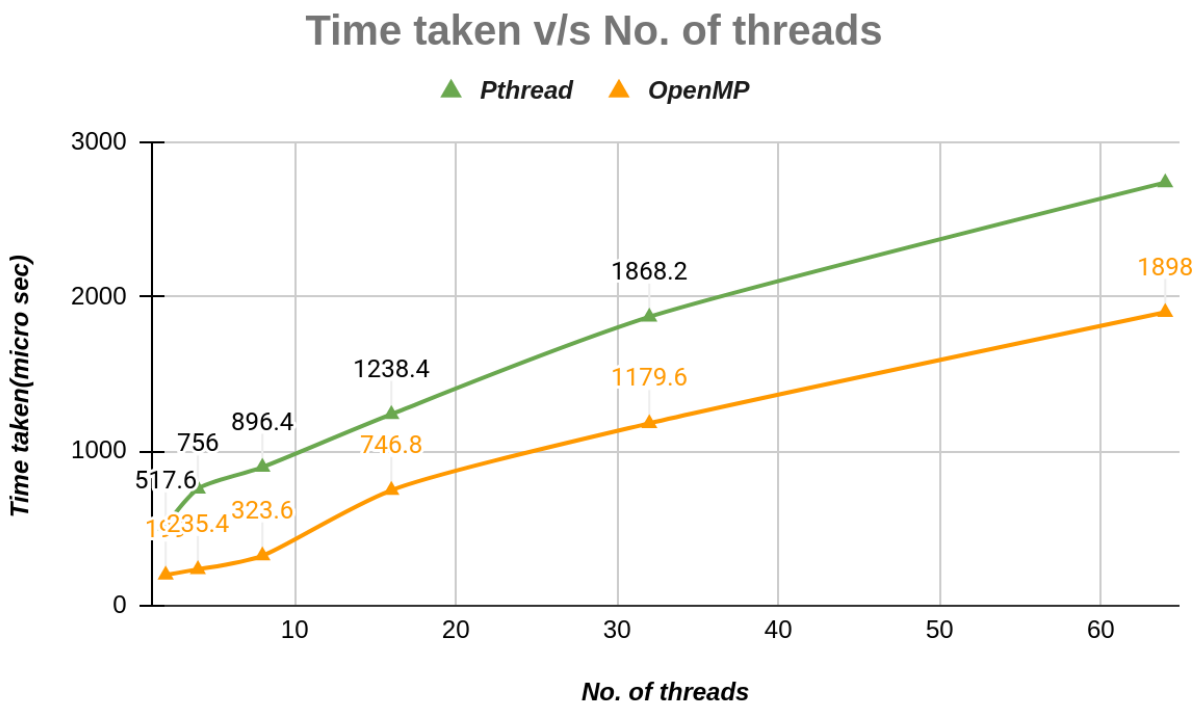
Distribution of work ➖

- Each thread is assign a number mod from 0 to k and we have to check n rows,n col and n grid. For (i=0 to n) when i%k == mod thread check that row or col or grid.

Plots ➖

Time is started just before creation of threads and stopped after checking by parent thread that sudoku is valid or not. Printing time to Output file by main thread is not included.

Variable threads & fixed size of sudoku ➖



Pthread	Time taken					Avg Time
	1	2	3	4	5	
2	505	537	535	492	519	517.6

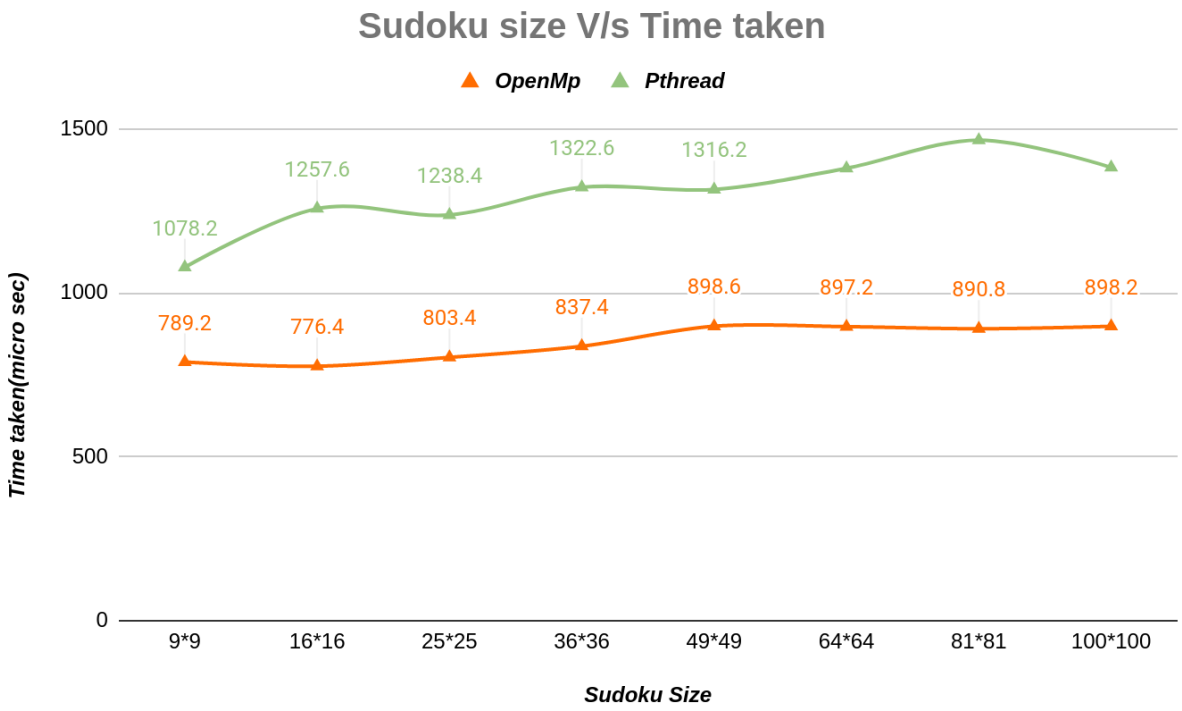
4	758	771	781	764	706	756
8	902	892	877	934	877	896.4
16	1197	1204	1293	1283	1215	1238.4
32	1980	1901	1730	1797	1933	1868.2
64	2736	2621	2645	2788	2896	2737.2

OpenMP	**Time Taken**					**Avg Time**
	1st	**2nd**	**3rd**	**4th**	**5th**	
2	187	202	201	194	211	199
4	219	218	251	224	265	235.4
8	323	336	319	325	315	323.6
16	731	755	695	775	778	746.8
32	1175	1135	1238	1153	1197	1179.6
64	1922	1897	1829	1848	1994	1898

Analysis ➖

- The time is increasing as no. of threads are increased as size of sudoku is 25x25 & algorithm time complexity [for thread created] O(n^2), for this small computation context switching takes more time comparatively. Hence time is increased in both OpenMP and pthreads.

Variable size of sudoku & fixed threads ➖



Pthread Sudoku Size	Time taken					Avg Time
	1st	2nd	3rd	4th	5th	
9*9	1090	1063	1026	1031	1181	1078.2
16*16	1220	1227	1268	1192	1381	1257.6
25*25	1197	1204	1293	1283	1215	1238.4
36*36	1359	1430	1293	1252	1279	1322.6
49*49	1309	1230	1309	1376	1357	1316.2
64*64	1373	1359	1469	1298	1405	1380.8
81*81	1485	1586	1546	1399	1319	1467
100*100	1413	1381	1289	1349	1484	1383.2

OpenMP Sudoku Size	Time taken					Avg Time
	1st	2nd	3rd	4th	5th	
9*9	864	815	732	816	719	789.2
16*16	773	775	744	785	805	776.4
25*25	731	779	859	794	854	803.4
36*36	931	834	814	789	819	837.4

49*49	847	894	812	988	952	898.6
64*64	981	812	839	996	858	897.2
81*81	895	865	838	910	946	890.8
100*100	974	936	834	906	841	898.2

Analysis

- The time taken if size of sudoku is increased keeping thread 16 is increasing very slowly for pthreads while const kind for OpenMP. For this also the sudoku size for O(n^2) algorithm to run n =100. There is very little change in time. For laptop today they execute 10^8 operations per sec. Comparing to this, 100*100 sudoku is so small.