ID1217 CONCURRENT PROGRAMMING

ASSIGNMENT -2

Programming with OpenMP

SUBMITTED BY-
KISHORE BAKTHA

PROBLEM 1

a)

In order to compute the maximum and minimum element along with their positions, I first initialized global variables called max and min along with their respective positions. Then to compute the maximum element, I first check if the element is greater than max and then in the critical section, I double check again before updating the corresponding max and the positions. Similarly, the value of min and its positions is computed. The value of total is computed using the reduction function which stores a local variable for each thread and then finally adds up the result to get the correct value.

Size- 1000*1000

Thread	Execution Time	Speedup
2	0.002535 sec	1.4564
4	0.001837 sec	2.0098
8	0.001786 sec	2.0672

Size- 5000*5000

Thread	Execution Time	Speedup
2	0.051399sec	1.5996
4	0.037057 sec	2.21864
8	0.036804 sec	2.2338

Size- 10000*10000

Thread	Execution Time	Speedup
2	0.226659 sec	1.34972

4	0.146011 sec	2.09523
8	0.143011 sec	2.1392

b) I have developed a variation of the first program to avoid the use of reductions. In this program, I have used parallel pragma primitive along with critical section. I have divided the matrix into strips so that each thread gets a particular strip to execute. I have used three critical sections-one for max,one for min and one for sum. I have used temporary variables to store the results of each thread.

Size- 1000*1000

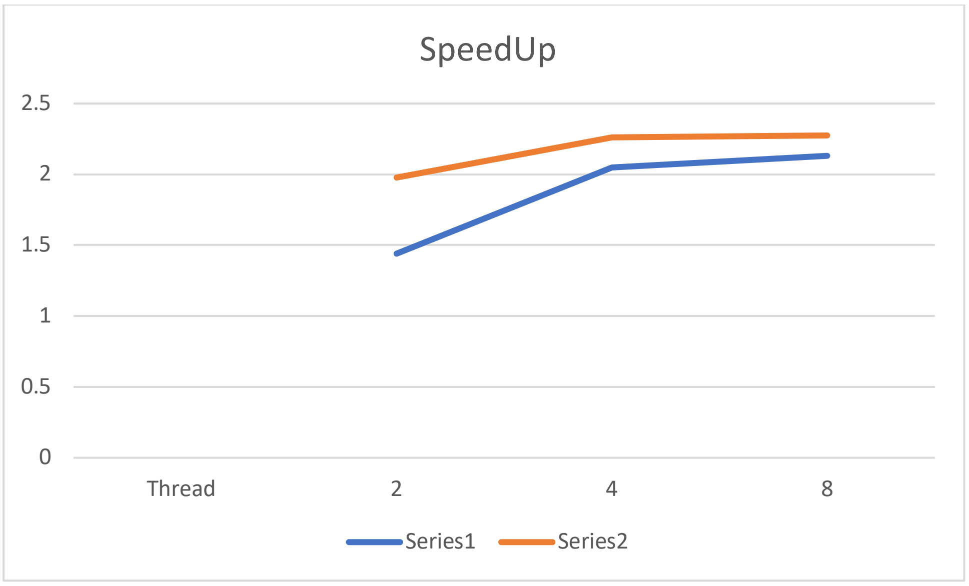
Thread	Execution Time	Speedup
2	0.001867 sec	1.9775
4	0.001634sec	2.2595
8	0.001623 sec	2.2748

Size- 5000*5000

Thread	Execution Time	Speedup
2	0.059071 sec	1.39181
4	0.034975 sec	2.35071
8	0.033779 sec	2.4339

Size- 10000*10000

Thread	Execution Time	Speedup
2	0.212389 sec	1.4404
4	0.149241 sec	2.0499
8	0.143564 sec	2.13094



PROBLEM 2

a) In order to implement quicksort using recursive parallelism, one recursive function is executed by the calling thread and then another recursive function can be executed by creating one more thread with the help of pragma task. First we define the pragma parallel primitive and assign one thread to execute the quicksort function. Then we define one recursive task to be executed by creating a task.

No. of workers-8

Size	Execution Time	Speedup
10000	0.006105 sec	0.3276
100000	0.082022 sec	1.4630
1000000	4.23277 sec	2.74288

No. of workers-4

Size	Execution Time	Speedup
10000	0.006415 sec	0.3118
100000	0.092112 sec	1.30276
1000000	4.83317 sec	2.402

No. of workers-2

Size	Execution Time	Speedup
10000	0.006531 sec	0.30623
100000	0.100831 sec	1.190
1000000	6.09181 sec	1.9058

Result: The programs have been implemented successfully.