Homework 2 Report

Concurrent programming, ID1217

Group members
[Sai Santhoshi Srinithya , darbha@kth.se]

Task 1: Compute Sum, Min, and Max of Matrix Elements

Code implementation

The code parallelizes sum, min and max (ant their positions) values in a matrix. The values of min and max and updates to their values are protected against race conditions using #pragma omp critical, which means only one thread can access at any given point.

Results

The program was executed and tested for matrix of size 1000 ,5000 ,10000 and below is the reported execution time with varying number of threads and performance analysis(the median time was obtained for 5 runs)

Threads	Median Time (s)	Speedup	Efficiency (%)
1	0.00257894	1.00	100.0%
2	0.00134111	1.92	96.0%
3	0.00108525	2.38	79.3%
4	0.000944236	2.73	68.3%

Table 1: Results for matrix of size 1000x1000

Threads	Median Time (s)	Speedup	Efficiency (%)
1	0.0586526	1.00	100.0%
2	0.031563	1.86	93.0%
3	0.0232818	2.52	84.0%
4	0.0157341	3.73	93.2%

Table 2: Results for matrix of size 5000x5000

Threads	Median Time (s)	Speedup	Efficiency (%)
1	0.223168	1.00	100.0%
2	0.117805	1.89	94.5%
3	0.0779314	2.86	95.3%
4	0.0652213	3.42	85.5%

Table 3: Results for matrix of size 10000x10000

Task 2:Quicksort

Code Implementation

The code parallelizes quick-sort for sorting an array. The array is partitioned into smaller sub-arrays using a pivot, and then parallel recursive quick-sort tasks are created using #pragma omp task. The #pragma omp single directive ensures that only one thread initiates the sorting process. To minimize overhead- tasks of sub-arrays executed in parallel.

Results

The programs was executed and tested for arrays of size 1000 ,5000 ,10000 and below is the reported execution time with varying number of threads and performance analysis(the median time was obtained for 5 runs)

Threads	**Median Time (s)**	**Speedup**	**Efficiency (%)**
1	0.000148	1.00	100.0%
2	0.000642	0.23	11.5%
3	0.001653	0.09	3.0%
4	0.001524	0.10	2.5%
5	0.004652	0.03	0.6%

Table 4: Results for array size 1000

Threads	**Median Time (s)**	**Speedup**	**Efficiency (%)**
1	0.001809	1.00	100.0%
2	0.001115	1.62	81.1%
3	0.001671	1.08	36.0%
4	0.002321	0.78	19.5%
5	0.003188	0.57	11.4%

Table 5: Results for array size 5000

Threads	**Median Time (s)**	**Speedup**	**Efficiency (%)**
1	0.001136	1.00	100.0%
2	0.001353	0.84	42.0%
3	0.002004	0.57	19.0%
4	0.003075	0.37	9.3%
5	0.003587	0.32	6.4%

Table 6: Results for array size 10000