CS 4380 / CS 5351
Parallel Programming / Parallel Processing
Fall 2020
Colten Murray

Project 4

Due: *Thursday, October 29 at 3:00 PM*

4.1 Collatz [25u/25g points]

Question 4.1a) What compute times do you get (in seconds)? List them in a table where the first column shows the number of threads used (increasing from top to bottom) and the remaining columns lists the compute time for the different schedules (using the same order as in the submission script). Use two digits after the decimal point for the compute times. Use the first compute time with one thread as the one-thread compute time for all schedules.

Serial Execution time: 18.41 seconds

Default schedule

Threads	Default
12	1.84 s
24	1.16 s
36	0.76 s
48	0.83 s
60	0.88 s

schedule(static, x)

x Threads	1	64	4096
12	1.94 s	1.81 s	1.76 s
24	1.19 s	1.12 s	1.10 s
36	0.86 s	0.75 s	0.75 s
48	0.65 s	0.73 s	0.70 s
60	0.86 s	0.84 s	0.94 s

schedule(dynamic, x)

x Threads	1	64	4096
12	2.80 s	1.76 s	1.75 s
24	2.41 s	1.05 s	1.01 s
36	2.12 s	0.69 s	0.69 s
48	1.93 s	0.62 s	0.61 s
60	1.75 s	0.68 s	0.60 s

schedule(guided, x)

X Threads	1	64	4096
12	1.75 s	1.75 s	1.75 s
24	1.04 s	1.04 s	1.05 s
36	0.69 s	0.68 s	0.69 s
48	0.63 s	0.70 s	0.66 s
60	0.64 s	0.59 s	0.60 s

Question 4.1b) Which schedule yields the worst parallel performance and why?

- Due to the overhead associated with dynamic scheduling, dynamic scheduling at low a chunk size, Schedule(dynamic, 1), yields the worst performance.

Question 4.1c) What is the main reason for why the best dynamic and guided compute times are lower than the best compute times of the remaining schedules?

- Dynamic and guided schedules allow are able to balance work imbalance as opposed to a static schedule.

Question 4.1d) How does the first schedule (the empty "" schedule) in the sub file distribute the loop iterations over the threads?

- The default schedule statically assigns each thread a block of loop iterations.

Question 4.1e) Even though we are using default(none), *maxlen* cannot be specified in either a shared() or private() clause. Why not?

- Maxlen is implied to be shared when included in the reduction clause.

Submit the **`collatz_omp.cpp`** source file on TRACS. As always, the answers to the questions need to be included in the project report.

4.2 Fractal [25u/25g points]

Question 4.2a) What compute times do you get (in seconds)? List them in a table where the first column shows the number of threads used (increasing from top to bottom) and the remaining columns show the compute time for the three code versions (in increasing version order). Use two digits after the decimal point for the compute times.

Threads	For(frames)	For(rows)	For(columns)
1	37.87 s	37.82 s	37.83 s
12	4.88 s	4.38 s	5.90 s
24	2.63 s	2.37 s	3.62 s
36	1.66 s	1.71 s	3.70 s
48	1.36 s	1.48 s	4.12 s
60	1.63 s	2.83 s	6.25 s

Question 4.2b) Which version is the fastest (for large thread counts) and why?

- The first version is the fastest because it is the most parallelized.

Question 4.2c) The loop-carried dependency does not have to be removed for all three versions. Which versions work fine if it is left in the code and why?

- The loop-carried dependency does not need to be removed in the second and third version of fractal. The variable delta is only incremented in the first for loop that is not parallelized in the second and third version .

Question 4.2d) The submission script runs one experiment twice. Even though the executable, the program input, and the hardware are the same, the compute times usually differ. Explain why.

Name the source files **`fractal1_omp.cpp`**, **`fractal2_omp.cpp`**, and **`fractal3_omp.cpp`** and submit them on TRACS.

4.3 Ray Tracer [25u/25g points]

Question 4.3a) What compute times do you get (in seconds)? List them in a table where the first column shows the number of threads used (increasing from top to bottom) and the remaining columns show the compute time for the two problem sizes (in increasing order). Use two digits after the decimal point for the compute times.

Threads	Width = 2000	Width = 4000
1	1.04 s	4.10 s
12	0.15 s	0.43 s
24	0.12 s	0.36 s
36	0.10 s	0.27 s
48	0.13 s	0.38 s
60	0.20 s	0.34 s

Question 4.3b) When increasing the thread count, the runtimes first decrease but eventually increase again for the largest tested thread counts. Explain this behavior.

- As the overhead for splitting the work evenly amongst the threads increases, the benefit of parallelization diminishes.

Question 4.3c) Even though we are using default(none), *ball_x* cannot be specified in either a shared() or private() clause. Why not?

- Ball_x cannot be specified because it is declared and initialized inside the #pragma omp for block

Question 4.3d) Why does the omp.h header file not need to be included?

- The code does not require any functions included in the omp.h

  header file.

Name the source file **`raytrace_omp.cpp`** and submit it on TRACS.

4.4 MIS [25u/25g points]

Question 4.4a) What compute times do you get (in seconds)? Present them in a table for increasing thread counts from top to bottom and use the same order as in the submission script for listing the inputs from left to right. Use three digits after the decimal point.

Threads	cit-Patents	coPapersDBLP	kron_g500-logn21	uk-2002	USA-road-d.USA
1	0.278 s	0.035 s	0.262 s	0.955 s	0.664 s
12	0.306 s	0.051 s	0.268 s	0.914 s	0.670 s
24	0.332 s	0.064 s	0.273 s	0.960 s	0.788 s
36	0.331 s	0.061 s	0.266 s	1.009 s	0.685 s
48	0.347 s	0.073 s	0.291 s	1.024 s	0.782 s
60	0.373 s	0.107 s	0.318 s	1.013 s	0.771 s

Question 4.4b) Compute the geometric mean compute time for each thread count (across the five inputs). Based on this mean, which thread count yields the best performance?

- Thread count = 1 results in the best performance.

Question 4.4c) Why is it not possible to specify a schedule for the outermost pragma?

- A schedule cannot be specified for the outermost pragma because there is no

  work yet to be scheduled amongst the threads.

Question 4.4d) Is the variable *goagain* thread private or shared? Explain.

- The goagain variable is thread private as it pertains to the thread specific node.

Name the source file **`mis_omp.cpp`** and submit it on TRACS.