Noah del Angel

Question 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 second column lists the compute times. Use two digits after the decimal point for the compute times. Highlight the cell showing the lowest compute time.

Threads	Compute Time
1	39.26
4	10.35
8	5.33
12	3.45
16	2.74
20	2.07
24	1.78
28	1.48
32	1.41
36	1.15
40	1.07
44	0.94
48	0.92
52	0.80
56	0.77
60	1.00
64	1.02

Question 1b) What is the highest observed speedup (relative to the pthread code running with one thread)?

The highest speedup, relative to the serial runtime is 56 threads. The speedup between the serial runtime and 56 threads is 50.99.

Question 1c) Explain how the measured runtimes show that hyperthreading is probably not enabled on the compute node.

It appears that hyperthreading is not enabled because performance is decreased when cores are oversubscribed.

Question 1d) Explain why no barrier is needed right before the timed code section.

This is because each thread is forked from the passed function, it doesn't start from the main. As a result, a barrier is unneeded.

Question 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), the second column shows the compute time for the smaller input, and the third column shows the compute time of the larger input. Use two digits after the decimal point for the compute times. Highlight one cell per inputs showing the lowest compute time.

1	7.06	27.77
8	0.99	3.94
16	0.50	2.00
32	0.38	1.51
40	0.25	1.01
48	0.25	1.01
56	0.25	1.01
64	0.22	0.78

Question 2b) Explain the main reason for why the performance is worse with 56 threads than with 64 threads on this 56-core compute node that does not have hyperthreading enabled (this behavior is in contrast to the performance behavior of the collatz code).

This is because the 64 threads is a multiple of the input, whereas 56 is not.

Question 2c) The cyclic partitioning potentially results in false sharing in the pic array. Looking at the code (rather than the compute times), is it likely that significant false sharing will occur for widths that are not powers of two? Explain why or why not.

Significant false sharing will occur for widths that are not powers of two, just like widths that are not powers of two. This is because cyclic distributions tend to result in more false sharing than block, or block cyclic distributions.

Question 2d) Why would the performance be substantially worse if we parallelized the for-col loop instead of the for-frame loop (using the same pthread parallelization approach)?

The performance would be substantially worse, because the entry for each function is still the outer loop, and thus each thread would run their respective cols but would have to wait for the cols to complete.