Noah del Angel

Question 1a) Check the contents of the `collatz_MPI56.sub` submission script. How many processes are being launched? How many compute nodes do these processes run on?

56 processes are launched in the collatz_MPI56.sub. These processes all run on one compute node.

Question 1b) What compute times do you get (in seconds)? Present the times in a table for increasing process counts from left to right and increasing problem sizes from top to bottom. Use three digits after the decimal point for all compute times (even if the last digit is a zero).

	1	28	56
3 2000000	0.291	0.011	0.008
5 20000000	3.394	0.130	0.067
7 200000000	38.699	1.475	0.761

Question 1c) Based on the compute times, calculate the speedups relative to running the MPI code with one process. Present the speedups in the same format as the compute times but show only two digits after the decimal point.

	28	56
3 2000000	26.45	36.37
5 20000000	26.10	50.65
7 200000000	26.23	50.85

Question 1d) Based on the speedups, calculate the efficiencies relative to running the MPI code with one process. Present the efficiencies in the same format as the speedups.

	28	56
3 2000000	0.94	0.64
5 20000000	0.93	0.90
7 200000000	0.93	0.90

Question 2a) Run the code with the submission scripts at /home1/00976/burtsche/Parallel/fractal_MPI*.sub. What compute times do you get (in seconds)? Present the times in a table for increasing process counts from left to right and "increasing" problem sizes from top to bottom (i.e., use the same order as in the submission script). Use three digits after the decimal point for all compute times (even if the last digit is a zero).

	1	32
512 32	3.406	0.130
512 64	6.809	0.258
1024 32	13.575	0.517
1024 64	27.146	1.04

Question 2b) Based on the efficiencies (which you must compute and mention in your answer), does the code scale well to 32 processes for all four inputs?

The efficiencies for 512 width are 0.81 with 32 frames, and 0.82 with 64 frames. However, for 1024 width the efficiencies are 0.82 with 32 frames, and 0.81 with 64 frames. Here we can see the parallelization weakly scales because the efficiency stays nearly the same with increases in input.

Question 2c) How many cores does a "small" compute node in Frontera have? How many sockets does it have? Check the documentation to find out.

The "small" compute node has 56 cores and 2 sockets.

Question 2d) Explain why no barrier is needed before stopping the timer, i.e., why the compute time printed by process 0 is guaranteed to be the compute time of the slowest process.

MPI_Gather is a blocking operation. Thus, the timing result will not be completed on the main thread until it was received all the data.