Project 2

Noah Torres

2.0) Vector Add
2.1) Collatz

a. Check the contents of the **collatz_MPI12.sub** submission script. How many processes are being launched? How many compute nodes do these processes run on?

- There are twelve processes being launched. There is one compute node in which each process runs on.

b. 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).

- **Run times in seconds**

| | | process counts | | |
|---|---|---|---|---|
| | | 1 | 12 | 24 |
| problem | | | | |
| sizes | 500000 | 0.136s | 0.014s | 0.009s |
| | 5000000 | 1.620s | 0.172s | 0.103s |
| | 50000000 | 18.757s | 1.892s | 1.159s |

c. 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.

- **Speedup**

| | | process counts | | |
|---|---|---|---|---|
| | | 1 | 12 | 24 |
| problem | | | | |
| sizes | 500000 | 1.00 | 9.71 | 15.11 |
| | 5000000 | 1.00 | 9.41 | 15.73 |
| | 50000000 | 1.00 | 9.91 | 16.18 |

d. 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.

- **Efficiency**

| | | process counts | | |
|---|---|---|---|---|
| | | 1 | 12 | 24 |
| problem | | | | |
| sizes | 500000 | 1.00 | 0.81 | 0.63 |
| | 5000000 | 1.00 | 0.78 | 0.66 |
| | 50000000 | 1.00 | 0.83 | 0.67 |

e. 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.

- **Because when the threads are done computing the reduce function acts as a barrier and we know that if reduce is executed all the processes have finished computing up to that point.**

2.2) Fractal

a. What compute times do you get (in seconds)? Present them 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.

**- Run time in seconds.**

| | | process counts | |
|---|---|---|---|
| | | 1 | 16 |
| problem sizes | 512, 64 | 3.862s | 0.327s |
| (width, frame) | 512, 128 | 9.490s | 1.067s |
| | 1024, 64 | 15.400s | 1.270s |
| | 1024, 128 | 37.927s | 4.264s |

b. Based on the efficiencies (which you must compute), does the code scale well to 16 processes for all four inputs?

- **No, based on the efficiencies calculated as the number of frames and width increases it appears to be slightly less efficient. Appears more efficient with less frames then the runtimes with larger frames**.

c. How many cores does a "normal" compute node in Lonestar 5 have? How many sockets does it have?

- From the TACC user portal website it says that compute nodes are dual socket, with 12 cores per socket resulting in 24 total cores.