ECE 56300 Spring 2019 Test 1 – take home due Monday, March 4, 11:59PM

1. Write a row distributed matrix multiply and measure its execution time on 1, 2, 4, 8, 16 processes. Note that you can run up to 20 processes on a node. After you have debugged your program, do the runs on multiple nodes, with up to four processes per node. Repeat the timings increasing the work done in the matrix multiply as needed by the isoefficiency analysis in the slides.

2. Use the communication explained in Cannon's algorithm to implement your matrix multiply. measure its execution time on 1, 2, 4, 8, 16 processes. After you have debugged your program, do the runs on multiple nodes, with up to four processes per node. Repeat the timings increasing the work done in the matrix multiply as needed by the isoefficiency analysis in the slides.

3. Run your program in 2 using the MKL math kernels to do the matrix multiply (see https://www.rcac.purdue.edu/knowledge/scholar/compile/intel_mkl) Run it on 1, 2, 4, 8, 16 processes with the same problem size and compare the times with the runs for 2. Feel free to post and respond to questions on Piazza about how to use the MKL math kernels.

What to turn in:

A. A directory whose name is your userid. If you are an online student, prefix the username with poe_. If you are an in-class student, prefix the username with wng_. The directory will contain X sub-directories.

B. In a subdirectory Code, put your code in a directory (all three programs) with instructions on how to execute it and the .sub files to run them.

C. In a directory called Graphs, X graphs.
i. A graph for with an X axis that gives the number of processes running your program and a Y axis that is the execution time. You should have lines for constant problem size runs for (1) above, the constant size runs for (2) above, and the run for 3.
ii. A graph with the X axis showing the number of processors, the Y axis showing the times, as with (I), for the scaled problem sizes of (1) and (2) above.
Iii. A graph with the X axis showing the number of processors, the Y axis showing the speedup, for the constant problem sizes.
iv. A graph with the X axis showing the number of processors and the Y axis showing the Efficiency for the constant problem size runs of (1) and (2)
v. A graph with the X axis showing the number of processors and the Y axis showing the Karp-Flatt e for the constant problem size runs of (1) and (2).

Notes:

Do your own work. Don't copy code from online. This will be considered cheating.

How big should your problem size be? Big enough that it takes between 20 and 50 seconds to solve sequentially.

When you submit your jobs, try to group sets of runs together in the same qsub submission. You will spend less time waiting.

When debugging your program, use small problems to keep from tying the machines up too much.

Ask questions on Piazza, and good luck!