EECS 587 Homework: Due at the start of class 15 October 2015

This program is to written on your own, not in collaboration. However, feel free to talk with others about MPI, problems with using flux, etc.

Using the Advanced Research Computing flux computer, run an efficient MPI program to use p cores to do a template calculation on an $n \times n$ double-precision real-valued matrix A[0:n-1, 0:n-1]. Do this for n = 1000 and 2000 and for p = 1, 4, 16, and 36. The same program must be used for all of the runs, and should work no matter what the entries of A are and no matter what $p, n \geq 1$ are, for p a perfect square $\leq n^2/9$.

You have to use flux for your timing runs but you can debug on any machines you wish.

Warning: do not put this off until the last minute, and debug using small matrices.

Procedure: For each n and p pair,

1. Initialize A using the definition below.
2. Use MPI_BARRIER, then have the root process (process 0) call MPI_WTIME.
3. Do 10 iterations, defined below.
4. Compute the verification values (see below) and send them to the root process.
5. Have the root process call MPI_WTIME.
6. Print out the elapsed time and the verification.
7. There can be significant timing differences between runs. Therefore, for $n = 1000$ and $p = 4$ do 3 different runs (*not* three timings on the same run, but rather, 3 different batch submissions) and report all 3. For all other combinations of n and p do just 1 run.

Turn in the program code, the timing and verification outputs, and the report.

To generate A: for all $0 \leq i, j \leq n-1$,

$$A(i,j) = \begin{cases} i \cdot \sin(\sqrt{i}) & \text{if } i = j \\ (i+j)^{1.1} & \text{otherwise} \end{cases}$$

Each entry of A can start on whatever processor you choose, but no entry of A can start on more than one processor.

Iterative step: In each iteration, for all $0 \leq i, j \leq n-1$, the new value of $A(i,j)$ is given by:

- $A(i,j) = A_o(i,j)$ if $i = j$ or $j = n-1$

- Otherwise, let

$$x = \sum_{k=1}^{10} |0.5 + A_o(i', j)|^{1/k} - |A_o(i'', j)|^{1/(k+1)} \cdot |A_o(i, j')|^{1/(k+2)}$$

where $i' = (i+1) \bmod n$, $i'' = (i-1) \bmod n$, and $j' = j+1$.
Then $A(i,j) = \max\{-10, \min\{10, x\}\}$

A_o denotes the value from the previous iteration. Be careful that you don't use values from the current iteration. Note: some systems return the value of -1 for $-1 \bmod n$. I want you to use the value $n-1$.

Because the indices use $\bmod$, you can think of the template as treating the array as if it were on a cylinder, rather than a standard square.

To verify a run: Print the values of

$$\sum_{0 \le i,j \le n-1} |A(i,j)| \quad \text{and} \quad \min_{0 \le i,j \le n-1} A(i,j)$$

The report: Turn in a short report which briefly describes how you decomposed the matrix, how you did communication, and the timing result. Analyze whether the timing represents perfect speedup and scaling, and, if not, why is the program not achieving it (or, if you have superlinear scaling, why that occurred). Note that you have scaling in terms of the size of the matrix, as well as in the number of processors. The report needs to be typewritten, though you can do drawings by hand. You will be graded on correctness, the quality of your report, and achieved performance. Since performance is important, you should make sure you do serial and parallel optimizations discussed in class.

Depending on how fast the programs are running and the turnaround time on flux, I may adjust the matrix sizes, number of iterations, or details of the nonsense equations.