EECS 587 Homework: Due at start of class 9 October 2014

Using the CAC flux computer, run an efficient general purpose program to use p cores to solve a PDE on an $n \times n$ double-precision real-valued matrix $A(0\!:\!n{-}1,\ 0\!:\!n{-}1)$. Do this for n = 1000 and 5000 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 $n > 2$ and $p > 0$ are, for p a perfect square $\leq 9 \cdot n^2$. If you are using row or column partitioning, instead of partitioning into squares, then it should work for arbitrary $p \leq 9 \cdot n^2$, not necessarily a perfect square. The factor of 9 is included to insure that there are no pathological cases.

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 call MPI_WTIME.
3. Do 500 iterations, defined below.
4. Compute the verification sum (see below) and send it to the root process.
5. Use MPI_BARRIER, then have the root process call MPI_WTIME.
6. Report the elapsed time and the verification.
7. There are significant timing differences between runs. Therefore, for each n and p pair, do 3 different runs (*not* three timings on the same run) and report all 3.

Turn in a program listing, the timing and verification outputs, and a report.

To generate A: the initial values are defined as follows:

- $A(0, j) = 0.0$ for $0 \leq j \leq n-1$
- $A(n{-}1, j) = 5\sin(\pi(j/n)^2)$ for $0 \leq j \leq n-1$
- All entries except the above are 0.5

Each entry of A can start on whatever processor you choose. If you want multiple copies of an entry, you must copy the value from its initial processor, and this must occur after the barrier.

Iterative step: In each iteration,

- $A(0, j)$ and $A(n{-}1, j)$ are unchanged for $0 \leq j \leq n-1$

- for all other entries, the new value of $A(i,j)$ is the average of $A_o(i,j)$ and the A_o values of the 8 neighbors of (i,j), where A_o denotes the value from the previous iteration. Be careful that you don't use values from the current iteration. A neighbor of (i,j) is any location $(i',j') \neq (i,j)$ where $i' = i$ or $i' = i \pm 1$ and $j' = j$ or $j' = j \pm 1 \bmod n$. Using the mod function eliminates the boundary condition in the second dimension, turning the matrix into the surface of a cylinder.

To verify a run: Calculate $\sum_{0 \leq i \leq n-1} A(i,i)$.

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. Please keep the report short.