EECS 587 Homework: Due by the start of class October 1, 2024

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

Using the Great Lakes computer, run an efficient MPI program to use p cores to do a template calculation on an $m \times n$ double precision real-valued matrix A[0:m-1, 0:n-1]. Do this for ($m = 2000$, $n = 500$) and ($m = 1000$, $n = 4000$) for p = 1, 4, 16, and 36. All calculations should be done in double precision. 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 $p, n, m \geq 1$ are, for p a perfect square such that $p \leq \min\{n, m\}/10$ (this latter condition is just to insure that you won't have any of the small matrix problems discussed in the handout, and you don't have to have the program check that this condition holds).

You have to use Great Lakes for your timing runs but you can debug on any machines you wish. When you submit your program to Great Lakes you must put **a time limit of 5 minutes or less**, though we might adjust this limit if necessary.

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

Procedure: For each combination of matrix size and number of processors:

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 (the WTIME of step 5 - WTIME of step 2) and the verification.

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

To initialize A: for all $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$ $A(i,j) = j \cdot \sin(i) + i \cdot \cos(j) + \sqrt{i+j+1}$
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, let A_o denote the previous value of A. Then for all $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$, the new value of $A(i,j)$ is given by:

- $A(i,j) = A_o(i,j)$ if $i = 0$ or $i = m-1$ or $j = 0$ or $j = n-1$ (i.e., it is unchanged along the border of the matrix)

- Otherwise, let

$$z = \frac{f(A_o(i-1,j-1)) + f(A_o(i-1,j+1)) + f(A_o(i+1,j-1)) + f(A_o(i+1,j+1)) + f(A_o(i,j))}{5}$$

and then $A(i,j) = \max\{-25, \min\{30, z\}\}$

Be careful that you use A_o, not values from the current iteration. Note that the initial values of A might not be in the range [-25,30], but for interior points, all subsequent iterations will be be.

The function f must be written as the Fortran or C/C++ equivalent of

```
function f(x)
double precision x, y
integer i

y=x*2.0
for i=1,10
   y= y+ x*cos(y+i)/1.5^i
end for
return y

end f
```

You cannot do any optimizations to speed up the evaluation of f. For example, it cannot be inlined. The cos function uses the standard mathematical convention of being in radians, not degrees, and `1.5^i` means 1.5^i.

To verify a run: Print the sum of all of the entries of A after the final iteration, and the sum of the squares of the entries.

The report: Turn in a short report which includes the verification and a brief description of how you decomposed the matrix, how you did communication, and your 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. For example, you should turn on the compiler's optimization options.

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

©Copyright Quentin F. Stout