Homework #2

Due April 1, 2011

Problem 1 – Laplacian in parallel

Consider the code you developed in HW#1 in which Laplacian smoothing was iteratively performed in parallel. You should have four different versions of this code, based on the following MPI calls:

i) Blocking send and recv: MPI_SEND, MPI_RECV

ii) Send-recv: MPI_SENDRECV

iii) Buffered send: MPI_BSEND, MPI_RECV

iv) Non-blocking send and recv: MPI_ISEND, MPI_IRECV

You will be assessing the parallel performance of these different versions on Janus. You should already have your account (if not, please contact me right away!), and you can find all information about using Janus on `http://ncar.janus.rc.colorado.edu/`.

For each of the four versions, modify your code using MPI_WTIME() in order to output run time. You should be careful not to include any I/O (e.g. the writing of the final solution) in this timing. Then, for each version of the code, generate two different plots:

1. Speed-up (or strong scaling) as a function of the number of cores. Consider different domain decomposition strategies. Using the fixed problem size of $n1 = n2 = 256$, increase the number of compute cores from 1 to 128 by decomposing in only one direction (say $p1$ increases while $p2$ remains constant at 1), then try increasing the number of compute cores from 1 to 100 by decomposing in both directions simultaneously ($p1 = p2$ increases).

2. Scale-up (or weak scaling) as a function of the number of processors. Using a reference problem size of $n1 = n2 = 64$ on 1 compute core, increase both the number of compute cores and the problem size in proportion, such that $(n1 \times n2)/(p1 \times p2)$ =constant. Again, explore different decomposition strategies, such as decomposing in only one direction (say $p1$ increases while $p2$ remains constant at 1), or decomposing in both directions simultaneously ($p1 = p2$ increases).

Comment on the results you obtain, in particular any difference between the different domain decomposition strategies, and the different communication strategies.

Problem 2 – Parallel conjugate gradient

Using your favorite language, implement a conjugate gradient algorithm that solves the two-dimensional Poisson equation on a periodic domain. We will consider the same 5-pt stencil Laplacian operator that we have been discussing in class (second order accurate discretization of $\partial^2 f/\partial x^2 \approx (f_{i+1} - 2f_i + f_{i-1})/\Delta x^2$).

The version of conjugate gradient you'll implement is a variant of the version discussed in class, where operations are re-ordered such that collective call happen simultaneously, see Fig. 1.

$r^0 = b - Ax^0$;
$q^{-1} = p^{-1} = 0; \beta_{-1} = 0$;
$s^0 = Ar^0$;
$\rho_0 = (r^0)^T r^0; \mu_0 = (s^0)^T r^0; \alpha_0 = \rho_0/\mu_0$;
for $k = 0, 1, \ldots$
  $p^k = r^k + \beta_{k-1} p^{k-1}$;
  $q^k = s^k + \beta_{k-1} q^{k-1}$;
  $x^{k+1} = x^k + \alpha_k p^k$;
  $r^{k+1} = r^k - \alpha_k q^k$;
  check convergence; continue if necessary
  $s^{k+1} = Ar^{k+1}$;
  $\rho_{k+1} = (r^{k+1})^T r^{k+1}$;
  $\mu_{k+1} = (s^{k+1})^T r^{k+1}$;
  $\beta_k = \rho_{k+1}/\rho_k$;
  $\alpha_{k+1} = \rho_{k+1}/(\mu_{k+1} - \rho_{k+1}\beta_k/\alpha_k)$;
end

Figure 1: Serial pseudo-code for conjugate gradient algorithm.

Consider a $[0,1]^2$ domain discretized on a 100×100 mesh, and the following equation:

$$\Delta f(x,y) = +\delta(x - 0.25, y - 0.25) - \delta(x - 0.75, y - 0.75), \tag{1}$$

where $\delta(0,0) = 1$, while $\delta(x,y) = 0$ elsewhere. Start from a zero initial guess, and run the CG algorithm until convergence. Provide the convergence history (residual magnitude as a function of number of iterations), as well as a 2D plot of the converged solution. Then explore the speed-up (strong scaling) properties of your algorithm on Janus using the same approach as in Problem #1 (both with 1-D and 2-D domain decomposition), and comment on the results.