HPC Bootcamp

MPI Part 1

Fortran: I uploaded my random.f90 module.

Project 1.

Computing Pi

A very inefficient way to compute pi is to use a Monte Carlo method. In this we randomly throw "darts" at a circle of radius 1 within a square with sides of length 2. The ratio of the areas is thus pi/4. If we compute the ratio of "hits" within the circle to the total number of throws, then our estimate of pi is 4*ratio.

Write a program that computes pi by this method. Start with a serial code and test it for a relatively small number of "dart throws." You will find it very inaccurate for a small number of throws but should be able to obtain an estimate at least close to 3.

Once you are satisfied that your program works in serial, parallelize it. Distribute the throws over N processes and use a reduction to get the totals for your overall estimate of pi.

Test your parallel code with 8 processes and 1e9 throws.

Project 2

We are going to find the maximum of a 3-d surface by "brute force" evaluation of x, y, z values. This is a method of optimization that has become increasingly popular since it is easily parallelizable. (Older methods some of you may have been taught in a numerical analysis class are often difficult to parallelize and even if they are parallelized, they may actually be slower than the "brute force" method.) The surface is defined by the following rules:

$$\mu_1 = \sqrt{2}$$
$$\mu_2 = \sqrt{\pi}$$
$$\sigma_1 = 3.1$$
$$\sigma_2 = 1.4$$
$$z_1 = 0.1 \sin(x) \sin(xy)$$
$$a = \frac{(x - \mu_1)^2}{2\sigma_1^2}$$
$$b = \frac{(y - \mu_2)^2}{2\sigma_2^2}$$
$$z_2 = \frac{e^{-(a+b)}}{\sigma_1 \sigma_2 \sqrt{2\pi}}$$
$$z = z_1 + z_2$$

The surface is defined over the ranges
$-10\pi \leq x \leq 10\pi$
And
$-10\pi \leq y \leq 10\pi$

Generate a list of N random values for each of x and y over the above range. For testing you can use N=800000. Be sure to measure the time.

Use MPI to solve this problem. Divide up the grid and use a reduction to find the maximum.

Project 3

Modify your program from Project 2 to use gather to obtain not just the maximum from each process, but the corresponding x and y values. Remember that gather collects items in strict rank order. Use either one or more built-ins for your language (Fortran, Python) or loops (C/C++) to find the x, y, and z for the maximum.