Assignment 01: Monte Carlo Simulation to determine the number PI

Before you start the assignment, please study examples in the MPI_notes.pdf, understand basic usage of MPI.

Introduction:

In this assignment you will be tasked with determining the value of the number PI through simulation. Determining an accurate value of PI is difficult to do through normal methods because PI is an irrational number i.e. there is no single fraction that can express the value of PI. Thus to determine the value of PI we must use other methods. In this case we will use Monte Carlo Simulation where we generate the value of PI by using random sampling.

In order to simulate this we will need to remember our formula for the area of a circle, which in case you have forgotten is PI * radius * radius. If we take a circle that has a radius of 1 we get an area of PI * 1 * 1 which simplifies to PI. We will then enclose this in a square that is 2 units wide and 2 units high, such that the circle touches the square on all four sides. The coordinates of the bottom left of the square is (-1, -1) and the top right of the square is (1,1) with the circle centered at (0,0). Random points should be generated between [-1,1] in both x and y. If the random point is less than or equal to a distance of 1 from (0,0) then it will be counted as being in the circle.

We then count up the number of samples that we took (call this n) and also the number of samples that were in the circle (call this x). By dividing x by n we get the percentage of samples in the circle (call this p). If we consider that the ratio of the area of the circle to the area of the square is PI/4 then to get the full value of PI we must multiply p by 4.

This is an example of an embarrassingly parallel problem that is well suited to a distributed system. You should see near linear performance increase in this application when you add more nodes.

Notes:

You have two weeks to do this assignment. Thus the deadline for this assignment will be 2016-03-06 at 23:55. Standard penalties will be applied to work that is submitted so much as a second late. The time of submission as displayed by the moodle will be the reference point for lateness.

You must submit a single zip file (naming does not matter) that contains two files: 1 c++ file containing your MPI source code, and 1 PDF file containing the report you write. Anything other than a PDF will not be accepted. (there are tools for converting DOC to PDF available e.g. the foxit reader plugin, those of you using libre office or LaTeX have direct export to PDF)

Code that fails to compile will incur a penalty of 30%. The accepted compression formats for your archives are tar.gz/tar.bz2/tar.xz/zip/rar/7z any format outside of this will incur a 10% penalty.

For the purposes of this assignment you will only need three standard header files <iostream>, <cstdlib> and <mpi.h> for the timing task you may need an extra header or two.

Task List:

01) write a main method that will initialise MPI, figure out the world rank and world size. Rank 0 should be the coordinator while all other ranks should be participants. Then finalise MPI and return a status of 0 to the OS (10%)

02) write a coordinator method that takes in a single variable (the world size). It should run a monte carlo simulation for a set number of samples. Then it should collect the number of hits from all participant notes (you may only use MPI_Recv() here) and calculate the overall hit ratio for the collection of samples. Display the result of PI on the console (25%)

03) write a participant method that does a similar simulation as 02 above but will communicate its total number of hits to the master (you may only use MPI_Send() here) when it is finished (15%)

04) modify your code in such a way that the total number of samples each node produces can be changed by simply changing the value of a global constant (5%)

05) modify your code in such a way that regardless of how many nodes are used (2, 4, 8, 16, etc) your code needs no modification to function correctly (5%)

05) write a report that analyses two things. First show the progress of your algorithm calculating the value of PI in increments of 4 million samples (use 4 nodes and increment your total samples by 1 million each time) and find where your algorithm returns a value of 3.14159. (20%)

06) compare the speed of your algorithm when using a single node, 2 nodes, and 4 nodes in increments of 4 million samples up to 40 million samples and display these results on a graph. What kind of speed up do you get and explain why. (20%)