Homework 7: Parallel and Distributed Computing

DUE: See exact time on blackboard

ICSI 520

Problem

In this homework you will implement a serial and a distributed implementation to calculate the value of PI as described in the following sections

Serial and Distributed Implementations

You will do both serial and Distributed implementations (can use previous serial implementation)

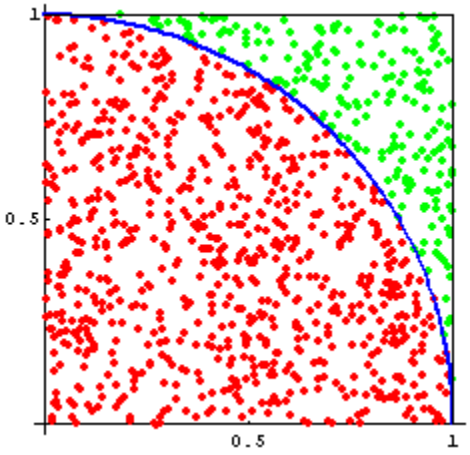
Use timing code to measure timings for this code.

Description

Monte Carlo simulation

Monte Carlo methods can be thought of as statistical simulation methods that utilize a sequences of random numbers to perform the simulation. In a typical process one computes the number of points in a set A that lies inside box R. The ratio of the number of points that fall inside A to the total number of points tried is equal to the ratio of the two areas. The accuracy of the ratio *p* depends on the number of points used, with more points leading to a more accurate value.

Monte Carlo simulation of PI



- Consider a circle inscribed in a unit square. Given that the circle and the square have a ratio of areas that is $p = \pi/4$, the value of π can be approximated using a Monte Carlo method
    - Draw a square on the ground, then inscribe a circle within it.
    - Uniformly scatter some objects of uniform size (grains of rice or sand) over the square.
    - Count the number of objects inside the circle and the total number of objects.

- The ratio of the two counts is an estimate of the ratio of the two areas, which is π/4 or *p*. Multiply the result by 4 to estimate π.

A simple Monte Carlo simulation to approximate the value of π could involve randomly selecting points $\{(x_i, y_i)\}_{i=1}^{n}$ in the unit square and determining the ratio $\rho = \frac{m}{n}$, where m is number of points that satisfy $x_i^2 + y_i^2 \leq 1$.

m = number of points inside the circle

n = total points

In a typical simulation of sample size $n = 1000$ if there were 787 points satisfying $x_i^2 + y_i^2 \leq 1$

$$\rho = \frac{m}{n} = \frac{787}{1000} = 0.787 \quad \text{and} \quad \pi \approx \rho * 4 = 0.787 * 4 = 3.148$$

Results

n	**Number of cores (p)**	**Serial execution time**	**Distributed Execution time**	**Speedup = serial time/ dist time**
700	<decide the best number – max out the cores that you can use on the cluster>			
1400				
2500				
5000				
10000				
20000				
30000				
50000				
60000				
70000				
80000				
90000				
100000				
<add more if needed>				

Implementation HINT: Divide the PI calculation tasks among the processes (cores) by dividing the *n* by *p*

Submission

Your submission should include the following:

3. **Makefile** that builds all required binaries:
    a. Part_N_type.out where N is the homework part and type can be serial/parallel
4. Serial code source file(s)
5. Parallel code source file(s)
6. Binary for both serial and parallel implementations
7. Document (word or excel) detailing the tabular results

All of the above should be zipped up into one file and uploaded on blackboard
If you are doing the bonus, add a comment in your code about it. (if applicable)
Naming convention of your zipped file – If you do not follow this, you get zero points:
FirstName_LastName_Homework_<n>.zip
Where n is the homework number.

Grading

Serial Implementation	5%
Distributed Implementation	30%
Correct execution Distributed	35%
Correct serial execution	5%
Speedup (greater than 1)	10%
Code clarity	5%
Timing competition	10%

Timing competition

You are competing with your colleagues for these 10% of the points. Lowest timing gets 10 full points. Highest timing gets 0.

Do not make up the speedup or the timing tables. You will get a 0 for all homework assignments (including previous).

Code clarity includes

- Use of variable names
- Function naming
- File names
- Formatting
- Comments where needed

Late submissions are subject to at least 20% grade deduction.

Reminder: If one or any of your binaries don't execute on student.rit.albany.edu, you get 0 points for this and any other homework where that happens.