Assignment 4
MPI Application

B. Wilkinson and Clayton Ferner: Modification date: Sept 30a, 2014

In this assignment, you will write and execute a sequential program for the Monte Carlo π calculation. Then you will write an MPI program for the Monte Carlo π calculation that implements general load-balancing workpool pattern. For convenience, this assignment will be done on your computer (Extra credit for using a remote cluster.)

Workpool

The workpool pattern can provide powerful load balancing whereby when a processor returns one result, it is given further work to do. *Figure 1* shows such a workpool with a task queue. Individual tasks are given to the slaves. When a slave finishes and returns the result, it is given another task from the task queue, until the task queue is empty. At that point, the master waits until all outstanding results are returned. (The termination condition is the task queue empty and all result collected.)

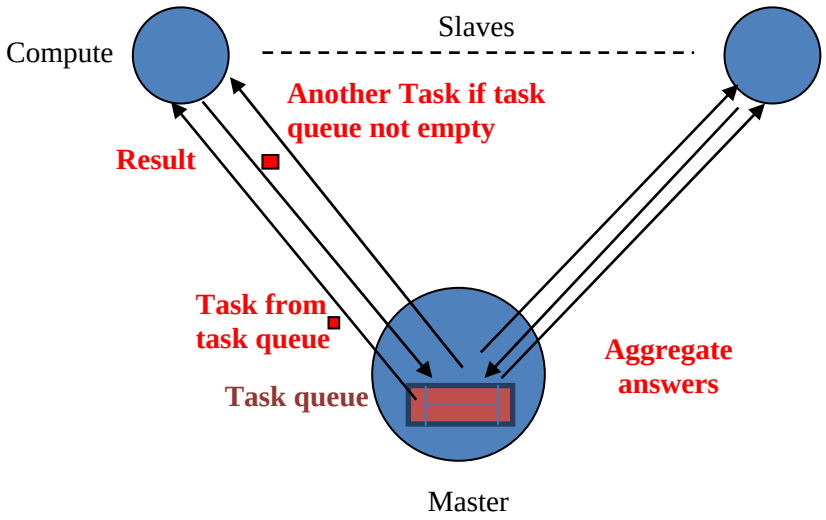

Figure 1: Workpool with a task queue

Monte Carlo π calculation

The Monte Carlo algorithm for computing π is well known and given in many parallel programming texts. It is a so-called embarrassingly parallel application particularly amenable to parallel implementation *but used more for demonstration purposes than as a good way to compute π.* However, it can lead to more important Monte Carlo applications. A circle is formed within a square. The circle has unit radius and the square has sides 2 x 2. The ratio of the area of the circle to the area of the square is given by $\pi(1^2)/(2 \times 2) = \pi/4$, as shown in Figure 2. Points within the square are chosen randomly and a score is kept of how many points happen to lie within the circle. The fraction of points within the circle will be $\pi/4$, given a sufficient number of randomly selected points.

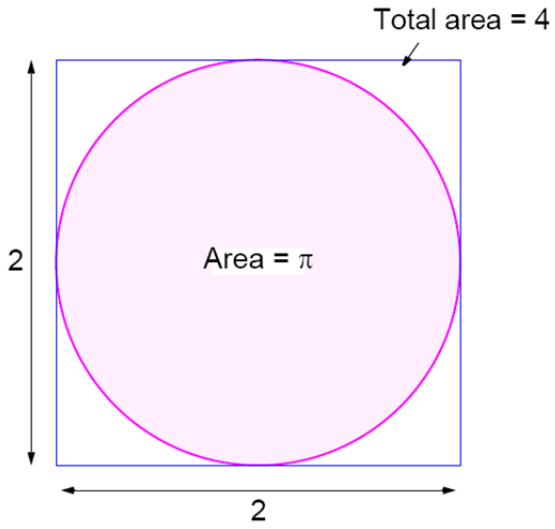

Figure 2 Monte Carlo π calculation

Usually only one quadrant is used as shown in Figure 3:

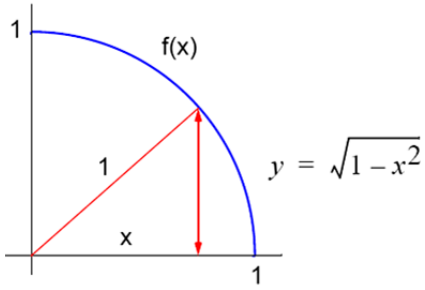

Figure 3 Using one quadrant of circle

so that the randomly selected points generated, x_r, y_r, are each between 0 and 1, and are counted as in the circle, if $y_r \leq \sqrt{1-x_r^2}$, i. e. $y_r^2 + x_r^2 \leq 1$. Note the integral $\int_0^1 \sqrt{1-x^2}\,dx = \pi/4$ is being evaluated.

Workpool. In the workpool implementation of the Monte Carlo π calculation here, the master process sends a different number to each of the slaves as shown in Figure 4. Each slave uses that number as the starting seed for their random number generator. Each slave then generates two random numbers as the coordinates of a point (x,y). If the point is within the quadrant of circle (i.e. $x^2 + y^2 <= 1$), a counter is incremented that is counting the number of points within the quadrant of circle. The slave continues to generate N points, and counts how many points are within the quadrant of circle. Each slave then returns the count to the master and asks for another starting seed. The master continues to give starting seeds to a maximum of S seeds and finally computes the final approximation for π. The final number of points will be N * S points and π/4 = (Total number of points within quadrant of circle)/(N * S).

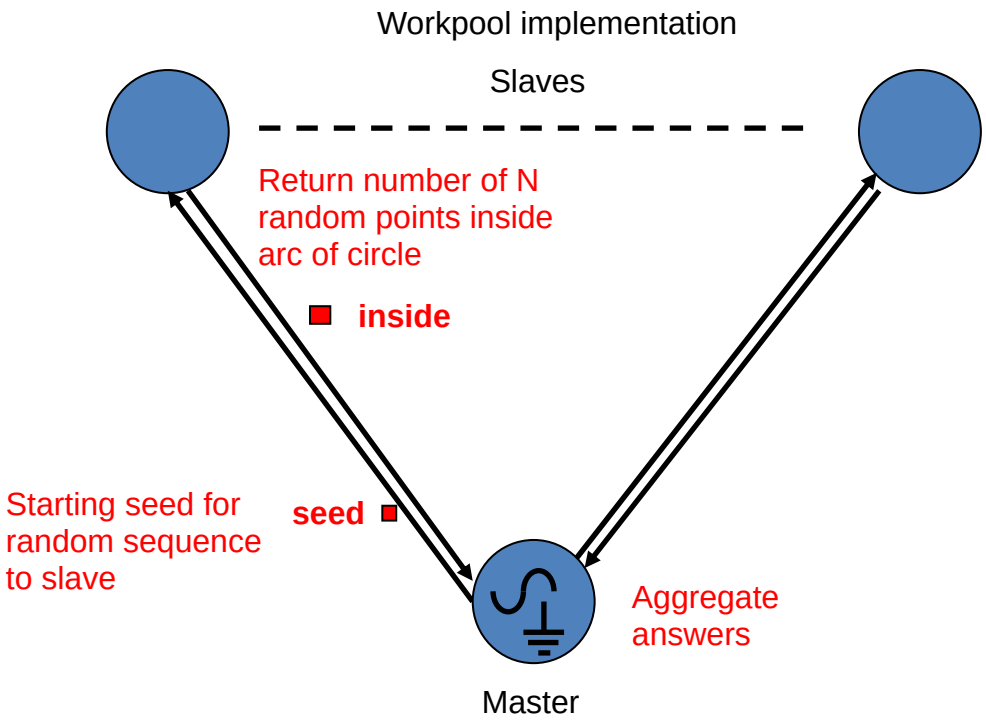

Figure 4 A Workpool implementation of Monte Carlo π calculation

Task 1

First write a sequential program in C to perform the Monte Carlo π calculation where *S* and *N* can be input (not a workpool). Test with at least N * S = 10,000 and other values. The program is to output both the value of π, its error from the exact value, and the execution time (done by instrumenting the code). Execute the program on your computer (VM or a native installation).

Task 2

Now write an MPI program to implement a workpool version for the Monte Carlo π calculation where *S* and *N* can be input. Test with 4 processes, and at least N * S = 10,000 and other values.

Execute the program on your computer (VM or a native installation).

Task 3 for extra credit (10%)

Execute the MPI program on a remote cluster (UNCC or UNCW). Run the MPI program with 4, 8, 12, and 16 processes. Compare the execution times with a sequential version and create graphs of the execution time and speedup.

Include in your submission document:

1. Sequential C program for the Monte Carlo π calculation
   a. Listing of your source program
   a. Screenshot of the program output on your computer, with various values of input

2. MPI program for the Monte Carlo π calculation
   b. Listing of your source program
   c. Screenshots of the program output on your computer, with various values of input
   d. Discussion on issues using random numbers

3 (Extra)

   a. Screenshot of the program output on a remote cluster
   b. Speedup graphs with 4, 8, 12, and 16 processes
   c. Conclusions

Assignment Submission

Produce a **single** pdf document that show that you successfully followed the instructions and performs all tasks.