Assignment 4
MPI Workpool Application

B. Wilkinson: Modification date: February 21, 2016

In this assignment, you are asked to write and execute a sequential program for the Monte Carlo π calculation. Then you area asked to write an MPI program for the Monte Carlo π calculation that implements a load-balancing workpool pattern, testing on your computer and on the cluster.

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 and is 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 1. 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 π/4, given a sufficient number of randomly selected points.

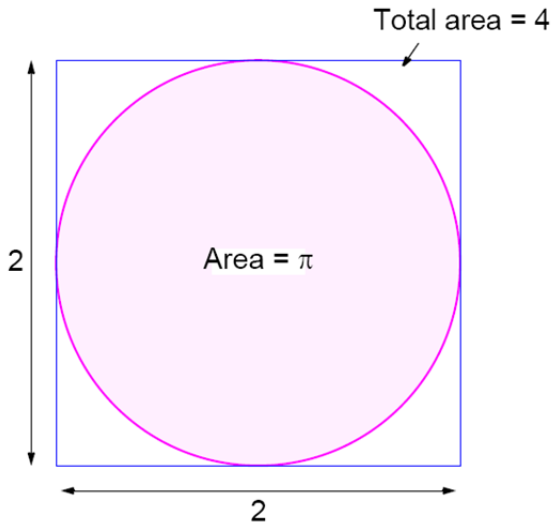

Figure 1 Monte Carlo π calculation

Usually in programs, only one quadrant is used as shown in Figure 2 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 (and in that way, any function could be evaluated).

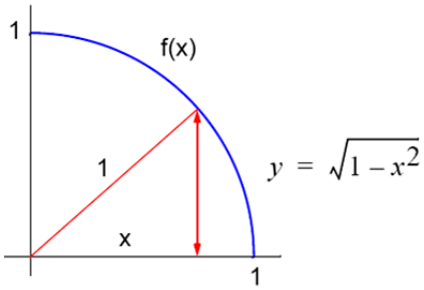

Figure 2 Using one quadrant of circle

Workpool

The workpool pattern is like a master-slave pattern but has a task queue that provides load balancing. Figure 3 shows a workpool with the task queue. Individual tasks are given to the slaves. When a slave finishes a task 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.

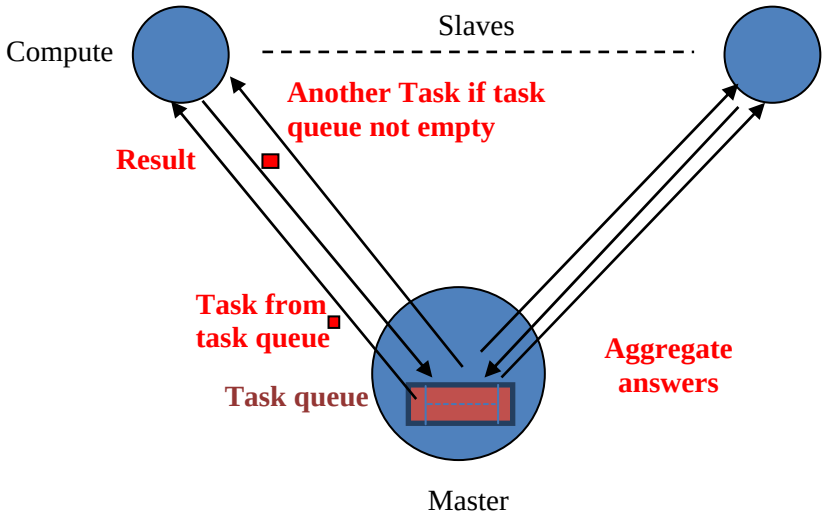

Figure 3: Workpool with a task queue

Implementing a Workpool for the Monte Carlo π calculation

In the particular workpool of the Monte Carlo π calculation you are asked to implement, the master process begins by sending a different number (called a "seed") to each of the slaves as shown in Figure 4. Each slave uses that number as the starting seed for its random number generator. The slaves generate 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 to indicate the point is within the quadrant of circle. Each slave repeats its actions with for a total of N points in each slave. The slaves then return how many of their N points are within the quadrant of circle to the master. When a slave returns its result, the master then gives the slave a new starting seed and the slave repeats its actions. The master continues to give starting seeds to a maximum of S seeds and then stops send out seeds. At that time, $N \times S$ random points will have been checked. Finally the master uses the results from the slaves to compute the approximation for π, given:

$$\frac{\pi}{4} = \frac{\text{Total number of points within quadrant of circle}}{N \times S}$$

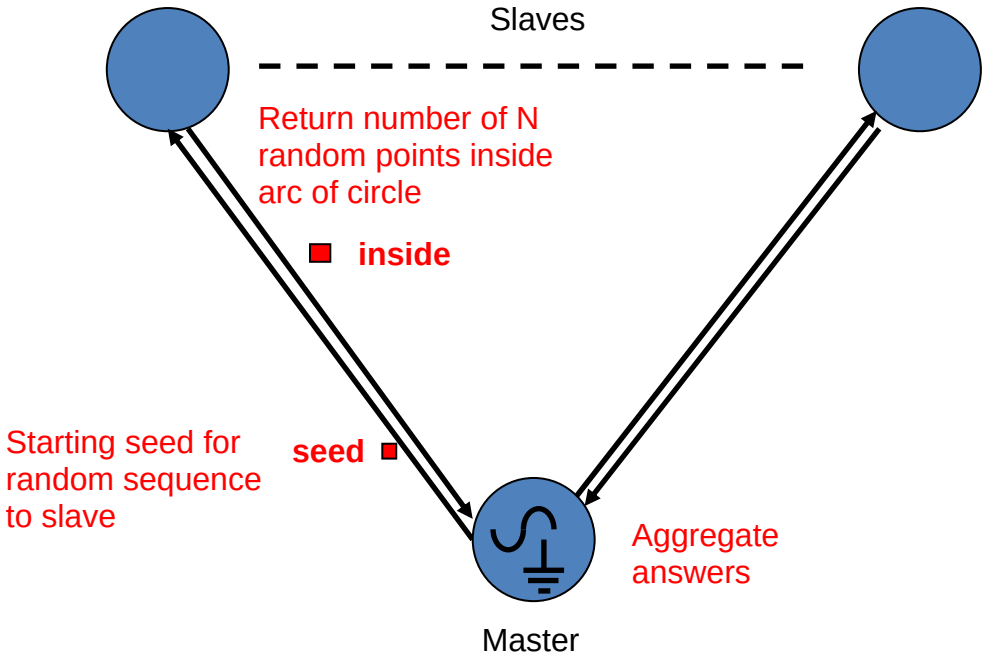

Figure 4 A Workpool implementation of Monte Carlo π calculation

Task 1 (25%)

First write a sequential program in C to perform the Monte Carlo π calculation (not implementing a workpool). Just one process performs the calculation and the total number of points is input at the keyboard. The process generates a pair of random numbers, determines whether the point is within the quadrant of circle. The process continues to generate random points, and counts how many points are within the quadrant of circle until all the points have been checked. It is acceptable to generate the random numbers each between 0 and 1 *not including 1* if it easier to generate random numbers. The error will be minimal with a large number of points.

The program is to output both the value of π and its error from the exact value, and also the execution time (done by instrumenting the code). For the exact value of π, use #define PI 3.14159265358979. Execute the program on your computer (VM or a native installation). Test with at least 10,000 points and other values.

Include in your submission document:

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

Task 2 (60%)

Now write an MPI program to implement a workpool version for the Monte Carlo π calculation where *S* and *N* can be input at the keyboard. It is critical that you implement the workpool as described earlier. You will lose many points if you do not do so. When a slave returns its result, the master then gives the

slave a new starting seed and the slave repeats its actions, until all S seeds have been sent. Try to make sure that the seeds being sent from the master to the slaves do not appear in the random number sequence number being generated by the slaves. This is very important to avoid interrelated sequences. This is not easy to guarantee, but develop a reasonable approach and justify.

Execute the program on your computer (VM or a native installation). Test with 4 processes, and at least $N * S = 10{,}000$ and other values.

Some clues on Task 2: It is important that you implement a workpool to get full credit. There are **S** tasks (seeds). Let us first assume **S** is larger than the number of slaves. One way to do a workpool is for the master to send one task to each of the slaves in turn and then wait for slaves to return results. As each slave returns a result, the master sends that slave a new task, until all S tasks have been sent. Then when the master receives a slave result, it sends a terminator message that the slave interprets for it to stop. The terminator message might be a unique tag or zero data. The destination can receive messages with any value of tag using **MPI_ANY_TAG** as the tag in MPI_Rec() and find out what tag has been received by checking the value of **status.MPI_TAG.** So one might have code such as:

```
MPI_Recv(…, …, …, …, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
tag = status.MPI_TAG;
if (tag == 999) terminate = 1;
```

One can accommodate when the number of slaves is greater than the number of tasks (an unlikely situation) by terminating those slaves not needed after send the **S** tasks to the first **S** slaves, but this is not strictly needed here as you are asked only to test with four processes. It is simple enough to add.

Include in your submission document:

1. Listing of your MPI source program for the Monte Carlo π calculation
2. Discussion on how you tried to avoid interrelated sequences
3. Screenshots of the program output on your computer, with various values of input

Task 3 Execute on the cci-gridgw.uncc.edu cluster (15%)

Add statements to your MPI program to print out which computer is being used for the master and each slave (use gethostname()). Compile on the cluster. Create a machines file to execute your program using four nodes, **cci-gridgw.uncc.edu, cci-grid05, cci-grid07, and cci-grid08**. Execute the MPI program with 4, 8, 12, and 16 processes. Create graphs of the execution time and speedup.

Include in your submission document:

1. Screenshots of the MPI program executing on the **cci-gridgw.uncc.edu** cluster
2. Graphs of execution time and speedup with 4, 8, 12, and 16 processes
3. Conclusions

Assignment Report Preparation and Grading

Produce a report that shows that you successfully followed the instructions and performs all tasks by taking screenshots and include these screenshots in the document.

Every task and subtask specified will be allocated a score so make sure you clearly identify each part/task you did. Make sure you include everything that is specified in the "Include ..." section at the end of each task/part as a minimum. **Include all code, not as screen shots of the listings but complete properly documented code listing.** The programs are often too long to see in a single screenshot and also not easy to read if small. Using multiple screenshots is not option for code. Copy/paste the code into the Word document or whatever you are using to create the report.

You will lose 10 points if you submit code as screenshots.

Assignment Submission

Convert your report into a ***single pdf*** document and submit the pdf file on Moodle by the due date as described on the course home page. It is extremely important you submit only a pdf file.

You will lose 10 points if you submit in any other format (e.g. Word, OpenOffice, ...). zip with multiple files is especially irritating.