# Operating System

# Report

# Assignment Simulation

# Based

## 10. Programming Assignment – 5.39 Page 250 Operating System By Galvin 9th Edition

5.39) Exercise 4.22 asked you to design a multithreaded program that estimated pi using the Monte Carlo technique. In that exercise, you were asked to create a single thread that generated random points, storing the result in a global variable. Once that thread exited, the parent thread performed the calculation that estimated the value of pi. Modify that program so that you create several threads, each of which generates random points and determines if the points fall within the circle. Each thread will have to update the global count of all points that fall within the circle. Protect against race conditions on updates to the shared global variable by using mutex locks.

Student Name: Shubham Nagar

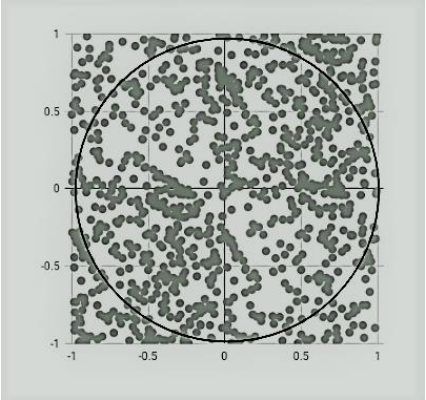
Student ID: 11605371 Section: K1605 Roll. No.: 44

Email Address: shbhmnagar@gmail.com

**GitHub Link: https://github.com/shbhmnagar/OS-Assignment**

## Estimating the value of Pi using Monte Carlo:

Random (x, y) points in a 2-D plane with domain as a square of side 1 unit. There is a imaginary circle inside the same domain with diameter 1 unit and inscribed into the square. We then calculate the ratio of the number points lied inside the circle and the total number of generated points.




The area of the square is 1 unit$^2$ while that of the circle is $\pi/4$. Now for the very large number of generated points,

$\Pi = 4 * (\text{no. of points generated inside the circle / no. of points generated inside square})$

## Algorithm:

1. Initialize circle_points, square_points and interval to 0.
2. Generate random point x.
3. Generate random point y.
4. Calculate d = x*x + y*y.
5. If d <= 1, increment circle_points.
6. Increment square_points.
7. Increment interval.
8. If increment < NO_OF_ITERATIONS, repeat from 2.
9. Calculate pi = 4*(circle_points/square_points).
10. Terminate.

Complexity: O (m / n) complexity. “m” is no. of points generated and “n” is no. of threads to be created.

## Conditions in Problem (Protect Against Racing Condition):

```
pthread_mutex_lock(&mutex);
incircle += incircle_thread;
pthread_mutex_unlock(&mutex);
```

“incircle” is a shared variable.

## Test Cases:

1. No. of threads = -1 or string and No. of points = -1 or string: Program asks for valid input again.

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
-1

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
-1

Invalid Input Enter Again
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
```

2. No. of threads = 1 and No. of points = 1: Program RUNS link a single threaded.

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
1

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
1

Calculating ...



Thread 1 is calculating
Point 1 ( 0.527567 , 0.121131 ) inside circle of diameter 1 unit

Final Estimated Value of Pi: 4.000000
Time: 0 sec
```

3. No. of threads = 1 and No. of points = 10.: Single thread is calculating all the points.

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
10

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
1

Calculating ...



Thread 1 is calculating
Point 1 ( 0.527567 , 0.121131 ) inside circle of diameter 1 unit
Point 2 ( -0.193739 , -0.465941 ) inside circle of diameter 1 unit
Point 3 ( -0.032782 , -0.911362 ) inside circle of diameter 1 unit
Point 4 ( 0.693669 , 0.274279 ) inside circle of diameter 1 unit
Point 5 ( -0.758932 , 0.621623 ) inside circle of diameter 1 unit
Point 6 ( -0.375909 , -0.226647 ) inside circle of diameter 1 unit
Point 7 ( 0.070143 , -0.254879 ) inside circle of diameter 1 unit
Point 8 ( 0.630645 , -0.164869 ) inside circle of diameter 1 unit
Point 9 ( -0.415574 , -0.956072 ) inside circle of diameter 1 unit
Point 10 ( 0.405709 , -0.155367 ) inside circle of diameter 1 unit

Final Estimated Value of Pi: 3.600000
Time: 0 sec
```

4. No. of threads = 4 and no. of points = $9^{10}$ : Result is calculated much faster and accurate.

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
9999999999

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
4

Calculating ...


Final Estimated Value of Pi: 3.141595
Time: 95 sec
```

5. No. of threads = 51 and no. of points = 80: program asks for user input again if the no. of threads is greater than 12.

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
134

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
51

Invalid Input Enter Again | Thread count is too large
```

6. No. of threads = No. of points: each thread will have one point to calculate.

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
4

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
4

Calculating ...


Point 1 ( 0.527567 , 0.121131 ) inside circle of diameter 1 unit
Thread 1 is calculated

Point 1 ( 0.612510 , 0.622072 ) inside circle of diameter 1 unit
Thread 2 is calculated

Point 1 ( 0.117964 , -0.510598 ) inside circle of diameter 1 unit
Thread 3 is calculated

Point 1 ( 0.964660 , 0.358786 ) inside circle of diameter 1 unit
Thread 4 is calculated

Final Estimated Value of Pi: 3.000000
Time: 0 sec
```

7. No. of threads < No. of points:

```
root@Conan:/mnt/d/OS-Assignment# gcc a.c -lpthread
root@Conan:/mnt/d/OS-Assignment# ./a.out
Enter the total no. of points you want to generate ('Accuracy of pi ~ no. of points')
2

No. of threads you want to use for calulation ('faster caluclation ~ no. of points ~ (depends on C.P.U.)')
4

Calculating ...


Final Estimated Value of Pi: 3.000000
Time: 0 sec
```

GitHub Link: https://github.com/shbhmnagar/OS-Assignment