LEBANESE AMERICAN UNIVERSITY
Department of Computer Science and Mathematics
CSC 447: Parallel Programing
Spring 2021

Lab IV (MPI)

Problem 1: Search Preimage

Write a parallel program that takes, as command line arguments, 3 integers A, B and C. Your program will search integers between A and B for a value J such that F(J) = C, where F(x) is any predefined function. **Note: do not forget to handle the case when no preimage is found**
Assume in this case $F(x) = 3x^3+5x+20$. The range should be divided among the available processes. Each process searches for a preimage within its range and sends its results back to the master. The master prints the results into an output text file output.txt.

Sample input:
A=0, B=1000, C=3070.

Sample output:

Process i found J = 10 verifying F(J) = 3070

Compile and run this MPI program and verify its output by following these steps:

- Use an IDE of your choice to create and save your program with *YourName*_Prob1.c.
- Compile your program within the IDE or from terminal window by using the command: mpicc –o Problem1 YourName_Prob1.c
- Then, run your MPI program by typing the command in a terminal widow: mpirun -n <NumCores> <compiledProgram>
- Verify that your program gives the correct output for 1, 2, and 4 cores

Measure the execution time in your program using the function $MPI\_Wtime()$. This can be done as follows:

```
Double start_time = MPI_Wtime();
//Insert your parallel code here
if(rank==0) {
double end_time=MPI_Wtime();
printf("Wallclock time elapsed: %.8lf seconds\n",end_time-
start_time);
}
```

- Report in a tabular form the execution time of your program when using 1, 2, and 4 processors.
- Draw your conclusions on the speedups obtained.

Problem 2: Compute the Value of Pi

Write a parallel program to calculate π using the Monte Carlo method. The process is as follws: randomly located points are generated within a 2×2 square which has a circle inscribed within it. The algorithm generates a large number of points and checks to see if the coordinates, x and y, of each point are inside the circle- x2+y2≤1. The ratio, P, of points inside the circle to the total number of points tried is calculated. Using this ratio, the approximation of π can be computed by assuming that P approaches the value of ratio of the area of the circle to the area of the square when the number of points, niter, is large. Thus we can say: $P= (\pi r^2)/4r^2 = \pi/4$ and solve for π. To do this, P is multiplied by 4 and the result is an approximation of π.
Use a total number of iterations **N=10000000**. In your program, the iterations are divided among the available processes such that each process will run a subset of these iterations in parallel. Each process will compute the number of points that it has found inside the circle and report the results back to the master. The master will aggregate the results to compute the approximate value of pi.

Hint:

To generate a random number in the range [0,1] in C:

```
//First set the seed for the random number generator (Each
process should have a different seed)
srand(rank+1);

//Next generate a random number in the range [0,1]
Double x=((double)rand())/RAND_MAX;
```

- Save your program with the name *YourName*_Prob2.c.
- Report in a tabular form the execution time of your program when using 1, 2, and 4 processors.
- Draw your conclusions on the speedups obtained.
- Save your program with the name *YourName*_Prob2.c.

Note: A report should be submitted where you describe your parallel implementation and report in, tabular form, the execution times and speedups obtained.

Submission Instructions:

You are required to submit your solutions **by Tuesday March 16 at 8:00 am**. Place your code

and report in one zipped folder and submit on the provided submission link on Blackboard.