CSC718 Operating Sys & Parallel Programming

Homework 2 OpenMPI Programming Assignments

The homework assignment will be graded based on the following criteria:

- Accuracy: 1) the solution meets specific requirements in the problem description; 2) the solution produces correct results; 2) the procedures adopted in the solution are technically sound.
- Efficiency: efficiency will be one of the criteria when grading programing assignment. The solution should produce the desired results efficiently.
- Effort/neatness: the solution includes excellent effort, and all relate work is shown neatly and organized well.

Homework assignment feedback will be available through the DropBox folder on D2L.

For all the programming assignments, you can choose any operating systems you want. I will usually provide C/C++ samples for the programming assignments. If you prefer to use other languages, e.g., Java, they are accepted too. A README.txt is required to submit any programming assignments. In the README.txt, you need to provide the following information:

1) How to compile your program?
2) How to run your program?
3) What is the average running time of your program?
4) What are the expected output results when I run your program?
5) Any descriptions which may help me to compile, run, and verify your answers. (FYI: I check every programming assignment turned in!)

Zip all you source code, project files, supporting files, and README.txt and submit the all-in-one zip file together to the D2L Dropbox. If you have any questions about the homework, please let me know.

(20 points) Programming Assignment 1: A prime number is a positive integer evenly divisible by exactly two positive integers: itself and 1. The first five prime numbers are 2, 3, 5, 7, and 11. Sometimes two consecutive odd numbers are both prime. For example, the odd integers following 3, 5, 11 are all prime numbers. Write a sequential program to determine for all integers less than 1,000,000, the number of times that two consecutive odd integers are both prime numbers.

```
mpiuser@Jefferson:~/Desktop/KC$ ./HW2-1
There are 8170 instances of consecutive primes for all integers less than 1,000,000.

<Program Runtime: 68.7962s>
```

(30 points) Programming Assignment 2: A prime number is a positive integer evenly divisible by exactly two positive integers: itself and 1. The first five prime numbers are 2, 3, 5, 7, and 11. Sometimes two consecutive odd numbers are both prime. For example, the odd integers following 3, 5, 11 are all prime numbers. Write a parallel program using openMPI to determine for all integers less than 1,000,000, the number of times that two consecutive odd integers are both prime numbers.

```
mpiuser@Jefferson:~/Desktop/KC$ ./mpiScript
HW2-2                                                 100%   17KB   9.9MB/s   00:00
HW2-2                                                 100%   17KB   9.6MB/s   00:00
HW2-2                                                 100%   17KB   9.1MB/s   00:00
mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 1 -machinefile machinefile.dsu ./HW2-2
There are 8170 instances of consecutive primes for all integers less than 1,000,000.

<Program Runtime: 91.2304s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 2 -machinefile machinefile.dsu ./HW2-2
There are 8170 instances of consecutive primes for all integers less than 1,000,000.

<Program Runtime: 66.3051s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 3 -machinefile machinefile.dsu ./HW2-2
There are 8170 instances of consecutive primes for all integers less than 1,000,000.

<Program Runtime: 47.7259s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 4 -machinefile machinefile.dsu ./HW2-2
There are 8170 instances of consecutive primes for all integers less than 1,000,000.

<Program Runtime: 38.0741s>
```

(30 points) Programming Assignment 3: The gap between consecutive prime numbers 2 and 3 is only 1, while the gap between consecutive primes 7 and 11 is 4. Write a parallel program using openMPI to determine, for all integers less than 1,000,000, the largest gap between a pair of consecutive prime numbers.

```
mpiuser@Jefferson:~/Desktop/KC$ ./mpiScript
HW2-3                                                 100%   17KB   9.6MB/s   00:00
HW2-3                                                 100%   17KB   3.4MB/s   00:00
HW2-3                                                 100%   17KB  10.8MB/s   00:00
mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 1 -machinefile machinefile.dsu ./HW2-3
The largest gap between a pair of consecutive prime numbers for all integers from 2 to 1,000,000 is
1,450.

<Program Runtime: 87.4439s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 2 -machinefile machinefile.dsu ./HW2-3
The largest gap between a pair of consecutive prime numbers for all integers from 2 to 1,000,000 is
1,450.

<Program Runtime: 60.3154s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 3 -machinefile machinefile.dsu ./HW2-3
The largest gap between a pair of consecutive prime numbers for all integers from 2 to 1,000,000 is
1,450.

<Program Runtime: 43.7801s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 4 -machinefile machinefile.dsu ./HW2-3
The largest gap between a pair of consecutive prime numbers for all integers from 2 to 1,000,000 is
1,450.

<Program Runtime: 74.2195s>
```

(20 points) Programming Assignment 4: The value of the definite integral

$$\int_0^1 \frac{4}{1+x^2}\,dx$$

is π . We can use numerical integration to compute π by approximating the area under the curve. A simple way to do this is called the rectangle rule (Figure 4.7). We divide the interval [0, 1] into *k* subintervals of equal size. We find the height of the curve at the midpoint of each of these subintervals. With these heights we can construct k rectangles. The area of the rectangles approximates the area under the curve. As *k* increases, the accuracy of the estimate also increases.

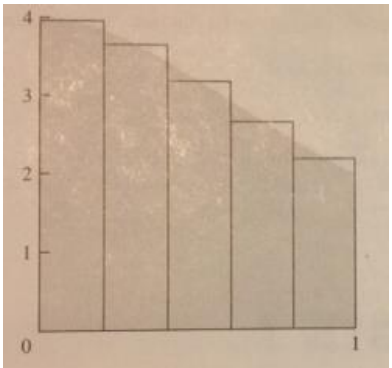

Figure 4.7 The rectangle rule is a simple way to approximate the area under a curve. In this example the function is $4/(1+x^2)$, and the area under the curve between 0 and 1 is π.

A C program that uses the rectangle rule to approximate π appears in Figure 4.8. Write a parallel MPI program to computer π using the rectangle rule with 1,000,000 intervals. (Use the virtual cluster provided in the class to develop and test the program.)

```c
#include <stdio.h>
#define INTERVALS 1000000
int main(int argc, char* argv[])
{
 double area; /* The final answer */
 double ysum; /* Sum of rectangle heights */
 double xi;   /* Midpoint of interval */
 int i;
 ysum= 0.0;
 for(i=0;i<INTERVALS;i++)
  {
   xi= ((1.0/INTERVALS)*(i+0.5)); /* Midpoint of interval */
   ysum += 4.0/(1.0 + xi*xi);
  }
   area=ysum*(1.0/INTERVALS);
   printf("Area is %13.11f\n",area);
 return 0;
}
```

Figure 4.8 A C program to compute the value of π using the rectangle rule.

```
mpiuser@Jefferson:~/Desktop/KC$ ./mpiScript
HW2-4                                                         100%   17KB  10.9MB/s   00:00
HW2-4                                                         100%   17KB   9.0MB/s   00:00
HW2-4                                                         100%   17KB  10.4MB/s   00:00
mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 1 -machinefile machinefile.dsu ./HW2-4
pi is 3.14159265359

<Program Runtime: 0.0158s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 2 -machinefile machinefile.dsu ./HW2-4
pi is 3.14159265359

<Program Runtime: 0.0052s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 3 -machinefile machinefile.dsu ./HW2-4
pi is 3.14159265359

<Program Runtime: 0.0036s>

mpiuser@Jefferson:~/Desktop/KC$ mpirun -np 4 -machinefile machinefile.dsu ./HW2-4
pi is 3.14159265359

<Program Runtime: 0.0101s>
```

For each programming assignment, you need to write the code, and run it on Rushmore Cluster to get the running time for the following number of processors: 1, 2, 3, 4. Put this into a table to see the performance difference by parallel computing.

Benchmark running on Rushmore with different number of processors.

Problem	Np=1	Np=2	Np=3	Np=4
Program1.c	68.7962			
Program2.c	91.2304	66.3051	47.7259	38.0741
Program3.c	87.4439	60.3154	43.7801	74.2195
Program4.c	0.0158	0.0052	0.0036	0.0101