ASSIGNMENT 1 MPI – Programming

Course Instructor : Dr. Pushparaj Shetty D

I. Write an MPI program to find maximum value in an array of 600 integers with 6 processes and print the result in root process using MPI_Reduce call. Compute time taken by the program using MPI_Wtime() function.

II.

a. Write a parallel program to compute the prefix sums for a large sequence X[1..N] of integers. Take large enough values of N. You need to report the best way to partition the program and running them in parallel which gives the best performance.

b. Your code should allow easy changing of parameters like the number of processors and the input size N. For a given input size N you can generate the input sequence of integers X[1..N] by a

series of calls to a function that generates random numbers.

c. Prepare a brief report of your experiments (for e.g. give tables that show the runtime of the two

programs for varying number of threads, processors, input size N, etc).

An example of prefix computation:

If a given ordered set is {5, 3, -6, 2, 7, 10, -2, 8} then the output is {5, 8, 2, 4, 11, 21, 19, 27}

III.

a. Implement Sieve of Eratosthenes for finding list of prime numbers up to a given list

Algorithm description:

Sieve of Eratosthenes is a simple and ancient algorithm used to find the prime numbers up to any given limit. It is one of the most efficient ways to find small prime numbers.

For a given upper limit *n* the algorithm works by iteratively marking the multiples of primes as composite, starting from 2. Once all multiples of 2 have been marked composite, the multiples of next prime, i.e 3 are marked composite. This process continues until p ≤ sqrt(n) where p is a prime number.

In the following algorithm, the number 0 represents a composite number.

1. To find out all primes under n, generate a list of all integers from 2 to n. (**Note**: 1 is not prime)
2. Start with a smallest prime number, ie p=2.
3. Mark all the multiples of p which are less than n as composite. To do this, mark the value of the numbers (multiples of p) in the generated list as 0. **Do not** mark p itself as composite.
4. Assign the value of p to the next prime. The next prime is the next non-zero number in the list which is greater than p.
5. Repeat the process until p≤sqrt(n).

Illustration:

Generate all the primes less than 11

Create a list of integers from 2 to 10. list = [2, 3, 4, 5, 6, 7, 8, 9, 10]

1. Start with p=2.
2. Since 2^2≤10, continue.
3. Mark all multiples of 2 as composite by setting their value as 0 in the list. list = [2, 3, 0, 5, 0, 7, 0, 9, 0]
4. Assign the value of p to the next prime, ie 3.
5. Since 3^2≤10, continue.
6. Mark all multiples of 3 as composite by setting their value as 0 in the list. list = [2, 3, 0, 5, 0, 7, 0, 0, 0]
7. Assign the value of p to 5.
8. Since 5^2≰10, stop.

We are done!

The list is [2, 3, 0, 5, 0, 7, 0, 0, 0]. Since all the numbers which are 0 are composite, all the non-zero numbers are prime. Hence the prime numbers less than 11 are 2, 3, 5, 7. □

b. Compare the time taken by the Serial and Parallel version of the program

c. Prepare a brief report of your experiments (for e.g. give tables that show the runtime of the two programs for varying number of processors, input value N, etc).

IV.

a. Write a parallel program to find the minimum and maximum element in a given array of large size N.
b. Your code should allow easy changing of parameters like the number of processors, the input size N. Use random function to generates random numbers.

c. Prepare a brief report of your experiments (for e.g. give tables that show the runtime of the two programs(serial and parallel) for varying number of processors, input size N, etc).

V.

a. Write parallel programs to implement Merge sort and Quick sort.

b. Prepare a comparison report of two algorithms for both serial and parallel versions. (for e.g. give tables that show the runtime of the two algorithms for varying number of threads, input size N, etc).

VI . Write MPI program to calculate the product of two matrices A (of size N*32) and B (of size 32*N), which should be a N*N matrix. Design a parallel scheme for computing matrix multiplication,

a. Blocking P2P (point-to-point) communication
b. Collective communication
c. Non-blocking P2P communication.
d. Observe the running time of your programs; change some of the parameters to see how it is associated with N and communication type . Write down the observations.

VII . The value of π is computed mathematically as follows:

$$\int_0^1 4/\left(1+x^2\right)=\pi$$

Write a MPI program to compute π using MPI _Bcast and MPI _Reduce. Compare execution time for serial code and parallel code.

- Broadcast the total steps value (num steps) to all the processs (process 0 could broadcast).

Broadcast a message from the process with rank “root” to all other processes of the group.

– Example: MPI_ Bcast(&n, 1, MPI_INT, 0, MPI_ COMM _WORLD);. process 0 broadcasts n, of type MPI_INT, to all the processs

MPI _COMM _WORLD.

- Each process calculates its partial π value.

Partial π = (4.0 /(1.0+x^2))× step).

- Use MPI_ Reduce to reduce the partial π values generated by each process. MPI _Reduce is executed by every process.

MPI_ Reduce : Reduce values on all processes to a single value.

Example:

MPI_ Reduce(&mypi, &pi, 1, MPI_ DOUBLE, MPI_ SUM, 0, MPI_ COMM_ WORLD);:

Perform MPI _SUM on mypi from each process into a single pi value and store it in process 0.

VIII. Write a MPI program to compute Dot Product

The task for this question is calculate the dot product of two arrays and to perform reduction on the product array. The working follows:

- The root process populates two arrays, each of size N.
- The arrays are distributed amongst P processes in the communicator.
- Each process calculates the dot product and the reduction operation.
- Root performs the final reduction operation to obtain the final answer.

The values for N and P are your choice. The arrays can be filled with random numbers. Implement two versions of this code.

a. Each process gets an equal sized chunk of both the arrays. (using MPI _Scatter).

b. Each process gets an unequal sized chunk of both the arrays. (using MPI _Scatterv).