School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS4040A & COMS7045A: MPI Lab 1

May 4, 2025

Objectives

- Apply the basic MPI functions to write simple MPI programs
- Compile and run MPI program on a cluster

Questions

1. Use MPI to implement the histogram program discussed in Lec7 slides. Use one process to generate (or read) the input data and distribute it among the processes, and also use the same process to output the histogram.

2. Implement matrix-vector multiplication, $\mathbf{y} = A\mathbf{x}$, where $A \in \mathbb{R}^{m\times n}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, using MPI. Consider the following data distribution strategies and implement both:

    - Row-block distribution of matrix A: The matrix A is distributed by rows across the MPI processes, and the entire input vector $\mathbf{x}$ is replicated and available on each process.
    - Row-block distribution of matrix A with block distribution of vector $\mathbf{x}$: The matrix A is again distributed by rows, but the input vector $\mathbf{x}$ is distributed by blocks, such that each process holds only a distinct portion of $\mathbf{x}$.

3. Write an MPI program that computes the sum of an array of random numbers. Implement it using the following data distribution methods, respectively.

    - Block distribution
    - Cyclic distribution
    - Block-cyclic distribution

4. We can use trapezoidal rule to approximate the area between the graph of a function, $y = f(x)$, two vertical lines along $y - axis$, and the x-axis (see Figure 1 (a)). The basic idea is to divide the interval on the x-axis into n equal subintervals. Then we approximate the area lying between the graph and each subinterval by a trapezoid whose base is the subinterval, whose vertical sides are the vertical lines through the endpoints of the subinterval, and whose fourth side is the scant line joining the points where the vertical lines cross the function graph. If the endpoints of the subinterval are x_i and x_{i+1}, then the length of the subinterval is $h = x_{i+1} - x_i$. Also, if the lengths of the two vertical segments are $f(x_i)$ and $f(x_{i+1})$, then the area, A, of the trapezoid is

$$A = \frac{h}{2}\left[f(x_i) + f(x_{i+1})\right]. \tag{1}$$

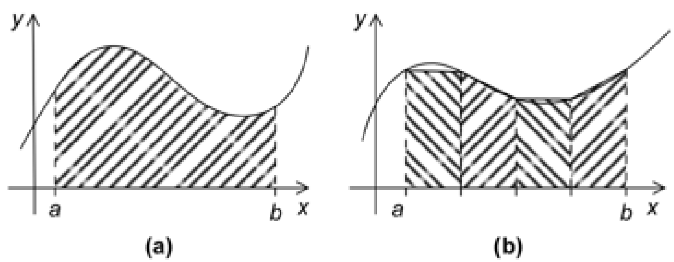

Figure 1: (a) Area to be approximated; (b) Approximate the area using trapezoids.

Given the region boundaries are determined by $x = a$ and $x = b$, and there are n equal subintervals, then

$$h = \frac{b-a}{n}. \tag{2}$$

Now, we can determine all the end points of the n subintervals as

$$x_0 = a, x_1 = a + h, x_2 = a + 2h, \ldots, x_{n-1} = a + (n-1)h, x_n = b.$$

The sum of the areas of the trapezoids, which gives the approximate area, is

$$h\left[f(x_0)/2 + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + f(x_n)/2\right]. \tag{3}$$

The pseudo code for a serial program could be:

```
h = (b-a)/n;
sum = (f(a) + f(b))/2.0;
for(i = 1; i < n; i++){
  x_i = a+i*h;
```

```
  sum += f(x_i);
}
sum = h*sum;
```

Implement an MPI program to compute the area of $f = \frac{4}{1+x^2}$ between interval $[0.0, 1.0]$ in parallel.

5. Write a simple MPI program in which two processes exchange packets of data back and forth multiple times, and record the total time taken. This type of program is commonly referred to as an MPI **ping-pong** benchmark.

   The packets should consist of arrays of dummy floating-point numbers, with sizes varying from 1 to 10,000 elements. Each packet should be sent back and forth between the two processes 100 times. Record the total time taken for each packet size.

   Using the recorded timing data, compute:

   - The average time per send/receive operation.
   - The effective transfer rate (bandwidth) in bytes per second.

6. A small college wishes to assign unique identification numbers to all current and future students. The administration is considering using six-digit identifiers but is unsure whether there will be a sufficient number of valid combinations, given certain constraints on what constitutes an "acceptable" identifier.

   Write parallel programs using both MPI and OpenMP to compute the total number of valid six-digit combinations using the digits 0–9, subject to the following constraints:

   - The first digit must not be 0.
   - No two consecutive digits may be the same.
   - The sum of the digits must not be equal to 7, 11, or 13.