Computer Science 335
Parallel Processing and HPC
Siena College
Fall 2024

Programming Project 3: MPI Recursive Digit Sum

Due: 9:00 AM, Tuesday, October 1, 2024

In this programming project, you will write your first more substantial MPI program.

You may work alone or with a partner or two on this programming project. AI assistance is permitted but must be documented, and intermediate development steps must be committed to your repository (regardless of the use of AI tools).

Learning goals:

1. To demonstrate the ability to write a program using MPI's point-to-point communication functions

---

Getting Set Up

In Canvas, you will find a link to follow to set up your GitHub repository, which will be named `mpirds-proj-yourgitname`, for this programming project. Only one member of the group should follow the link to set up the repository on GitHub, then others should request a link to be granted write access.

All GitHub repositories must be created with all group members having write access and all group member names specified in the `README.md` file by 4:00 PM, Thurday, September 26, 2024. This applies to those who choose to work alone as well!

While you will not be ready to complete the entire program until after you have completed Lab 4, you can get started on understanding the problem and writing a serial version to solve it.

Recursive Digit Sum

We define the "recursive digit sum" of a number to be the end result of taking the number's digits, adding them up, then taking that sum's digits, adding them up, repeating this process until a single digit number results. So we always get a result in the range from 0 to 9.

For example, the "recursive digit sum" of 182,172 is 3, because the sum of the digits of 182,172 is 21, and the sum of 21's digits is 3.

Programming Task

Write a C program using MPI called `mpirds.c` to study the distribution of the recursive digit sums of the numbers 0 to N-1 for some large N. Specific requirements are listed below.

- Your program should work for any number of processes where each process is responsible for computing the recursive digit sum of some part of the range.
- The program may read the value for N from the terminal or specify it as a command-line parameter. You may require that the number of processes evenly divides N, but in that case you must check for that in your program and return an error if it doesn't. Be sure to call the `MPI_Finalize` function on all processes before exiting with the error condition if it is necessary to terminate the program for this reason.
- Each process should determine its range of values, compute a local array of counts for how many times each digit has occurred. At the end, all processes except rank 0 should send their array to rank 0. The rank 0 process will receive each and accumulate a global sum and print the result.
- Each process may print out its own range of values that it will consider, but all remaining printouts should be done only by the rank 0 process.
- Each process should also measure the time taken to process its range of values. Use `MPI_Wtime` for this. These times should also be sent to the rank 0 process and at the end, the rank 0 process should report all of those times as well as the fastest, slowest, and average time taken across all processes. These times should include only the main computation loops on each process. Additionally, the rank 0 process should measure and report a total time, including all compuation and communication (everything between the `MPI_Init` and where you report the timings before `MPI_Finalize`).

My program's result for N=1048576 and 8 processes are in your repository as `sampleoutput.txt`. Your program should produce similar outputs in a similar format.

Run your program for N=1073741824 for 2, 4, 8, 16 and 32 processes, redirecting the results to files `run2.txt`, `run4.txt`, ..., `run32.txt`.

Submission

Commit and push!

Grading

This assignment will be graded out of 40 points.

Feature	Value	Score
MPI setup correctness	2	
Command-line processing/checking	3	
Determining task ranges	3	
Local RDS frequency computations	5	
Gather/report frequencies	10	
Timings	7	
Documentation/style	5	
`run*.txt` files	5	
Total	40	