CX 4220 / CSE 6220 Introduction to High Performance Computing
Spring 2017
Programming Assignment 1
Due Friday, March 3

In this assignment you will implement an MPI application that evaluates a simple polynomial in the form of,

$$y = P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$$

Making use of what you have learned in class regarding efficient communication between processors, you are to implement the functions `Scatter`, `Broadcast`, `ParallelPrefix` and `PolyEvaluator` using the MPI primitives `MPI_Send/MPI_Recv` or `MPI_Isend/MPI_Irecv` for a hypercube network.

Function Name	**Purpose**[1]	**Runtime**[2]	**Space**
`Scatter`	Block decompose an $O(n)$ size array of real numbers (residing in processor P_i) among p processors such that each processor will have $O(\frac{n}{p})$ elements	$O(n)$	$O(\frac{n}{p})$
`Broadcast`	Broadcast a real number (residing in processor P_i) among p processors	$O(\log p)$	$O(1)$
`ParallelPrefix`	Run parallel prefix algorithm on p processors by taking an $O(\frac{n}{p})$ element local array of real numbers and sum/product as the operator as inputs.	$O(\frac{n}{p} + \log p)$	$O(\frac{n}{p})$
`PolyEvaluator`	Evaluate the polynomial function $P(x)$ for a $O(\frac{n}{p})$ element local array of constants and an x value. Returns result from the evaluation.	$O(\frac{n}{p} + \log p)$	$O(\frac{n}{p})$

Your design of the algorithms should adhere to the following:

- You will be graded based on correctness of your algorithms and for efficient implementation of the functions `Broadcast`, `ParallelPrefix` and `PolyEvaluator`.
- You can implement a naive `Scatter` function such that Processor P_i having array of size $O(n)$ simply sends blocks of size $O(\frac{n}{p})$ individually to rest of the processors. (i.e., Processor-0 can use $O(n)$ space and does $O(n)$ work for this. At the end each processor will have $O(\frac{n}{p})$ portion of the array).
- DO NOT USE the existing functions `MPI_Bcast` `MPI_Scatter` or `MPI_Scan`.

[1] for any integer n and p such that $n \geq p > 0$

[2] latency τ and bandwidth μ constants are omitted in the runtime

Code Framework

Download the code framework provided at `http://b.gatech.edu/2kSZFP1`. Implement the functions defined in `mpi_evaluator.h` header file inside `mpi_evaluator.cpp` source file using `C` language (avoid having any `C++` functions or data structures in your final submission). Additionally in order for you to test the correctness of the results from running your parallel algorithm you may implement the serial version of the polynomial evaluation by implementing the functions defined in `evaluator.h` header file inside `evaluator.cpp` source file.
Please refer to the `README.md` provided with the code framework for more details about the structure of the framework.

Input Format

Input to the application will be 2 files containing the constants ($a_0, a_1, \dots, a_{n-1}$ where $a_i \in \mathbb{R}$) of the polynomial equation and a set of values for x ($x_0, x_1, \dots, x_{m-1}$ where $x_i \in \mathbb{R}$) to evaluate the polynomial on.

eg:

Listing 1: Constants in polynomial equation

n
a_0
a_1
a_2
$\dots$
$\dots$
$\dots$
a_{n-1}

Listing 2: Values of x to evaluate upon

m
x_0
x_1
x_2
$\dots$
$\dots$
$\dots$
x_{m-1}

Teams

You are expected to work in teams of two. You may divide the work among the team members as you see fit, however we ask the student to be knowledgeable of other team members portion of contribution to the assignment. All team members in a team will be given the same grade for the programming assignment.

Testing Your Implementation

You are given access to the Jinx computing cluster. You should test your code locally in your own computer before you move the testing to the Jinx cluster. However due to limited resources in Jinx we urge you to not to wait until the last few days before the assignment deadline to use Jinx.

Before starting the assignment, test your access to Jinx. If you have trouble accessing it, please contact the TA's immediately at hpc17ta@lists.gatech.edu. For all other issues you might face in compiling and/or executing your programs, we recommend you post the problem on piazza for everyones benefit of learning.

Submission Guideline

This assignment is due **March 3, 11:55pm EST** on T-Square. One member from each team should submit zip/tar file containing the following.

1. A text file containing the names of all team members and their contribution in terms of percentage of work done.
2. All source files. Your implementations should be well commented and easy to read.
3. A report in pdf format containing the following:
   - Short design description of your algorithms.
   - Graph plots and analysis for run-times of the `Broadcast` + `PolyEvaluator` functions vs. the number of processors for a large n. You pick a fixed and large n that will show meaningful execution times (i.e., larger than 1 milliseconds) for all p values you have tested. What observations can you make?

References

Apart from MPI lecture slides, additional helpful resources have being added[3] in T-Square to get you started on working with MPI programs. More resources will be added as needed.

`GettingStartedwithMPIlocally.pdf`	An introductory guide to writing MPI programs
`DebuggingMPIprograms.pdf`	Some helpful hints on how to debug MPI programs
`UsingJinx.pdf`	A simple introduction on the Jinx cluster and how you can test your programs in it

If you are new to programming in MPI, `GettingStartedwithMPIlocally.pdf` is a good place to start. We recommend you code your first MPI hello world program and execute it locally and in the `Jinx` cluster before you start working on the assignment.

[3]We acknowledge the author of these documents Flick, Patrick