CX 4220/CSE 6220 Introduction to High Performance Computing
Spring 2022
Programming Assignment 2
Due Friday, March 18

1 Collective operations

In this assignment, you will implement the broadcast and parallel prefix algorithms discussed in class for hypercubic permutations.

Your broadcast implementation should work for any root. You may assume the number of processes is a power of 2 and follow the definition of broadcast from the slides.

Your parallel prefix implementation should work for any datatype, any operator, and any number of processes. This contrasts with the algorithm from the slides, which assumed integers and addition. Your parallel prefix code will take an input array, input prefix function, and output the computed prefixes.

After implementing the algorithms, you will compare the runtime of your implementations with the runtime of `MPI_Bcast` and `MPI_Scan` for various n and p for an operator of your choice. Plot the runtime comparisons and describe your observations of the results. Provide a brief introduction of your code or details specific to your implementations. Contrast the performance of your implementation with the default implementation

2 Code framework git notes

For this assignment, we are providing a framework in the git repo below.

`https://github.gatech.edu/hpc-sp22/prog2_framework.git`.

DO NOT FORK THIS REPO. DO NOT PUSH PROJECT SOLUTIONS PUBLICLY. THIS IS AN ACADEMIC INTEGRITY VIOLATION.

Github Forks are publicly available. Thus, you must clone the repository rather than fork it. If you are unfamiliar with the process of creating a different upstream, please follow the following instructions:

1. For a new repo, first create a new **PRIVATE** empty repo on github.gatech.edu.

2. Then, clone the framework repo.

```
git clone https://github.gatech.edu/hpc-sp22/prog2_framework.git
```

3. Now change the url of the locally cloned repo to point to your **PRIVATE** empty repo on github.gatech.edu. Our version will be saved as "upstream", while your version will be "origin".

```
git remote rename origin upstream
git remote add origin <your_repo_url>
git push -u origin main
```

4. Give your group members access to this **PRIVATE** repo by adding them as collaborators in the repo settings.

5. Collaborators can add the official framework repo as an upstream repo using this command:

   ```
   git remote add upstream https://github.gatech.edu/hpc-sp22/prog2_framework.git
   ```

   If you run `git remote -v`, then you should see two repos listed—the official public repo and your **PRIVATE** repo.

   ```
   >$ git remote -v
   origin <your_repo_url> (fetch)
   origin <your_repo_url> (push)
   upstream   git@github.gatech.edu:hpc-sp22/prog2_framework.git (fetch)
   upstream   git@github.gatech.edu:hpc-sp22/prog2_framework.git (push)
   ```

If you set up your repo using the previous instructions, you can merge our updates to the framework by following the following steps:

1. Pull changes from main upstream branch. (This will get our updates from our repo).

   ```
   git fetch upstream
   ```

2. Merge the changes in the main upstream branch into your main branch.

   ```
   git checkout main
   git merge upstream/main
   ```

3. This will create a merge commit. If there are merge conflicts, you will need to fix them. Once the merge is committed, you can push the changes to your repo.

   ```
   git push
   ```

3 Code framework details

You will be implementing two functions in `collective.cpp`. You are restricted to using only MPI point-to-point communication, which means you can only use sends and receives (including variants such as `MPI_Isend`, `MPI_Irecv`, `MPI_Sendrecv`, etc.).

1. The function `HPC_Bcast()` should run the broadcast algorithm. This has the same arguments as `MPI_Bcast()`.

2. The function `HPC_Prefix()` should run the parallel prefix algorithm. The arguments to the function are defined in `collective.h`.

You can find more details on the function arguments in `collective.h`.
There are more details about other files in the framework in the README.

4 Submission instructions

We highly recommend working in groups. Groups are allowed to have up to 3 members, and only one member should submit the project for the group. Each group member is expected to understand every part of the code and contribute equally to the project.

We want to ensure the groups are tracked correctly. Therefore, when submitting, all group members must be listed in the following three places:

- in a comment at the top of your group's code file,
- on the first page of your group's report,
- and on Gradescope as part of the group assignment submission. Only one member of your group should submit on Gradescope, and they should add group members using the Gradescope interface. See this link for more information on adding group members on Gradescope: `https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members`

Submit exactly 2 files: `report.pdf` and `collective.cpp`. Do not submit a zip file.

5 Compilation

- In order to be consistent with the autograder on Gradescope, use `module load gcc openmpi` on PACE.
- We will compile the code with `--std=c++17 -O3`, which is in the Makefile of the framework.
- Your code must compile and run correctly as is (with only your submitted modifications to collective.h) on both PACE and the autograder.

6 Grading (25 pts total)

Grading consists of the following components:

1. Gradescope autograder (4 pts for broadcast, 10 pts for parallel prefix)
    - Automated tests on Gradescope will test your broadcast and parallel prefix functions. These tests are included in the git repo so you can run them on your own computer.
2. Broadcast code (3 pts) and parallel prefix code (3 pts)

- We will look over your code to make sure it follows the hypercubic permutation structure of the algorithm in the slides, uses only point-to-point communication, and doesn't just check for which test case is being run in order to output the right answer.

3. Report (5 pts)

   - The report should include your plots and the descriptions of your plots and algorithms, as well as the name of each member of your team.