Homework 1: MPI communications

George Bosilca

September 4, 2020
Create a collective function, that much be called by all processes on the communicator that will move a token from one process into the next, and once it reaches the last process of the communicator, back to the first process (rank 0). The token always starts at process with rank 0. Upon return from this function each process gets two information:

- an integer flag to indicate it now has the token, which means the associated data contains the token information
- the data which contains whatever is now considered as the token.

The prototype of this function is: `tk_next(void* buf, int count, MPI_Datatype type, MPI_Comm comm, int *token).

- the tuple `buf, count and type` indicates either the buffer where the token is located or the buffer where the token must be received.
- `comm` the communicator where the operation will happen.
- `token` will be zero if the token is not currently located on this process and to 1 otherwise (in which case the buffer contains the expected token data).
Finding a specific value

Assume a one dimensional array of integers distributed among all the processes from a communicator. This array is initialized with randomized value in a particular range. You need to write a library call that find all instances of a particular value and distribute the number of instances on each process to all participants in the communicator.

The expected API of the call is `find_value(int val, int* local_array, int count, MPI_Comm comm, int* array_of_counts)` where

- `local_array` has a length of count integers
- `array_of_counts` has a length of number of process in `comm`.

Upon return from this collective function, the `array_of_counts` of all processes must contain the same values indicating how many instances of the value `val` the corresponding index process rank has.
Homework

The homework submission is an automated process, including the submission itself. Upon the homework HWX deadline, a robot will automatically pull your HWX branch, test it, and partially grade it (for correctness and sometimes performance). If these tests are not successful, the grade will be directly affected.

Git

I assume your name is bstudent, and your fork of the homework repo is bstudent/homework. In case you haven’t yet forked my repo https://github.com/COSC462-UTK/homework, do it ASAP. To facilitate your git use, you should setup you access to GitHub via ssh keys, and make sure you have a working ssh agent. This document is not a git cheat-sheet, it focuses on the minimal usage of git for the purpose of the COSC462 homework.
Clone your repo

```bash
git clone git@github.com:bstudent/homework
cd homework
```

Create a branch for HW1

```bash
git checkout -b hw1
git branch
```

Switch to the HW1 branch (if it exists)

```bash
git branch hw1
```
Setting a remote to the homework repo

```bash
git remote add upstream \
  git@github.com:COSC462-UTK/homework.git
```

Getting updates from your repo

```bash
git pull
```

Getting updates from the homework repo

```bash
git pull --rebase upstream master
```
Changes
Pushing changes

Adding new files

```bash
git add *files*
```

Commit local changes

```bash
git commit -s
```

Making the changes visible

```bash
git push
```
Do/Don’t

▶ Read a good git cheat-sheet (or the manual) to learn a little bit more about git
▶ Don’t force push your changes (push -f). It will affect the history of all the copies of your repo.
▶ Don’t create a PR against the homework repo with your homework. Just take care of the correct branch, and everything else will be done automatically.
▶ Do create a PR against the homework repo if you have identified a problem with the skeleton code, and would like to provide a fix.
▶ Make sure all dependencies, outside the mainstream one, are in your repo, accessible to the homework branch.