Lab #8: MPI Point-to-Point

PCSE 2015

1 Send and Receive

1. Start a new file called `ping_1.*` (`c` or `F90`).
2. Inside, write a program that only works with 2 processes. If any more or less processes are used, then abort sending an error message to the user. Show the output if you give your program 3 processes instead of 2.
3. Use `MPI_Send` and `MPI_Recv` to pass a single character from process 0 to process 1 and from process 1 to process 0. Make sure each send/receive tag is unique.
4. You'll need to create two character variables `inchar` and `outchar`. Initialize `outchar`, the outgoing character, to anything you like but leave `inchar` uninitialized.
5. From each process, print the rank and values of `inchar`, and `outchar` before and after the message passing has occurred.
6. From each process, also print out the source ID and tag ID using information from the `MPI_Status` variable used in your receive function calls. See here. For C folks, you'll need use `Stat.MPI_SOURCE` and `Stat.MPI_TAG` assuming you create an MPI_Status struct called `Stat`. Fortran folks, you'll need to use `Stat(MPI_SOURCE) Stat(MPI_TAG)` assuming you create an integer array of length MPI_STATUS_SIZE called `Stat`.
7. Verify that you get the results you expect. Make sure to run your code multiple times to gain confidence that the results are deterministic.

2 Sendrecv

1. Copy Part 1's code to a new file `ping_2.*`. Redo Part 1 using only `MPI_Sendrecv` calls instead.
2. Verify that you obtain the same results.

3 Isend and Ireceive

1. Copy Part 1's code to a new file `ping_3.*`. Redo Part 1 using only `MPI_Isend` and `MPI_Irecv` calls instead.
2. You may ignore printing of the source ID and tag ID using MPI_Status queries. Instead, just print the rank, `inchar`, and `outchar` before and after your non-blocking calls.
3. Show the output you expect. Show the output you don't expect. Why is this version nondeterministic (run it many times)?
4. Try using `MPI_Wait` to fix this.