Lab #9: MPI P2P Quiz + Collectives

PCSE 2015

1 Announcements

1. Would you be interested in building a compute cluster from scratch with cutting edge parts, installing the OS and a bunch of programs, measuring the performance given certain restrictions and objectives, and then competing internationally to see if you've got what it takes to take home the SC15 Student Cluster Competition Title for the fourth year in a row? More information can be found here (URL embedded). Informational meeting next week. Contact Cyrus via email if you would be interested.
2. We'll be using X11 tunneling today. Before you start PuTTY, go to the Windows Start Menu, search for Xming, and start Xming first. Next, follow the directions given here(URL embedded) for configuring PuTTY to enable X11 Forwarding.
3. We'll also be (optionally) using Allinea DDT today. Follow the directions here(URL embedded) to start DDT from a login node. Make sure to compile programs with the "-g" if you would like to profile/debug them with DDT.

2 Bug Finder!

1. I have looked through the quizzes from Tuesday and found the most common bugs. They are now coded and provided in the code/part_1 folder. There are 7 buggy codes – ring_1a, ring_1b, ring_1c, ring_2a, ring_2b, ring_3a, and ring_3b for C and Fortran.
2. Your job is to figure out what's wrong with each one. Some will run but provide incorrect answers. Some will never complete.
3. Try your hand at debugging each code with the help of DDT and/or print statements.
4. In your lab writeup, clearly identify the mistake(s) in each version. Describe the behavior of the buggy code and provide the fix for each one.

3 Collectives #1

Create a new file called `collect_1.*` where * is `c` or `F90` and write a code following the instructions given below:

1. Given the following variable block (C):

   ```
   #define NPROCS  4
   #define ASIZE   8
   #define ASIZEPP (ASIZE / NPROCS)
   #define ROOT    2
   double master_vec[ASIZE], local_vec[ASIZEPP];
   int i, j, rank, numprocs;
   double root_sum, local_sum, global_sum;
   ```

   Or, given the following variable block (Fortran):

   ```
   integer, parameter :: NPROCS  = 4
   integer, parameter :: ASIZE   = 8
   integer, parameter :: ASIZEPP = ASIZE / NPROCS
   integer, parameter :: ROOT    = 2
   double precision  master_vec(ASIZE), local_vec(ASIZEPP)
   integer i, j, rank, numprocs, ierr
   double precision root_sum, local_sum, global_sum
   ```

2. If on the root process, initialize `master_vec` to 100, else initialize to -1.
3. Use `MPI_Scatter` to scatter the contents of `master_vec` from the root process to all other processes and store in the `local_vec` variable.
4. On all processes, add `rank` to each value of `local_vec`.
5. Use the provided code block to output the value of `local_vec` (C):

   ```
     for(i=0; i<numprocs; i++){
       if(rank == i){
         printf("***(Rank␣%i):␣local_vec␣Output:\n",rank);
         for(j = 0; j < ASIZEPP; j++){
           printf("(Rank␣%i):␣%i␣%f\n",rank,j,local_vec[j]);
         }
       }
       MPI_Barrier(MPI_COMM_WORLD);
     }
   ```

   (Fortran):

   ```
     do i=0, numprocs-1
       if(rank == i)then
         write(*,'(a,i0,a)')"***(Rank␣",rank,"):␣local_vec␣Output:"
         do j = 1, ASIZEPP
           write(*,"(2(a,i0),f10.2)")"(Rank␣",rank,"):␣",j,local_vec(j)
         end do
       end if
       call MPI_BARRIER(MPI_COMM_WORLD,ierr)
     end do
   ```

   to verify that each process has successfully received it's piece of `master_vec`.
6. For each process, sum each element of `local_vec` and store the answer in `local_sum`.
7. Use `MPI_Reduce` to sum each process's copy of `local_sum` onto the root process in the `global_sum` variable.

8. Now, use `MPI_Gather` to gather the modified pieces of `local_vec` back into `master_vec` on the same root process that you did the scatter from.
9. For the root process, sum the contents of `master_vec` and store the answer in `root_sum`.
10. For each process, print the elements of `master_vec` and compare `global_sum` to `root_sum` (they should match).
11. Use 4 MPI processes. Your output should look like (C):

```
TACC: Starting up job 5057379
TACC: Setting up parallel environment for MVAPICH2+mpispawn.
TACC: Starting parallel tasks...
***(Rank 0): local_vec Output:
(Rank 0): 0 100.000000
(Rank 0): 1 100.000000
***(Rank 1): local_vec Output:
(Rank 1): 0 101.000000
(Rank 1): 1 101.000000
***(Rank 2): local_vec Output:
(Rank 2): 0 102.000000
(Rank 2): 1 102.000000
***(Rank 3): local_vec Output:
(Rank 3): 0 103.000000
(Rank 3): 1 103.000000
***(Rank 0): master_vec Output:
(Rank 0): 0 -1.000000
(Rank 0): 1 -1.000000
(Rank 0): 2 -1.000000
(Rank 0): 3 -1.000000
(Rank 0): 4 -1.000000
(Rank 0): 5 -1.000000
(Rank 0): 6 -1.000000
(Rank 0): 7 -1.000000
(Rank 0): The ROOT process is: 2
(Rank 0): Root           Sum: 0.000000
(Rank 0): MPI_Reduce     Sum: 0.000000
***(Rank 1): master_vec Output:
(Rank 1): 0 -1.000000
(Rank 1): 1 -1.000000
(Rank 1): 2 -1.000000
(Rank 1): 3 -1.000000
(Rank 1): 4 -1.000000
(Rank 1): 5 -1.000000
(Rank 1): 6 -1.000000
(Rank 1): 7 -1.000000
(Rank 1): The ROOT process is: 2
(Rank 1): Root           Sum: 0.000000
(Rank 1): MPI_Reduce     Sum: 0.000000
***(Rank 2): master_vec Output:
(Rank 2): 0 100.000000
(Rank 2): 1 100.000000
(Rank 2): 2 101.000000
(Rank 2): 3 101.000000
(Rank 2): 4 102.000000
(Rank 2): 5 102.000000
(Rank 2): 6 103.000000
(Rank 2): 7 103.000000
(Rank 2): The ROOT process is: 2
(Rank 2): Root           Sum: 812.000000
(Rank 2): MPI_Reduce     Sum: 812.000000
```

```
***(Rank 3): master_vec Output:
(Rank 3): 0 -1.000000
(Rank 3): 1 -1.000000
(Rank 3): 2 -1.000000
(Rank 3): 3 -1.000000
(Rank 3): 4 -1.000000
(Rank 3): 5 -1.000000
(Rank 3): 6 -1.000000
(Rank 3): 7 -1.000000
(Rank 3): The ROOT process is: 2
(Rank 3): Root            Sum: 0.000000
(Rank 3): MPI_Reduce      Sum: 0.000000

TACC: Shutdown complete. Exiting.
```

(Fortran):

```
TACC: Starting up job 5057379
TACC: Setting up parallel environment for MVAPICH2+mpispawn.
TACC: Starting parallel tasks...
***(Rank 0): local_vec Output:
(Rank 0): 1    100.00
(Rank 0): 2    100.00
***(Rank 1): local_vec Output:
(Rank 1): 1    101.00
(Rank 1): 2    101.00
***(Rank 2): local_vec Output:
(Rank 2): 1    102.00
(Rank 2): 2    102.00
***(Rank 3): local_vec Output:
(Rank 3): 1    103.00
(Rank 3): 2    103.00
***(Rank 0): master_vec Output:
(Rank 0): 1     -1.00
(Rank 0): 2     -1.00
(Rank 0): 3     -1.00
(Rank 0): 4     -1.00
(Rank 0): 5     -1.00
(Rank 0): 6     -1.00
(Rank 0): 7     -1.00
(Rank 0): 8     -1.00
(Rank 0): The ROOT process is: 2
(Rank 0): Root       Sum:      0.00
(Rank 0): MPI_Reduce Sum:      0.00
***(Rank 1): master_vec Output:
(Rank 1): 1     -1.00
(Rank 1): 2     -1.00
(Rank 1): 3     -1.00
(Rank 1): 4     -1.00
(Rank 1): 5     -1.00
(Rank 1): 6     -1.00
(Rank 1): 7     -1.00
(Rank 1): 8     -1.00
(Rank 1): The ROOT process is: 2
(Rank 1): Root       Sum:      0.00
(Rank 1): MPI_Reduce Sum:      0.00
***(Rank 2): master_vec Output:
(Rank 2): 1    100.00
(Rank 2): 2    100.00
(Rank 2): 3    101.00
(Rank 2): 4    101.00
(Rank 2): 5    102.00
```

```
(Rank 2): 6    102.00
(Rank 2): 7    103.00
(Rank 2): 8    103.00
(Rank 2): The ROOT process is: 2
(Rank 2): Root       Sum:    812.00
(Rank 2): MPI_Reduce Sum:    812.00
***(Rank 3): master_vec Output:
(Rank 3): 1     -1.00
(Rank 3): 2     -1.00
(Rank 3): 3     -1.00
(Rank 3): 4     -1.00
(Rank 3): 5     -1.00
(Rank 3): 6     -1.00
(Rank 3): 7     -1.00
(Rank 3): 8     -1.00
(Rank 3): The ROOT process is: 2
(Rank 3): Root       Sum:      0.00
(Rank 3): MPI_Reduce Sum:      0.00

TACC: Shutdown complete. Exiting.
```

12. Try modifying `NPROCS`, `ASIZE`, and `ROOT`. Do you still get the output you expect?

4 Collectives #2

Create a new file called `collect_2.*` where * is `c` or `F90` and write a code following the instructions given below:

1. Given the following MODIFIED variable block (C):

```
#define NPROCS  4
#define ASIZE   8
#define ASIZEPP (ASIZE / NPROCS)
#define ROOT    2
double master_vec[ASIZE], local_vec[ASIZEPP];
int i, j, rank, numprocs;
int sendcount[NPROCS], recvcount[NPROCS], displacement[NPROCS];
double root_sum, local_sum, global_sum;
```

Or, given the following MODIFIED variable block (Fortran):

```
integer, parameter :: NPROCS  = 4
integer, parameter :: ASIZE   = 8
integer, parameter :: ASIZEPP = ASIZE / NPROCS
integer, parameter :: ROOT    = 2
double precision  master_vec(ASIZE), local_vec(ASIZEPP)
integer i, j, rank, numprocs, ierr
integer sendcount(NPROCS),recvcount(NPROCS), displacement(NPROCS)
double precision root_sum, local_sum, global_sum
```

2. This code should be a very similar to the code you created in Part #2. The difference should be in the routine calls outlined below.
3. Change `MPI_Scatter` to `MPI_Scatterv`.
4. Change `MPI_Gather` to `MPI_Gatherv`.
5. Change `MPI_Reduce` to `MPI_Allreduce`.

6. You'll need to properly set `sendcount`, `recvcount`, and `displacement` to get your scatterv and gatherv calls to act like the scatter and gather calls of Part #2.
7. Highlight the differences in your output as a result of your `MPI_Allreduce` call.