ACM40660/PH504 Practical 8

ICHEC

2022/23 Spring

1 Simple collective operations

Write simple codes for the following scenarios

1. Process 0 reads from keyboard a number and then distributes it to all other processes in MPI_COMM_WORLD. (use broadcast)

2. Each process generates a random number between 97 and 122, transforms it in a character (use achar in fortran) and then gather all of the resulting characters in process 0. Print the result then. What happens if you gather everything in process 1. Is the result the same? (use mpi gather).

3. You have a real vector A of size m which is distributed over n processes. Each process contains a chunk of the vector s_i properly initialised and nothing else (remember the distributed memory from the previous assignment). Each process computes his local sum. Process 0 gets to compute the total sum and prints it. (use mpi reduce with the proper operation)

2 Find the Deadlock

The find_the_deadlock.c (find_the_deadlock.f90) code in Section 3 contains a deadlock. Find and describe the deadlock and try to correct it. The expected behavior is:

- processor 0 reads from a file (cut&paste the file values.dat) 25 random integers;
- processor 0 broadcasts these integers to all the other processors;
- every processor saves in a separate files these values.

NOTE: There is a bug but also some errors in the program. Solving the deadlock does not mean you have reproduced the expected behavior. Take a look at all the program to see if everything is consistent and matches the specs. The expected wall-time of the corrected program is less than 30 second.

3 Source Code

File: *values.dat*:

```
13
23
21
19
20
15
25
3
11
7
8
14
12
5
10
18
2
17
6
16
22
24
9
1
4
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
FILE *fr;

int main(int argc, char **argv){
   int i, ierr, rank, nprocs;
   long elapsed_seconds;
   int indata[25];
   char buf[256];

   ierr = MPI_Init(&argc,&argv);
   ierr = MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   ierr = MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

   if (rank == 0 ) {
    /* open the file for reading */
      fr = fopen ("values.dat", "rt");

      /* fill indata */
      i = 0;
      while (!feof(fr)) {
        fscanf(fr, "%d", &indata[i]);
        i = i + 1;
     }
      /* close the file */
     fclose(fr);

     ierr = MPI_Bcast( indata, 25, MPI_DOUBLE, 1, MPI_COMM_WORLD);
   }
     // output process rank 0 in file output.0
     // output process rank 1 in file output.1
     // output process rank 2 in file output.2
     // and so on ...
     snprintf(buf, sizeof(buf), "output.%d", rank);


     i = 0;
     while ( i<25 ) {
       fprintf(fr, "%d\n", indata[i]);
       i = i + 1;
     }

   ierr = MPI_Finalize();
   return 0;
}
```

```
program find_the_deadlock
   implicit none
   include 'mpif.h'
   integer :: rank, nprocs,ierror,fileunit
   integer :: indata(25)
   character(len=10) :: filenam

   call MPI_Init(ierror)
   call MPI_Comm_Size(MPI_COMM_WORLD,nprocs,ierror)
   call MPI_Comm_Rank(MPI_COMM_WORLD,rank,ierror)

   if (rank.eq.0) then
     open(1,FILE='values.dat')
     read(1,*,end=10) indata
     call MPI_Bcast(indata,25,MPI_DOUBLE_PRECISION,1,MPI_COMM_WORLD,ierror)
     close(1)
10   continue
   endif

   write(filenam,'(a,i0)') 'output.',rank

   ! output process rank 0 in file output.0
   ! output process rank 0 in file output.1
   ! output process rank 0 in file output.2
   ! and so on
   fileunit = rank+100
   write(fileunit,*) indata


  call MPI_Finalize(ierror)
end program find_the_deadlock
```