Practical4

ICHEC

April 26, 2020

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 Derived datatypes

Create derived datatypes which help one pass around a matrix column, matrix row and a sub matrix. The code will contain:

1. Use dynamic allocation for your original matrix. In C be careful how you allocate the matrix. You need a contiguous block of memory for it. If you have trouble have a look in dgemm.c for a way achieve this.

2. In order to create a new type for each of the three situations. (use MPI_Type_vector)

3. Write small parts of code that demonstrate the usage of the new types. For brevity you can use only two proccesses.

3 Cartesian topology

Consider a n by m matrix with integer elements. Write a code that does the following:

1. the matrix is split in $p \times q$ blocks with each block distributed on a process, see fig. 1. (p·q=number of processes.)

2. create a Cartesian topology that matches the above block distribution. Use periodicity.

3. initialise he matrix with random values of 0 and 1. You can use ran2.c/ran2.F90 from Assignment 02 to generate random numbers. Be sure that running the code on 1 process or on more, with the same starting seed, results in the same random distribution for your matrix. **Hint:** let process 0 generate the data for each sub-matrix and then send it to the right process. An alternative is to put the random number generator in the proper state in each process.
4. for each process find its Cartesian neighbours.
5. add a function that computes for the sum of all elements in the local block.
6. reduce this in the process 0.

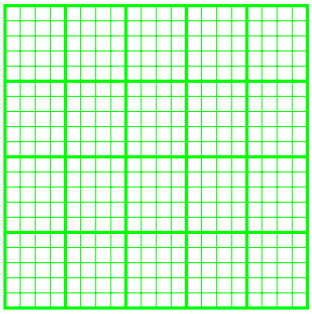

Figure 1. A matrix and its sub-blocks. Each sub-block shall be associated to a process from the Cartesian topology.