Université Grenoble Alpes, Grenoble INP, UFR IM^{2}AG

Master 1 Informatique and Master 1 MOSIG

UE Parallel Algorithms and Programming

Lab # 4

2019

Exercice 1: Broadcast

Write a program in which the process with rank 0 sends an array of 10 integers to all other processes.

Exercice 2: Total sum

Write a program which computes the sum of all processes ID and sends the result to all of them. Write two versions: one with a reduction followed by a broadcast and another one using a global reduction.

Exercice 3: Barrier

Write a program in which processes synchronize using a barrier. To better understand the semantic of a barrier, use the `sleep` function to simulate different execution time before calling the barrier, and observe the total execution time of the application.

Exercice 4: Cartesian virtual topologies

A virtual topology describes a mapping/ordering of MPI processes into a geometric *shape*. The are two main types of topologies supported by MPI: Cartesian (grid) and Graph. These topologies are built upon MPI communicators and groups.

Note: You will be able to reuse some of the code written in this lab for the graded lab on MPI. It can be good to encapsulate the solution of each question in a separate function to be able to easily reuse the code later.

(1) Write a program in which you create a virtual grid of processes (use `MPI_Dims_create` to define the dimensions of the grid and `MPI_Cart_create` to create the grid). Print the new and the previous rank of each process and their coordinates in the grid (use `MPI_Cart_coords`). Try different numbers of processes. Figure 1 presents an example of cartesian topology[1].

[1] source: `https://computing.llnl.gov/tutorials/mpi/#Virtual_Topologies`

0 (0,0)	1 (0,1)	2 (0,2)	3 (0,3)
4 (1,0)	5 (1,1)	6 (1,2)	7 (1,3)
8 (2,0)	9 (2,1)	10 (2,2)	11 (2,3)
12 (3,0)	13 (3,1)	14 (3,2)	15 (3,3)

Figure 1: A cartesian topology

(2) Modify your program to create a virtual $q \times q$ grid of processes. Be sure to create a square grid even if the number of MPI ranks is not a square number.

(3) Create a communicator per row and per column with the function `MPI_Cart_sub` to group all processes in the same row or in the same column in one communicator.

(4) Modify your program to pass a token from rank to rank in each row and each column. Initially rank 0 in each communicator owns the token.

(5) Modify your program so that each process broadcasts an integer to other processes on the same row.

(6) Make global sums on columns.

(7) Make shifts on rows and columns and print the neighbours of each process in the grid (use `MPI_Cart_shift`). What is happening if a process doesn't have its four neighbours in the grid?

(8) Scatter vectors on the grid of processes (use `MPI_Scatter` and `MPI_Gather`).