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

Master 1 Informatique and Master 1 MOSIG

UE Parallel Algorithms and Programming

Lab # 4

F. Desprez (Inria), J-F. Méhaut (UGA/Polytech), T. Ropars (UGA/IM^{2}AG), E. Saillard (Inria)

March 28th, 2017

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. Use loops or `sleep` functions to obtain different execution time among processes.

Exercice 4: Cartesian virtual topologies

In MPI terms, 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. In the following, we are going to focus on grids.

(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 with different numbers of processes to understand the behavior of `MPI_Dims_create()`.

(2) Modify your program to create a virtual $q \times q$ grid of processes. Be sure to create a square grid out of the application processes (not necessarily a square number).

(3) Create communicators for rows and columns with the function `MPI_Cart_sub`.

(4) Modify your program so that a token is passed among processes (one on rows and one on columns). At the beginning, process 0 owns the token. Then it is passed from process to process on the different columns

and rows.

(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`).

Bonus

You will find on Moodle a description of the *Froggy* cluster, as well as a short guide that explains how to use it. Try running your MPI on *Froggy*.