|epcc|

Advanced Parallel Programming Exercises
MPI Neighbourhood Collectives on ARCHER

Dan Holmes

21st March 2015

1 Introduction

The purpose of this exercise is to investigate the neigbourhood collective functions defined in MPI 3.0, specifically comparing the readability relative to point-to-point MPI operations and measuring the relative communication performance for a typical halo-exchange code. You are given example code (taken from the MPI 3.0 Standard), which demonstrates a 2-D halo-exchange communication pattern using a 2-D cartesian topology and the `MPI_Neighborhood_Alltoallw` function. The sample code uses a vector data-type, created with `MPI_Type_vector`, to describe each of the four non-contiguous halo regions.

2 Compiling and Running

The sample code is contained in `APP-Neighbours-Code.tar.gz` on the APP web pages.

On ARCHER, you should be able to compile it without code changes. It can be executed but it will not do anything useful yet, other than prove that the MPI library installed on ARCHER has support for the necessary MPI functions.

Read the sample code, especially `main.c`, and make sure that you understand what it is doing before proceeding.

3 Experiments

Take your existing image processing code (e.g. from the MPP course) and modify it to use a single neighbourhood collective call per loop iteration instead of multiple point-to-point communication calls. Make sure that the setup code for the neighbourhood collective is only executed once before the loop begins and that the clean-up code is executed only once after the loop has finished. The comments in the sample code should help to identify which pieces of sample code need to go where in your code. Make sure that you modify the sample code to fit your code, in particular your array size will likely be different so the data-type definitions will need to be changed as will the send and receive displacements.

Compile your new version of code and verify that it executes correctly (and still gives the correct answer). Measure the performance of your new version of code and compare it with the performance of the original version.

Is your new code faster? Is it easier to read?

4 Further Experiments

Modify the new version of your code so that it uses the non-blocking neighbourhood collective function and compare the performance of this version with the blocking neighbourhood collective version as well as with the blocking and non-blocking point-to-point versions.