

Parallel Programming: Worksheet 4

Exercise 1: Creating an MPI type for a class

When carrying out Lagrangian simulations it is useful to have a class that stores the position of a point/particle (as well as all its other properties). Create a class to store a 2D position and velocity. Choose a maximum vertical and horizontal extent for your domain and then create 10 000 randomly located particles within these extents on processor zero. Divide the domain up into vertical stripes of equal width so that there can be a domain assigned to each of the processes.

Create an MPI type that can send all the information in an object of the class' type together. Send the particles individually from the root to the appropriate processor according to its horizontal position. This can be done using either blocking or non-blocking sends and receives. You can send an empty communication to indicate that all the particles have been sent.

Exercise 2: Sending rows and columns of a matrix

Create the same size 2D matrix on each process. On each process also create 4 MPI_Datatypes one each for the top, bottom, left and right hand boundaries of the 2D matrix. Put data into the matrix on process zero and transfer the edges of this data to all the other processes.

Exercise 3: Creating a temporary MPI type

In the exercise 1 doing a large number of communications to send all the particles is very inefficient. Modify the code to do the transfer as a single communication for each process. On the zero process you need to create temporary variables for the transfer of the data to each of the processes. This is because the data for each processes will be randomly scattered within the list of data created on processor zero. On the other processes you will need to use a probe to determine how many particles they are to receive. You will not need to create temporary types on these processes as you can store them as continuous memory and you can use a blocking receive as you are only receiving a single piece of information.

Exercise 4: File writing with MPI

Create some data on each process (you could use your GCD program). Instead of sending this data back to process zero and displaying it to screen, write the data to file. Have one version of the code where this is written to individual appropriately

Workshop Exercise 4

Build on the program created in class. Give each of the particles a velocity and integrate the position forward in time. Allow the particles to bounce off the edges of the domain. Include gravity in the calculation of the velocity and position. When a particle passes outside the region their current processor they should be passed on to the appropriate process.

The cycle should consist of one step of time integration followed by the transfer of particles where appropriate. You can use an `MPI_Alltoall` to tell each processor how many particles to receive from each other processor. Once this has been done the actual particles can be communicated using a temporary MPI data type so that they can all be sent at the same time.

As the particles will need to be cut out of the middle of the lists it may be efficient to use a linked list for the storage of the particles rather than a dynamic array, though other methods can also be used.

Get each process to write out its data to a separate file at appropriate intervals. Ensure that the file names are numbered in such a manner that their output is identifiable.

Write a post-processing script that will read in the data from these files and output the results to a single appropriately structured file. You can do this post-processing code using either C++ or Python.