**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 together with any other position that you wish. Choose a maximum vertical and horizontal extent for your domain and then create 10 000 randomly located particles within these extents on processor zero. Decide how to divide up the domain between the processes and then send the appropriate particles from processor zero 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. 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: Creating a temporary MPI type**

In the previous exercise doing a large number of communications to send all the particles is very inefficient. Modify the above 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 then as continuous memory.

**Exercise 3: Exploratory decomposition – Optional hard problem**

Implement the solution of the 15 puzzle problem in parallel. With only 4 processes the implementation is relatively straight forward as each of the processes can be responsible for one of the moves from the initial state, with all the subsequent moves carried out on those processes. With more available nodes the decomposition is more complex as each of the processes need to send new moves on to available processes while keeping some of the solution for itself. At each step there needs to be a communication between the nodes to determine if a solution has been found. If one is found the chain of moves required needs to be sent back to the root.

**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, though other methods can also be used.