Parallel programming exercises

Parallelising operations

Suggest strategies to parallelise the following operations. If you can think of more than one way, discuss pro's and con's. Do you expect the parallel result to be identical to the serial result? Why/why not?

1. Multiply all elements of a large array by (i) 4, or (ii) by a value prompted from the user
2. Add two very large arrays (a,b) to find c=a+b.
3. Compute the sum of all variables in a (large) array.
4. Given a set of particles (with mass and position)
   (a) compute the position of the centre of mass
   (b) shift set of particles to centre of mass system
   (c) compute density on a rectangular grid using NGP[1] interpolation
   (d) find all particles at a given distance from a given location
   (e) write all particles to disk
5. Smooth a 2D image with a top-hat.

[1] nearest grid point

Example codes

1. The basics: write an mpi programme in which each process writes its name, the total number of processes and 'hello world', to standard I/O. Adapt it so processes write *in order*.

2. Write a Makefile for this programme

3. Adapt your programme such that

   (a) process 0 reads a real value, prompted from the user

   (b) .... and broadcasts it to all.

4. Adapt the above programme to sum all values and write the result to standard I/O. Adapt it such that all processes know the total sum at the end of the operation.

5. Adapt your programme such that an array of values is broadcast, and summed

6. Adapt this programme to send integers, instead of reals.

7. Create a *structure* (called particle), that has 3 reals, and one integer (representing for example, the 3 coordinates of a particle, and an identifier). Create an array of structures. Create 100 particles on process 0, and send them to process 1.

8. Adapt the above programme, so that processes 0 and 1 exchange their particles.

9. Adapt it such that the number of particles sent and received need not be the same.

10. Adapt your program so that it performs these operation using (i) blocking, and (ii) non-blocking sends and receives.