PHYS 5210: Computational Methods in Astrophysics, Assignment 3

Set Nov 3rd due Nov 26th

Notes: For the questions in which you are expected to write a code, please email me a copy.

Q1. Suppose we have algorithm that uses a 1024^3 grid of 8-byte words for a particular calculation. For each grid cell 112 calculations are required. We can run the calculation on a distributed memory system that has sufficient memory on each node to store the entire calculation, so that 1 CPU benchmark can be obtained. The network connecting the processors has a bandwidth of 80 MB/s, and the latency is proportional to the inverse bandwidth. Each CPU on the system is capable of calculating this particular algorithm at 600 Mflops.

(a) Perform a scaling analysis of 1-d, 2-d and 3-d virtual topologies for the domain decomposition. Derive formulae for the communication times similar to those shown in lecture 8 for the 2-d problem. Plot these formulae for 32 to 1024 processors (you may do a continuous plot even if a certain number of processors is disallowed by a given domain decomposition).

(b) Add in the total execution time for the algorithm itself (although you must include how this scales with the number of processors as well) and calculate overall scaling figures. Graph the total execution time for 32 to 1024 processors. Graph the overall speed-up as well. How much of an advantage is there for the 3-d decomposition in this case?

Q2. Distributed termination can be detected via a "ring termination algorithm. In this case all processes are arranged in a virtual ring structure P_1, ..., P_n, where periodic wrap-around means that P_n passes a message to P_1 (and vice versa). When P1 completes its allotment of work it forwards a black token to its neighbour. The neighbour process will only forward the black token when it too has completed its work, and so on. The token forwarding process continues until the black token is passed back to the first process to end its work, which was also where the termination token originated. This process then sends a termination signal to all processes. Implement this ring termination algorithm in MPI (use a simple dummy routine for the work on a node, further there need not be communication between nodes in the work). Email me your code as well as providing a printed copy.

Q3. Interpolation of particles on to grids is common problem in both graphics (splatting) and HPC. Suppose we have a vector of positions r(3,N) and a 3-dimensional grid of size L^3 (in the question L=64), where the spatial positions of the particles are commensurate with the grid. For each particle we wish to assign weights to the nearest neighbours on the grid according to the following equation:

$$W(dr) = W(dx, dy, dz) = \begin{cases} (1-|dx|)(1-|dy|)(1-|dz|), & \text{if } |dx| \leq 1, |dy| \leq 1, |dz| \leq 1, \\ 0, & \text{otherwise} \end{cases}$$

where dx, dy, dz are the distances from the particle to the grid point being considered. If all the distances to the nearest 8 grid points are different then clearly 8 different weights will be necessary. Further, assume that the grid is triply periodic so that weights that would normally extend beyond the grid wrap around and appear on the other side of the grid (for example, if the grid is on the interval $[1:65)$ the 65th position on a grid 64 across would wrap around to 1).

(a) Write and optimize (avoid using `if`, for example) an algorithm to perform this operation for the list of 64^3 particles given on the class website on a grid of size 64^3. Note that the particle positions are defined on [0:1] and you should scale them up to be on the same interval as the grid cells.
(b) (HARDER) Parallelize this operation using MPI assuming a power of 2 number of cpus for simplicity. Read the data into the zero rank node and pass out sections of the particles to the remote nodes - you should pass particles that are confined to a specific region in space, passing all of them is not allowed (this would use excessive amounts of memory in a more realistic situation). Each of the remote nodes should store a section of the grid which the particles will splat on to, and you may use a 1-d domain decomposition for simplicity. You must then map the sections of the grid on the remote nodes back to the entire grid on the zero rank node.
Overall you will need code that does the following:
Read in data
Figure out for a given number of processors how the grid is divided up
Figure out which particles need to be sent to which processor
Send particle data to appropriate processor
Perform splatting on grid section
Send pieces of arrays back to zeroth node

Be careful to ensure that you add up the values in the cells along the boundaries of the domain decomposition. Email me your code as well as providing a printed copy.