SCIF30001: Assessed Exercise 2

Laura Ratcliff

Weeks 13-14

1 Overview

The aim of this exercise is to write C++ code for calculating the number of nearest neighbours within a given cut-off radius, r_c, for each atom in a list of atoms. Note, you do not need to determine *which* atoms are neighbours of a given atom, only *how many* neighbours each atom has in total. The code should also output the average, maximum and minimum number of neighbours across a given system. You should use MPI to parallelise the code, and test it on a single node of BlueCrystal. Only 1 MPI task (the root task) needs to know and report the final simulation outcome.

You will be given the atomic coordinates for three different system sizes. The positions are in three dimensions; they are in free boundary conditions (i.e. no periodic interactions), but are contained within a cube of length L. You will find further information about how to approach this problem, including suggestions for the MPI approach, in the week 13 video on Finding Neighbours.

In the first instance, you should prioritise correctness, however higher marks will be awarded for more efficient implementations, e.g. cell lists over a brute force approach, more efficient load balancing etc. You are also encouraged to try more than one approach, e.g. if you have implemented both cell list and brute force approaches, you should give the code and timings for both. Hint: for the cell list approach, don't forget to also check for neighbours within a cell, not just with adjacent cells.

2 Inputs

You will be given a set of inputs containing atomic coordinates for three systems of different sizes:

- 120 atoms, $L = 18$ Å
- 10549 atoms, $L = 100$ Å
- 147023 atoms, $L = 200$ Å

Each set of coordinates is given in an .xyz file, named for the number of atoms it contains, and with the length L also given in the file.

You will also be given a skeleton code which includes a function for reading in the atomic coordinates – the atomic species (Ar) is not relevant to this task, and is therefore ignored. This function returns the number of atoms, the side of the cube, L, and a vector of vectors, `positions`, where `positions[i][0]` contains the x-coordinate of atom i, `positions[i][1]` contains the y-coordinate of atom i, and `positions[i][2]` contains the z-coordinate of atom i.

3 Submission Instructions

You should submit two documents:

- a single source file containing your code – if you have multiple versions (e.g. brute force and cell list approach) these should be implemented in different functions which can be called from the same main function
- a short report (≤ 2 pages) containing:
  - a brief description of your code
  - the average, maximum and minimum number of neighbours for each of the provided inputs, for $r_c = 8$ Å
  - for each of the provided inputs, and each approach you have implemented:
    - the time taken to run the code on 1 MPI task
    - the fastest time to solution when running on 1 node of BlueCrystal, and the number of tasks used to achieve it, e.g. if the code is quickest when run with 4 MPI tasks, you should state this and give the time taken for 4 MPI tasks
    - in order to save time running the code, if for a given calculation the time taken exceeds 10 minutes, it is enough to give the time taken as simply "$>$ 10 minutes"

The deadline for submission is **midday on Thursday of week 14**.