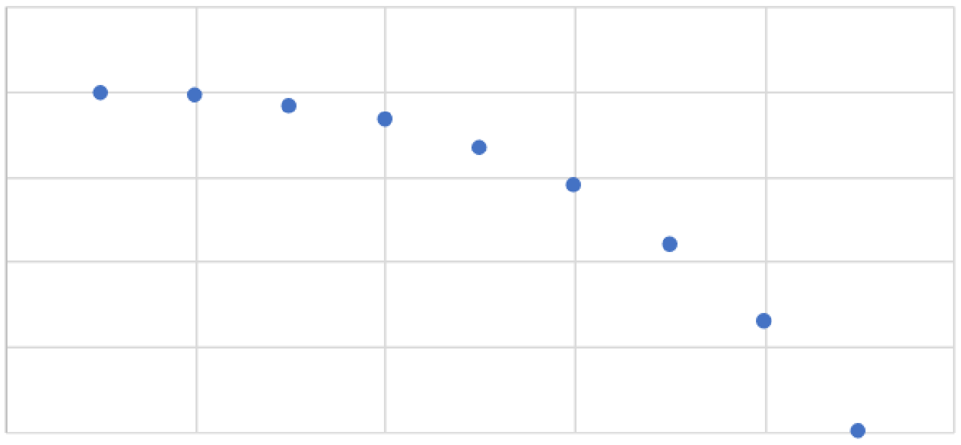
Parallel Scientific Computing I
Kieran Lankester

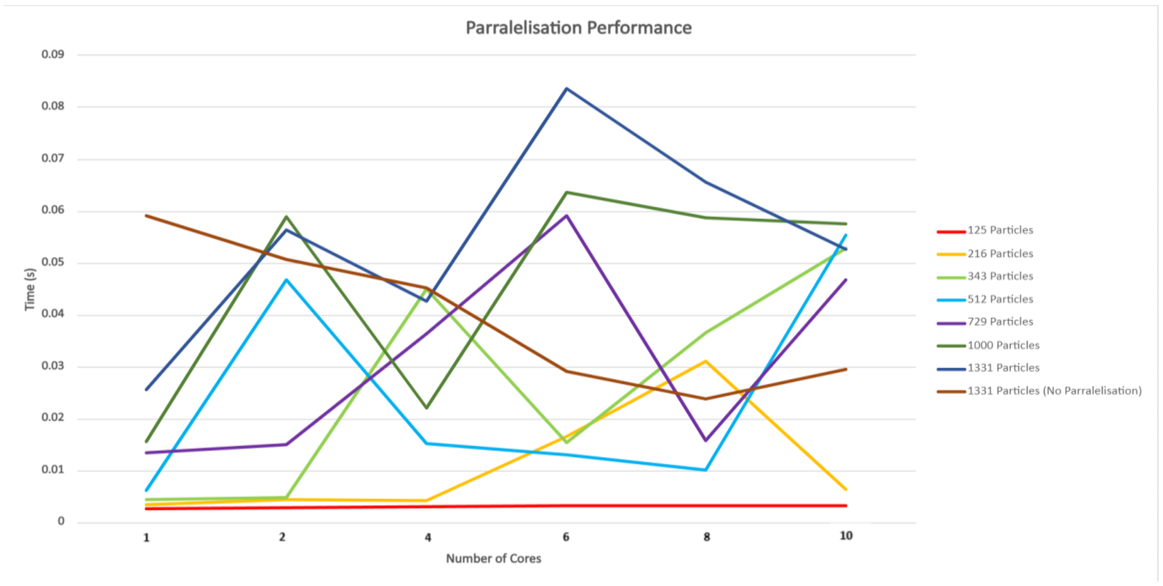
N-body simulator has complexity $O(n^2)$. Two nested loops were used, one to calculate the forces acting on each particle, and one to determine if any two particles had collided. There were also single loops used to initialise the force data structure, update the particle positions and velocities, and update the maximum velocity. The force calculations were made more efficient by taking advantage of the principle that each pair of particles exhibit equal and opposite forces.
An improvement could be to implement a prior check to see if any particles need to be merged. This prior check can be done by using a variable that keeps track of the maximum mass across all particles, along with the minimum distance variable. In some timesteps, this would remove having to iterate through each pair of particles to check for collisions.

To test convergence behaviour, a two-particle setup was used. The particles were set up so that initially one particle was spiralling around the other. Both particles had equal mass. The distance between the particles was recorded up until the point at which they collided. It was observed that the pattern represented a first order convergence between the particles, as was expected due to the similar masses and force calculation used. The pattern of this convergence can be seen below.



A linked cell data structure was used for step 2. Each cell was defined as a struct which included the lower x, y, and z coordinates, and a vector contained the index of each of the neighbouring cells. The size of the data structure was chosen to be $3 \times 3 \times 3$, although this value can be changed by adjusting the *CellDim* variable. The length of the sides of each cell were calculated by finding the max range between any two sets of coordinates, and this side length was applied to find the lower coordinate values for each cell. Particles were then arranged into their respective cells based on their position in space. For the force calculations, the cut off radius was chosen to be the same as the side length of the cells. Furthermore, it was decided to ignore particles that moved outside of the cell grid.
If the particles were moving very quickly, an alternate data structure would be preferable as there is a higher chance that the particles will move outside the boundary of the pre-defined cells. However, if the bodies were only vibrating, then the linked cell method used would still be efficient as this would have the opposite effect.

The simulations were executed using a 4900Mhz, 10 core Intel i9 10900K CPU. The operating system used was Ubuntu 20.02.3. All the code was compiled using the GNU Compiler.
A study was done to investigate the scalability of a parallelised version of the implementation. Multiple tests were undertaken involving varying numbers of particles and cores. For each test combination, the total runtime was obtained from five separate simulations and the average of these was recorded. The results of this study can be seen in the graph below.



The results obtained show varying performance in most cases, but it cannot be observed that the parallelisation techniques used increased performance. Surprisingly, when using 1331 particles, the non-parallelised implementation produced better results. Although it cannot be seen from these results, performance could improve if the simulation were tested on an increased number of cores. Alternatively, an increased number of timesteps may show different results.