Molecular Dynamics with C++ report

Eslam Salah Zaki Elshiekh

5051183

University of Freiburg

Faculty of Engineering

Department of Microsystems Engineering

September 30th, 2022

Contents

1 Introduction 1

2 Methods 3

2.1 Velocity-Verlet integrator 3

2.2 Lennard-Jones potential force 4

2.3 Thermostat 4

2.4 Neighbor list search 5

2.5 Embedded-atom method potential force 6

2.6 Parallelization 7

3 Implementation 9

3.1 Velocity-Verlet integrator 9

3.1.1 Test strategy for Verlet integrator 9

3.2 Lennard-Jones potential force 10

3.3 Berendsen thermostat 11

3.3.1 Test strategy for Berendsen thermostat 11

3.4 Neighbor List 13

3.5 Embedded atom method 13

3.6 Units and specification of the time unit 14

3.6.1 time step for the gold potential 15

3.7 Parallelization using MPI 15

4 Results **17**

4.1 Total energy as a function of time for different time steps 17

4.2 Snapshots sequence of LJ simulation 18

4.3 simulation time as a function of the size (number of atoms) without neighbor list 21

4.4 simulation time as a function of the size (number of atoms) with neighbor list 22

4.5 total energy vs temperature 23

4.6 melting point versus cluster size 25

4.7 heat capacity versus cluster size 25

4.8 latent heat versus cluster size 27

4.9 Energy conservation with MPI parallelization 27

4.10 Nanowire defects 27

5 Conclusion **29**

References **31**

1 Introduction

Molecular dynamics simulations are becoming an essential part of modern technology, because they allow to study the behavior of complex systems in a controlled way. The simulation of molecular systems is a very complex task, because the number of atoms in a system can be very large. Therefore, the simulation of a system requires a lot of computational power. In the last years, the computational power of computers has increased dramatically. This development has led to the fact that molecular dynamics simulations are now possible for systems with millions of atoms. However, the simulation of such systems is still a very demanding task. Therefore, the simulation of such systems is usually done on supercomputers or clusters of computers.

In this report, we will discuss the design and Implementation of a molecular dynamics simulations for atoms and molecules. We will also discuss the different potential forces that are used in the simulations, and how they are implemented, initializing atomic systems in random positions and preserve the atoms from evaporating using Thermostates, how to make the simulation run faster by using only neighbor list search, and how to parallelize the simulation using MPI.

2 Methods

Here we will discuss the different methods that we will use in our simulation, then in the next chapter we will discuss the implementation of these methods.

2.1 Velocity-Verlet integrator

The velocity-Verlet algorithm is a numerical method based on the Taylor expansion of the position and velocity of the particles for solving the equations of motion of a system of particles. It is a symplectic integrator, which means that it conserves the total energy of the system that's because the velocity is calculated at the middle of the time step, and the position is calculated at the end of the time step. This means that the velocity is calculated at the same time as the acceleration, and the position is calculated at the same time as the velocity. This means that the total energy of the system is conserved. A disadvantage of the velocity-Verlet algorithm is that it is not very accurate. The error in the position is proportional to Δt^2, and the error in the velocity is proportional to Δt. This means that the velocity-Verlet algorithm is not very accurate for large time steps.

2.2 Lennard-Jones potential force

The Lennard-Jones potential is a potential that is used in molecular dynamics simulations to model the interatomic interactions between atoms and molecules. It models both the attractive and repulsive forces between atoms and molecules in a system in a simple way. It is a simple model because it only depends on the distance between the atoms and molecules. It is not a very accurate model though because it does not take into account the charge of the particles, and the different types of interactions between the particles. This potential is very popular to use in molecular dynamics simulations because it is simple to use, and it is computationally cheap. It can also be used to model different types of interactions between atoms and molecules, like the interaction between the electrons of an atom with the electrons of another atom, the interaction between the electrons of an atom with the nuclei of another atom, and the interaction between the nuclei of an atom with the nuclei of another atom.

The Lennard-Jones potential is given by:

$$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \tag{1}$$

where ϵ is the depth of the potential well, and σ is the distance at which the potential is at a minimum. The force is given by the following equation:

$$F(r) = -\frac{dV(r)}{dr} = 24\epsilon \left[\frac{2}{r} \left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \tag{2}$$

2.3 Thermostat

The thermostat is an algorithm that is used to preserve the atoms from evaporating. The idea is when we initialize the atoms in random positions, the atoms will try to get to the equilibrium position. This means that the atoms will move very fast and will collide with each other and the temperature of the system will increase very fast. This will lead to the atoms evaporation. The thermostat is used to prevent this from happening. The thermostat is used to slow down the atoms and make them move slower until they reach the equilibrium position and the temperature of the system is stable.

The Berendsen thremostate is the thermostat that we will use in our simulation. It's one of the most popular thermostats. It is a simple thermostat that is easy to implement. To maintain the temperature, the system is coupled to an external heat bath with fixed Temperature T_0[1]. The velocities of the particles are rescaled with a factor λ at each time step:

$$\lambda = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_0}{T} - 1\right)} \tag{3}$$

where τ is the relaxation time of the thermostat, and T is the temperature of the system. The relaxation time is the time it takes for the system to reach the equilibrium temperature. The relaxation time is an important parameter in the thermostat. If the relaxation time is too small, there will be a strong coupling between the system and the heat bath, this means that the system will reach the equilibrium temperature very fast. And if the relaxation time is too big, there will be a weak coupling between the system and the heat bath, this means that the system will take a long time to reach the equilibrium temperature. And in any case it needs to be bigger than the time step, otherwise the system will not reach the equilibrium temperature.

2.4 Neighbor list search

The problem of using potential forces in molecular dynamics simulations is that the force is calculated between all pairs of atoms. This means that the computational cost of the simulation is proportional to N^2, where N is the number of atoms in the system. This is a very big computational cost, and it is not very efficient. To solve this problem, we can use a neighbor list search. It's an algorithm that is used to speed up the calculation of the forces that act on the particles. It is based on the idea of only calculating the force between atoms that are close to each other. This means that we will only calculate the force between atoms that are in the same cell, and the neighboring cells. Therefor the computational cost of the simulation becomes proportional to N instead of N^2. And this is a very big improvement in the computational cost of the simulation.

2.5 Embedded-atom method potential force

The embedded-atom method (EAM) potentials are methods for calculating the forces between atoms. EAM can describe many different scenarios from the crystalline structure of unary metals or alloy, to amorphous alloys (metallic glasses) and melts. The EAM is based on the assumption that the energy of an impurity in a host crystal lattice is a functional of the overall electron density of the system (that leads to an attraction), plus some form of repulsion (i.e. due to Pauli exclusion). This can be written as:

$$E_{pot} = F[\rho(r)] + \phi \tag{4}$$

where $F[\rho(r)]$ is the embedding function, and ϕ is the electron electron repulsion function. The embedding function is a function of the electron density of the system, and it is used to describe the interaction between the impurity and the host crystal lattice. The electron electron repulsion function is a function of the distance between two electrons, and it is used to describe the repulsion between the electrons. And when we consider each individual atom in the system as an impurity in the host crystal lattice, then the embedding function is a function of the local electron density around the atom. The embedding function is given by the following equation:

$$F[\rho(r)] = \sum_i F(\rho_i) + 0.5 \sum_{i,j} \phi(r_{ij}) \tag{5}$$

And the local electron density around the atom is given by the following equation:

$$\rho_i = \sum_j f(r_{ij}) \tag{6}$$

In this simulation, we will use the Gupta potential [2] as the embedding function, and use the default parameters of Cleri & Rosato [3] for gold.

2.6 Parallelization

To scale and improve the performance of the simulation, we will use parallelization. The parallelization is used to divide the simulation into multiple parts (domain decomposition), and each part is calculated by a different processor. This means that the simulation will be calculated faster, and we can simulate even bigger systems for a longer time. The parallelization is using MPI (Message Passing Interface) to communicate between the processors. MPI is a standard for message passing between processors and for the communication between the processors. It is used to send and receive messages between the processors in the simulation. Periodic boundary conditions are used in the simulation. This means that the simulation is a periodic system, and the atoms that are at the edge of the simulation will appear at the other edge of the simulation. This means that the simulation is a closed system, and the atoms will not escape the simulation.

3 Implementation

3.1 Velocity-Verlet integrator

The velocity-Verlet algorithm consists of two steps: the first step is to calculate the new positions of the atoms, and the intermediate velocities, and the second step is to calculate the new velocities of the atoms. The first step is done by using the following equation:

$$\begin{aligned} \mathbf{v}_{i+1} &= \mathbf{v}_i + 0.5 F_i \Delta t / m_i \\ \mathbf{r}_{i+1} &= \mathbf{r}_i + \mathbf{v}_i \Delta t \end{aligned} \tag{7}$$

And the second step is done by using the following equation:

$$\mathbf{v}_{i+1} = \mathbf{v}_i + 0.5 F_i \Delta t / m_i \tag{8}$$

where F_i is the force on atom i, Δt is the time step, m_i is the mass of atom i, $\mathbf{v}_i$ is the velocity of atom i and $\mathbf{r}_i$ is the position of atom i.

3.1.1 Test strategy for Verlet integrator

We test the Verlet integrator by comparing the results of the Verlet integrator with the results of the analytical solution of the equations of motion of a particle. the analytical solution of the equations of motion of a particle is given by the following equations:

$$\begin{aligned} x_i(t+dt) &= x_i(t) + v_i(t)dt + \frac{1}{2m} f_i(t)dt^2 \\ v_i(t+dt) &= v_i(t) + \frac{1}{2m_i}(f_i(t) + f_i(t+dt))dt \end{aligned} \tag{9}$$

If we assume that the there is no acting forces (constant and equal to zero) on the particles, then the analytical solution of the equations of motion of a particle after N time steps is given by the following equations:

$$\begin{aligned} x_i(t + N * dt) &= x_i(t) + \sum_{i=0}^{N} v_i(t)dt \\ v_i(t + N * dt) &= v_i(t) \end{aligned} \tag{10}$$

where dt is the time step, N is the number of time steps, $x_i(t)$ is the position of the particle i at time t, $v_i(t)$ is the velocity of the particle i at time t, $f_i(t)$ is the force acting on the particle i at time t, and m_i is the mass of the particle i.

That means if we compare the result after all the integration steps of the two Verlet steps with the expected output of the analytical solution, then we can be sure that the Verlet integrator is working correctly.

3.2 Lennard-Jones potential force

The implementation of the Lennard-Jones potential force is done by using the following steps:

1. Calculate the distance between all pairs of atoms.

2. Calculate the potential between all pairs of atoms using the Lennard-Jones potential:

$$V_{LJ} = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{11}$$

3. Calculate the force between all pairs of atoms using the following equation:

$$F_{LJ} = -\frac{\partial V_{LJ}}{\partial r} = \left[\frac{48\epsilon\sigma^{12}}{r^{14}} - \frac{24\epsilon\sigma^{6}}{r^{8}}\right]\mathbf{r} \tag{12}$$

where ϵ is the depth of the potential well, σ is the distance at which the potential is zero, r is the distance between the two atoms, and $\mathbf{r}$ is the vector between the two atoms.

3.3 Berendsen thermostat

The Berendsen thermostat is implemented by using the following steps:

1. Calculate the temperature of the system.
2. Calculate the temperature ratio $\frac{T_0}{T}$.
3. Calculate the scaling factor $\sqrt{\frac{T_0}{T}}$.
4. Rescale the velocity of the particles in the system with the scaling factor.

where T is the temperature of the system, T_0 is the target temperature, and $\frac{T}{T_0}$ is the temperature ratio.

3.3.1 Test strategy for Berendsen thermostat

We have implemented two test cases for the Berendsen thermostat. The first test case is a test case where we test the Berendsen thermostat on a system with a single particle. The second test case is a test case where we test the Berendsen thermostat on a cubic lattice of size 5x5x5 with lattice constant=1.12. The test cases are implemented in the file.

Test case 1: Berendsen thermostat on a system with a single particle

Here, we make use of the derived equation in the lecture notes:

$$T(t) = T_0 + (T_1 - T_0) * exp(-\frac{t}{\tau}) \tag{13}$$

where $T(t)$ is the temperature of the system at time t, T_1 is the initial temperature of the system, T_0 is the target temperature of the system, and τ is relaxation time constant. and using this equation we can calculate the temperature of the system at every iteration t before and after the thermostat, and then make sure that the temperature relaxes exponentially to the target temperature.

Test case 2: Berendsen thermostat on a cubic lattice of size 5x5x5 with lattice constant=1.12

This test is confirming if the temperature really converges to the target temperature after running the simulation for some time. So, the test is as follows:

1. Create a cubic lattice of size 5x5x5 with lattice constant=1.12.

2. using time_step=0.01 , target temperature=1.0, and relaxation time constant= 10* time_step.

3. Run the simulation for 10000 time steps with the Berendsen thermostat.

4. In the last half of the simulation (5000 time steps), calculate the system temperature in every time step and make sure that the temperature converges to the target temperature.

3.4 Neighbor List

The neighbor list search is calculated by using the following steps:

1. Divide the simulation box into a grid of cells of size $cutoff$.
2. Sort all the particles into the cells.
3. Construct a neighbor list for each particle by looping over all the cells that are within a distance of $cutoff$ from the cell that the particle is in.
4. The $cutoff$ distance needs to be smaller than the cell size.
5. This $neighbor\_list$ is used to calculate the force between the particle and its neighbors.
6. Update the neighbor list in every time step.
7. Shift the potential energy by

$$4\epsilon[\left(\frac{\sigma}{cutoff}\right)^{12}-\left(\frac{\sigma}{cutoff}\right)^{6}] \tag{14}$$

3.5 Embedded atom method

The used Gupta potential is following the steps:

1. For each pair of atoms in the neighbour list, calculate the density contribution of the pair.
2. Compute the embedding energies of the atoms.
3. Compute embedding contribution to the potential energy.

4. Compute the repulsive energy and its derivative with respect to the distance between the atoms.
5. Compute the derivative of the embedded energy contributions.
6. Compute the pair force.
7. Sum up the pair forces to get the total force on each atom.
8. Sum up the repulsive energies to get the total potential energy.

3.6 Units and specification of the time unit

To work with EAM potential, we need to convert the units of the system to the units of the EAM potential. So, typically in Molecular dynamics simulations, the energy and the length are constants. and here we use E = eV and L = Angstrom. Where

$$\begin{aligned} 1eV &= 1.602176634 \times 10^{-19} J \\ 1Angstrom &= 10^{-10} m \end{aligned} \tag{15}$$

and forces are typically calculated in units of eV/Angstrom, which is the natural force unit for this system of units. To conclude, the units of the EAM potential used are:

- Energy: eV
- Distance: Angstrom
- Mass: amu
- Temperature: K

- Time: fs

3.6.1 time step for the gold potential

We have to fix either the mass unit or the time unit. And in this implementation we fixed the time unit of 1 fs for the gold potential. because using this time step is small enough to keep the energy conservation and also large enough to get to the melting point of gold in a reasonable time for clusters of layers from 3:12. To convert to a 1 fs time step we use the equation below:

$$m = \frac{[E][t^2]}{[L^2]} = 1.66 * 10^{-29} kg \tag{16}$$

where m is the mass of the particle, E is the energy unit, t is the time unit, and L is the length unit. and the commonly used mass unit is amu, so we can convert the mass unit to amu using the following equation:

$$m = 1.66 * 10^{-29} kg * 6.022 * 10^{23} kg/mol * 1000 = 0.009648 amu \tag{17}$$

And accordingly, we multiply all atoms masses by 103.642 to get a time step of 1 fs.

3.7 Parallelization using MPI

The parallelization of the simulation is done using MPI. The algorithm is as follows:

1. Initialize the MPI environment.

2. Get the number of processes.

3. Get the rank of the process.

4. Initialize the atoms.
5. Calculate the first verlet step to update the positions and velocities.
6. Exchange the atoms between the processes.
7. Update the ghost atoms and the neighbor list.
8. Calculate the forces.
9. Calculate the total energy.
10. Calculate the temperature.
11. Calculate the second verlet step to correct the velocities.
12. Do a Berendsen thermostat step.
13. To export result we have to gather the atoms from all processes to the root process and export the results then distribute the atoms again.
14. Keep looping until reaching the equilibrium state.
15. Finalize the MPI environment.

4 Results

4.1 Total energy as a function of time for different time steps

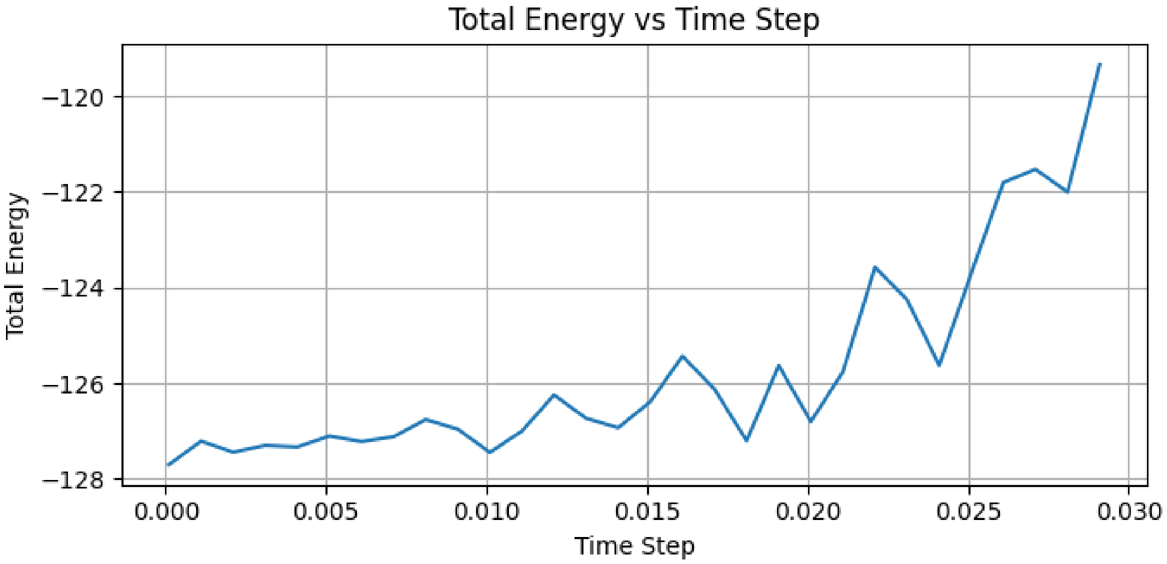

Figure 1: Total energy vs Time steps using parameters: start_time_step=1e-4, end_time_step=30e-3,step=1e-3, sigma=1, mass=1, epsilon=1, total_time=5000

As we increase the time step the total energy of the system increases and at some point the system becomes unstable and the atoms start to move away from each other and the simulation stops. This is because the time step is too large and the atoms are not able to move to the correct position in the next time step.

4.2 Snapshots sequence of LJ simulation

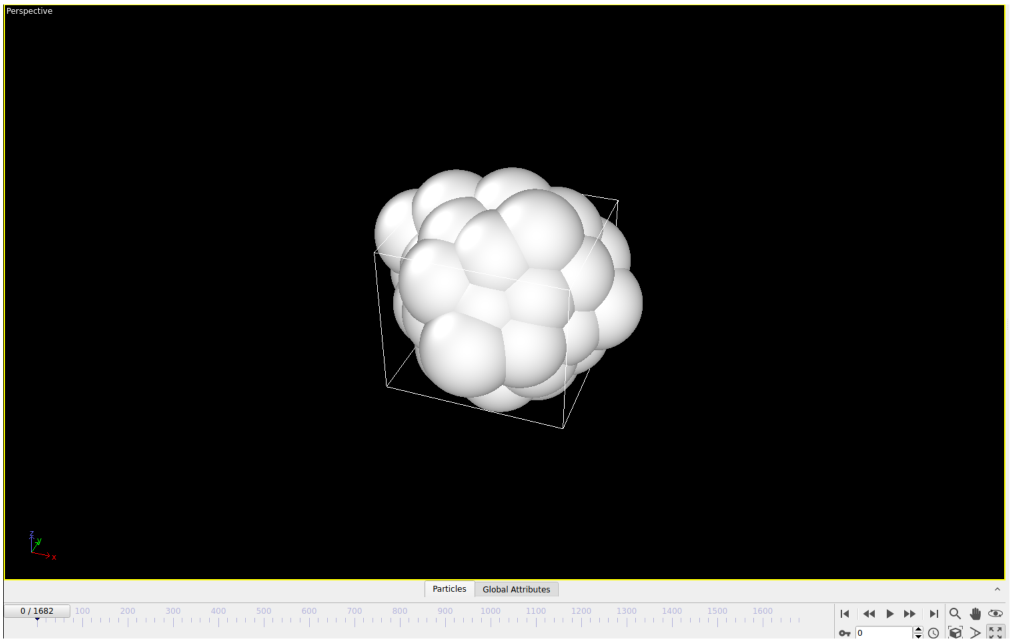
Perspective
Particles
Global Attributes
0 / 1682
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
0

Figure 2: LJ simulation snapshot1

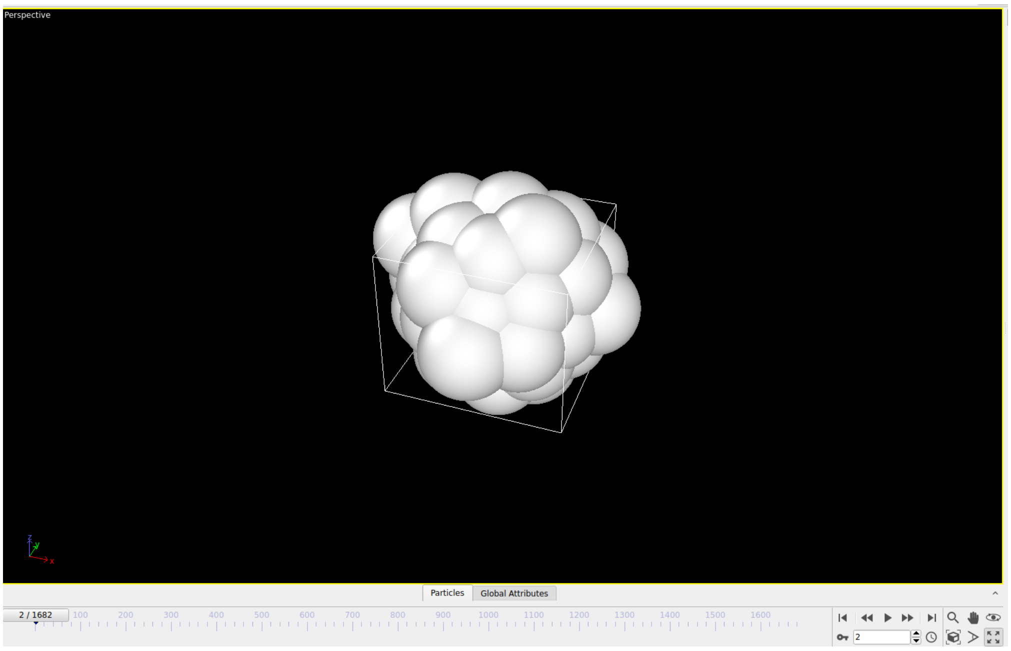
Perspective
Particles
Global Attributes
2 / 1682
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
2

Figure 3: LJ simulation snapshot2

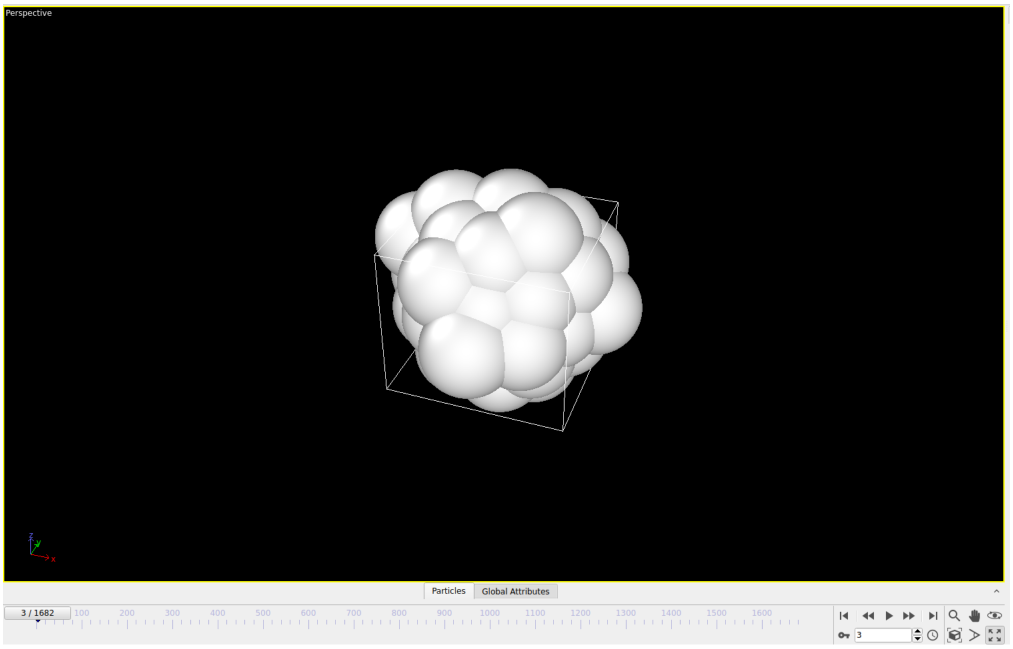
Perspective
Particles
Global Attributes
3 / 1682
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
3

Figure 4: LJ simulation snapshot3

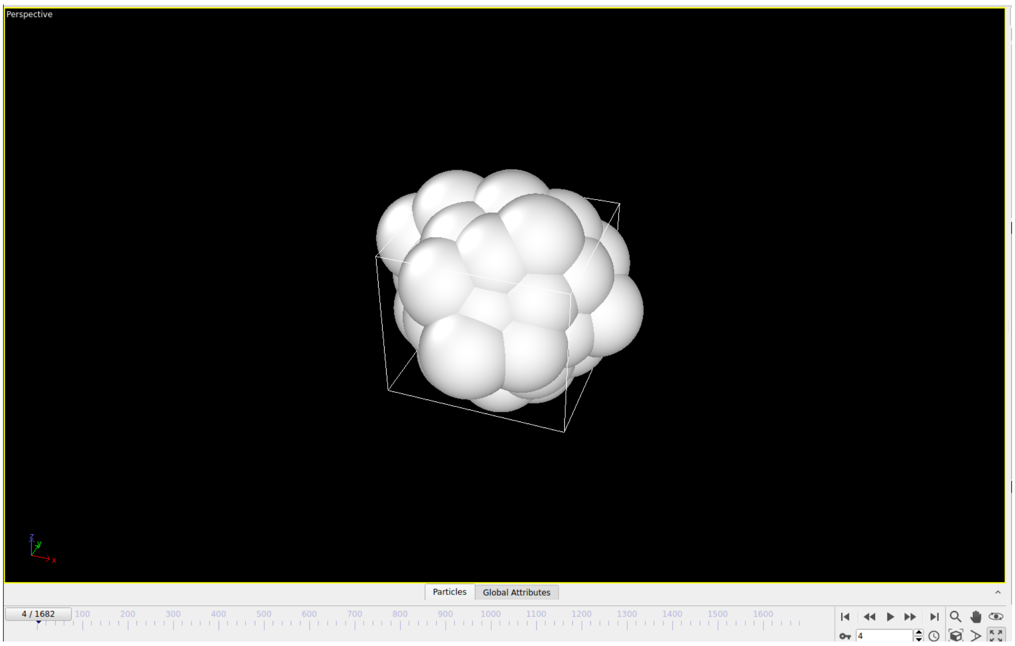
Perspective
Particles
Global Attributes
4 / 1682
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
4

Figure 5: LJ simulation snapshot4

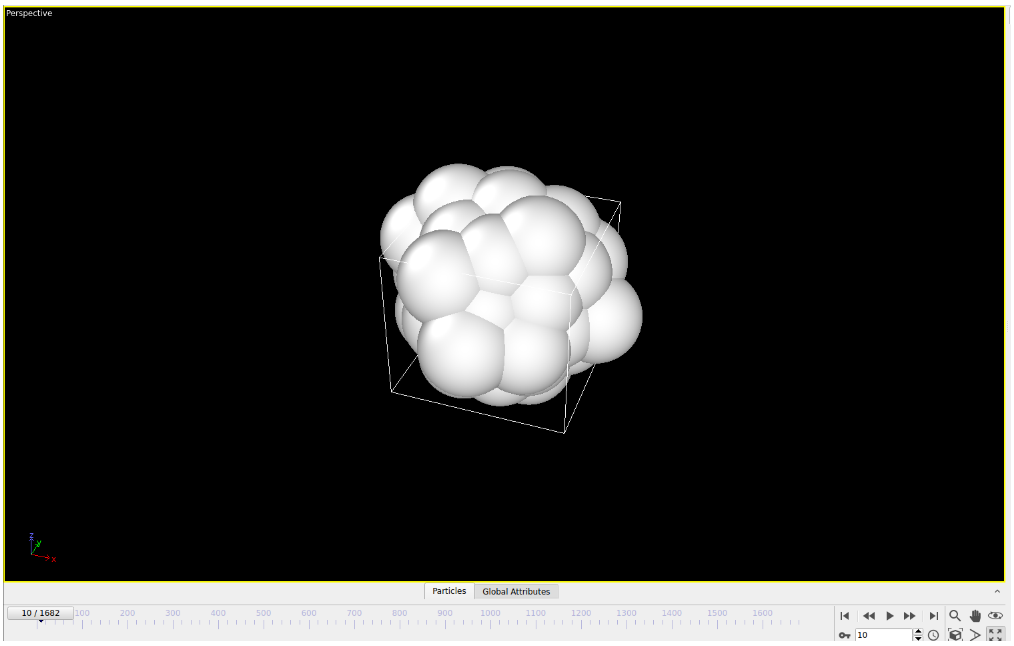
Perspective
Particles
Global Attributes
10 / 1682
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
10

Figure 6: LJ simulation snapshot5

4.3 simulation time as a function of the size (number of atoms) without neighbor list

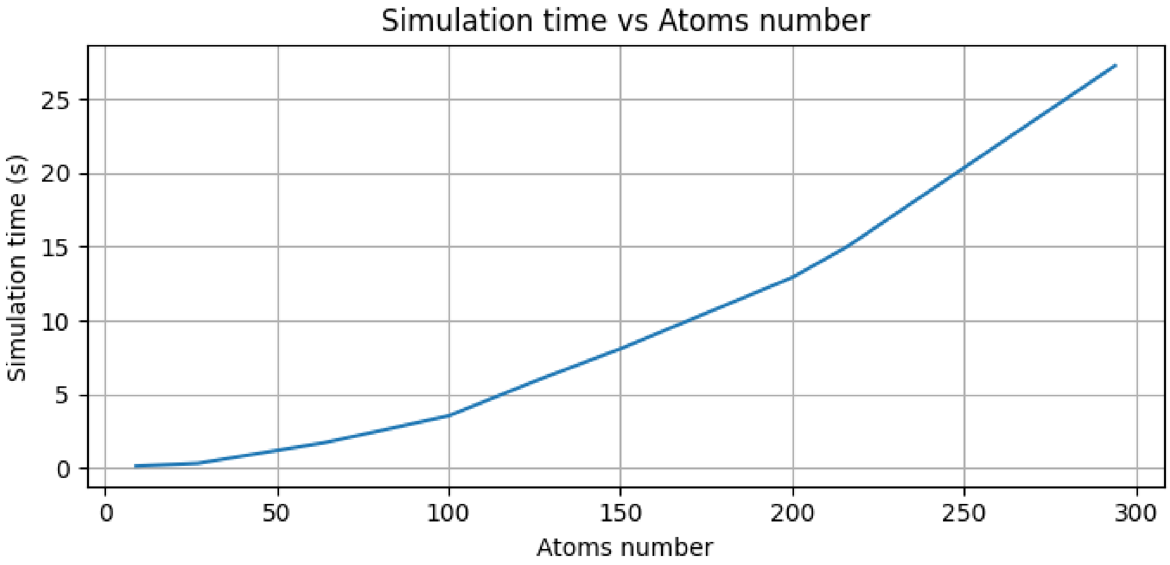

Figure 7: Simulation time vs Atoms number using parameters: time_step=1e-3, sigma=1, mass=1, epsilon=1, total_time=5000, relaxation_time_start = 10* time_step,then after reaching equilibrium, it increases to 50*time_step.

Here we notice that the simulation time increases quadratically with the number of atoms in the system. This is because in the energy update step in the LJ simulation we have to calculate the energy between all the pairs of atoms in the system. So the time complexity of the energy update step is $O(N^2)$ where N is the number of atoms in the system. For example, In the figure above,The simulation time for 100 atoms is 3.5s and for 200 atoms is 12.9s which is almost 4 times more than the simulation time for 100 atoms.

4.4 simulation time as a function of the size (number of atoms) with neighbor list

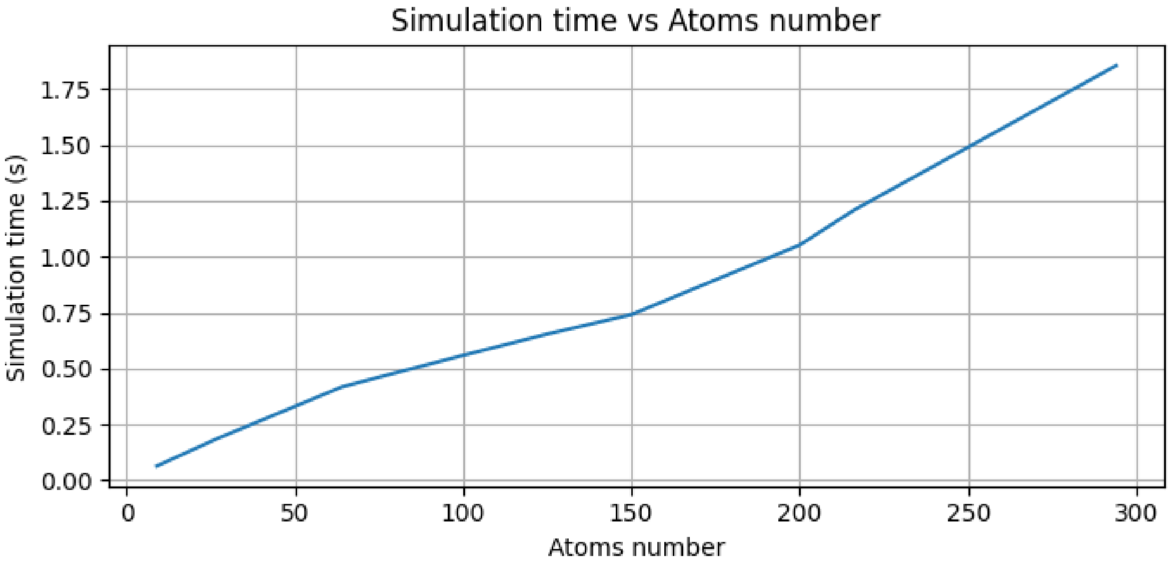

Figure 8: Simulation time vs Atoms number using parameters: time_step=1e-3, sigma=1, mass=1, epsilon=1, total_time=5000, relaxation_time_start = 10* time_step,then after reaching equilibrium, it increases to 50*time_step, and cutoff_radius = 1.5.

Here, it's very clear the linearity of the simulation time with the number of atoms in the system. By introducing the concept of just calculating the energy between the atoms that are close to each other, we have reduced the time complexity of the energy update step from $O(N^2)$ to $O(N)$ where N is the number of atoms in the system. For example, In the figure above,The simulation time for 100 atoms is 0.56s and for 200 atoms is 1.05s which is almost 2 times more than the simulation time for 100 atoms.

4.5 total energy vs temperature

The setup for this section is as follows:

- time_step=1fs
- mass=20405.7294 amu
- relaxation_time_start = 10* time_step,then after 500 step, it increases to 1e10*time_step
- cutoff_radius = 10 Angstrom
- added_energy_in_each_experiment=0.01*no_atoms
- atomic_distance = 2.885
- cluster_sizes=3:12
- for cluster sizes 3:8

  target_temp = 300 K

  initial_simulation_time=10000fs

  experiment_simulation_time=5000fs

  number_experiments=26
- for cluster sizes 9:12

  target_temp = 500 K

  initial_simulation_time=5000fs

experiment_simulation_time=2000fs

number_experiments=40

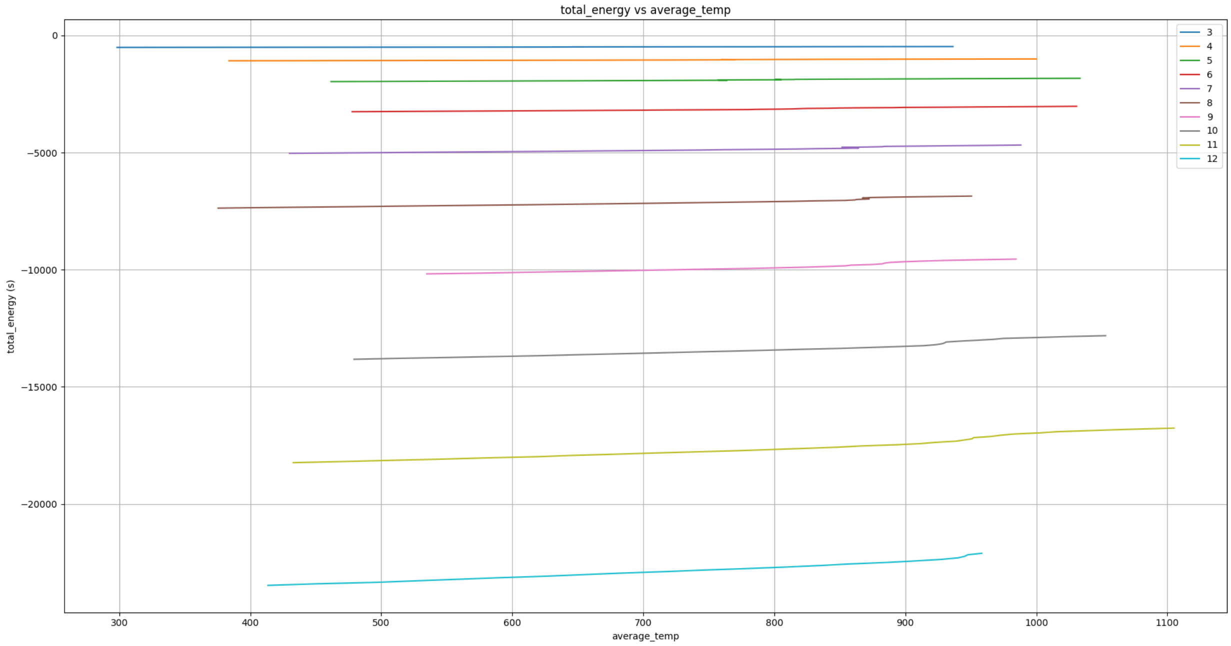

Figure 9: Total energy vs Temp

As the temperature increases the total energy of the system increases. This is because the atoms are moving faster and they are colliding with each other more often. This leads to more energy being transferred between the atoms and the total energy of the system increases. The total energy of the system is conserved and it is equal to the sum of the kinetic energy of the atoms and the potential energy between the atoms. So as the temperature increases the kinetic energy of the atoms increases and the potential energy between the atoms decreases. This is why the total energy of the system increases as the temperature increases.

Near the melting point we notice that Adding more energy to the system doesn't increase the total energy of the system anymore but instead it decreases the total energy of the system. This is because the atoms are not able to absorb the energy and the system is in equilibrium. So the atoms are vibrating around their equilibrium position and the energy is being transferred between the atoms and the total energy of the system decreases.

4.6 melting point versus cluster size

The setup for this section is the same as the one used in the previous section. Here,

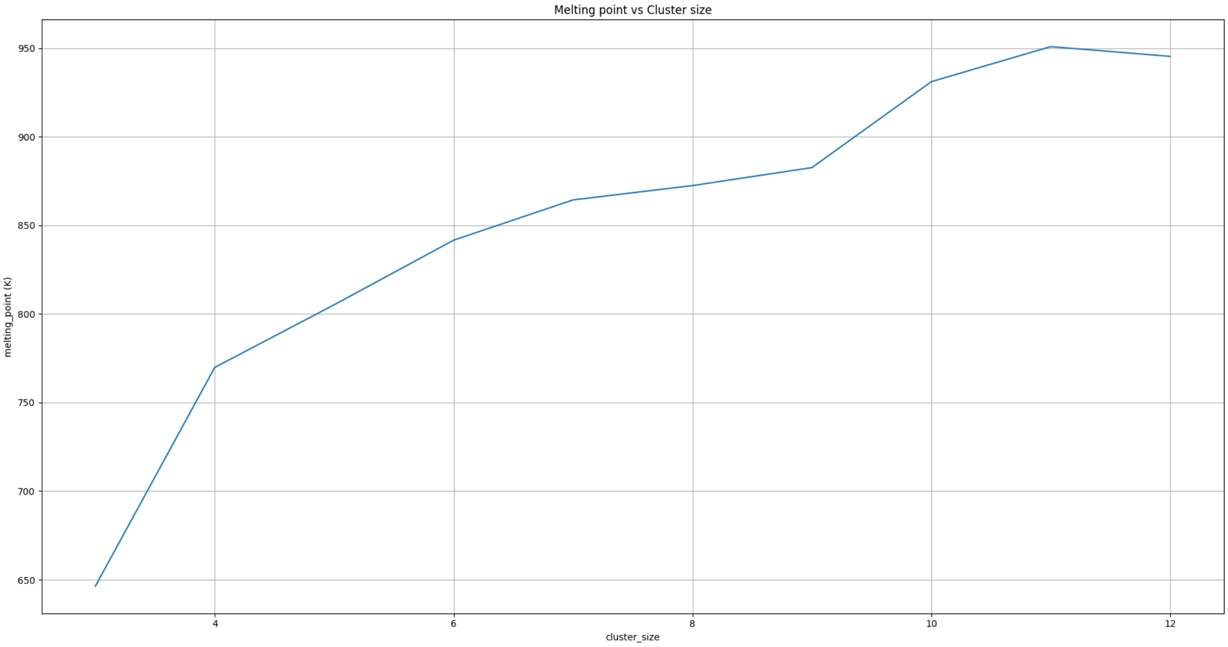

Figure 10: Melting point vs cluster size

as the cluster size increases the melting point increases because to melt a bigger cluster of atoms we need to add more energy to the system. This is because the atoms in the bigger cluster are more tightly packed and they are more difficult to melt.

4.7 heat capacity versus cluster size

The setup for this section is the same as the one used in the previous section. Here,

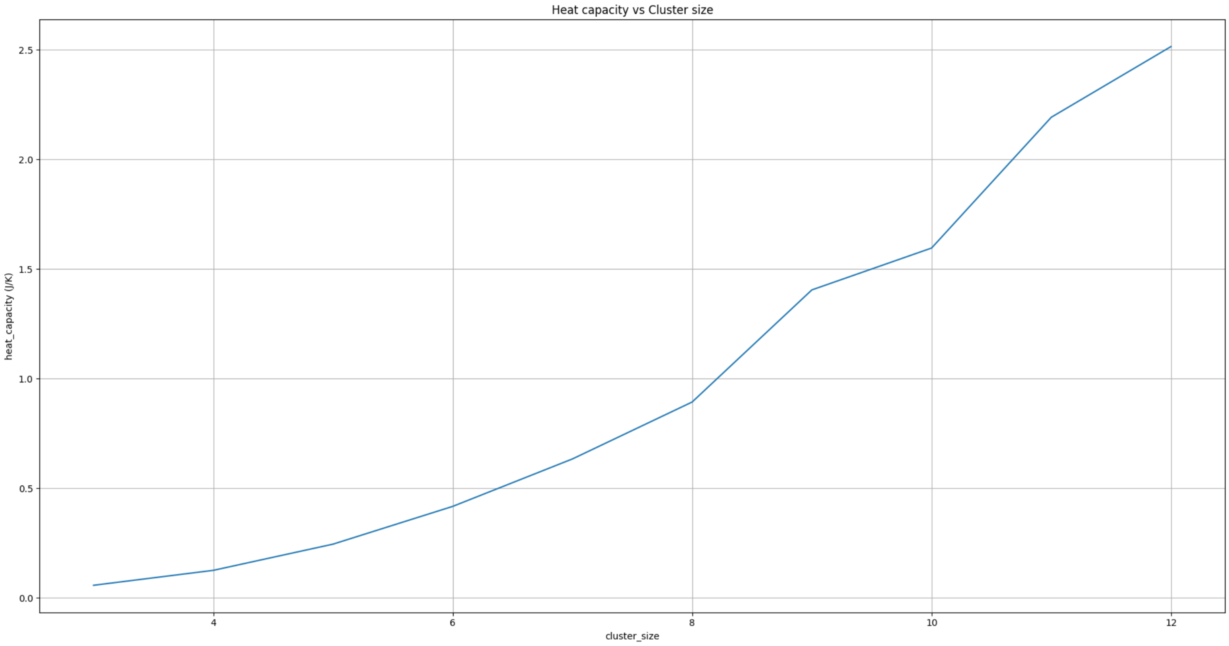

Figure 11: Heat capacity vs cluster size

as the cluster size increases the heat capacity increases.

4.8 latent heat versus cluster size

The setup for this section is the same as the one used in the previous section. Here,

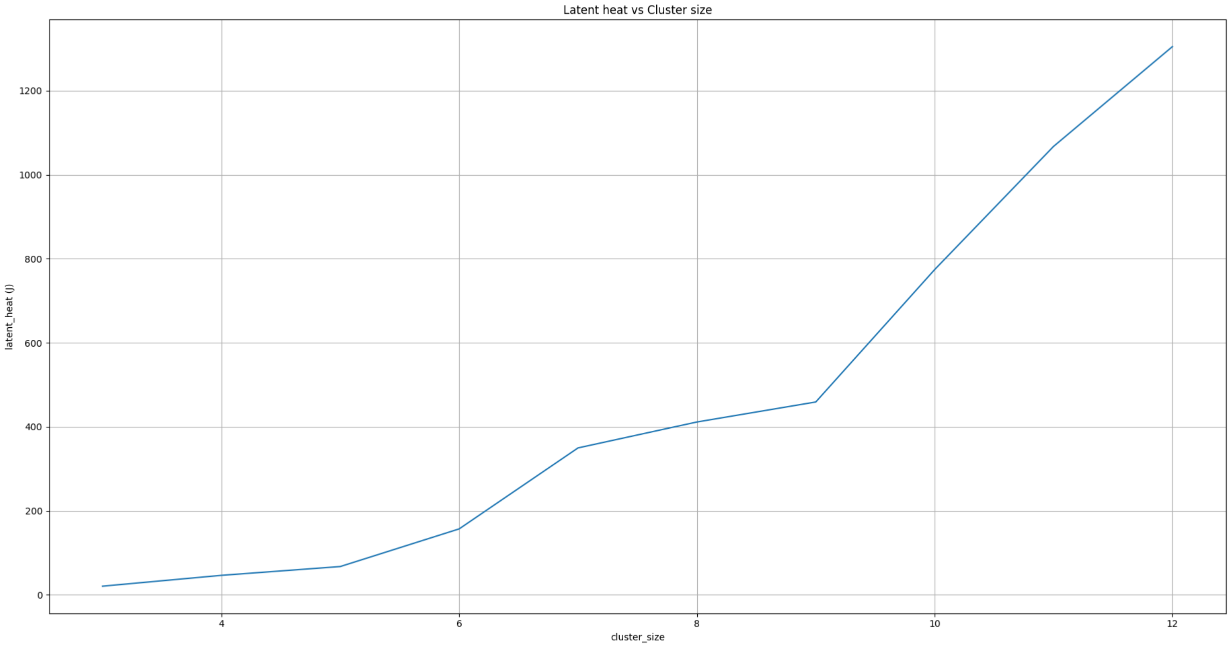

Figure 12: Latent heat vs cluster size

as the cluster size increases the latent heat increases.

4.9 Energy conservation with MPI parallelization

4.10 Nanowire defects

5 Conclusion

Bibliography

[1] H. J. Berendsen, J. v. Postma, W. F. Van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of chemical physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[2] R. P. Gupta, "Lattice relaxation at a metal surface," *Physical Review B*, vol. 23, no. 12, p. 6265, 1981.

[3] F. Cleri and V. Rosato, "Tight-binding potentials for transition metals and alloys," *Physical Review B*, vol. 48, no. 1, p. 22, 1993.