

Parallel Computing for Molecular Dynamics Simulation

Yi Zhang

Introduction

- ▶ In materials modeling, we often need to follow the evolution of the materials by calculating the positions of all the atoms.
- ▶ Each atom is under the influence of other atoms. The forces of other atoms will cause the atom to move.
- ▶ This is a complex multi-body problem and does not have an analytic solution. As a result, computer is often used to simulate the evolution of the system.
- ▶ This final project will focus on one particular material: Gold. Both serial code and parallel code will be produced to simulate the system. Code will be written in C. In the parallel case, MPI is used.

The system to simulate

- ▶ Gold is chosen as system to simulate due to its simplicity: there is only one type of atoms in the system.
- ▶ Some experimental properties of gold:
 - Symbol: Au
 - Atomic weight: 196.96655
 - Structure: fcc, with cell size: $a = b = c = 4.0782$ Angstrom
- ▶ Bulk modulus: 220 GPa
- ▶ Melting point: 1337.33 K

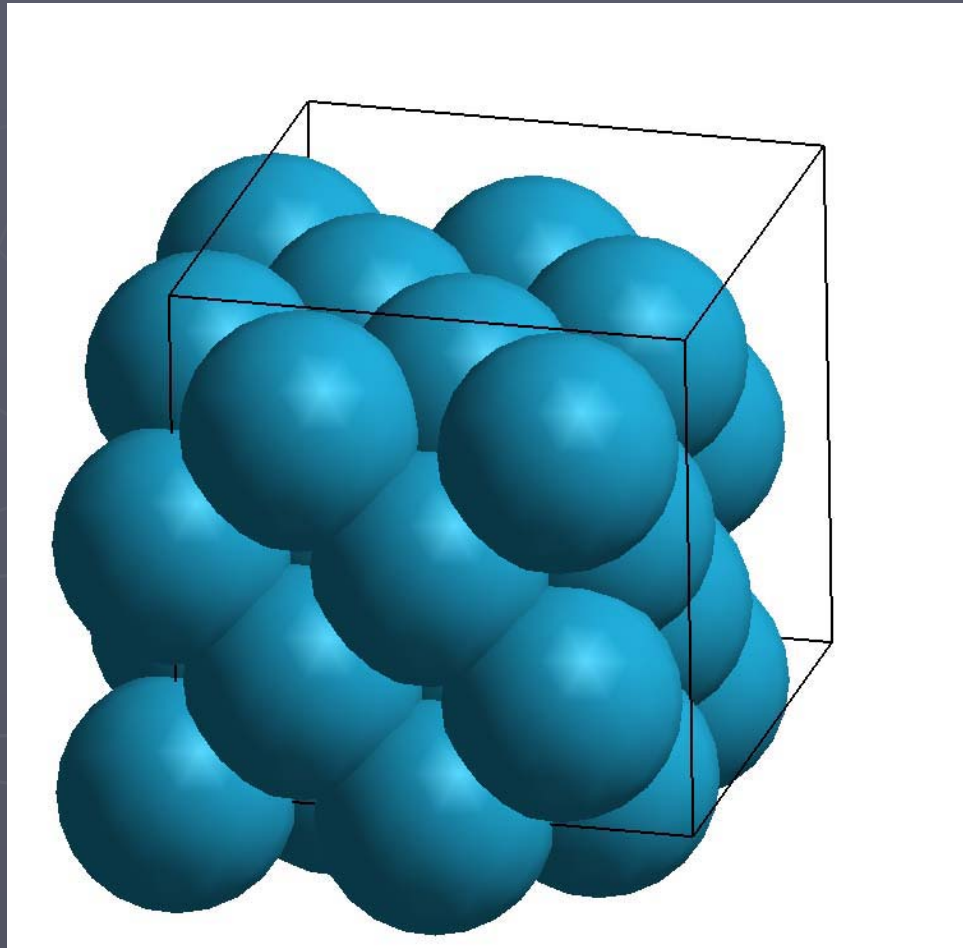
The system to simulate

- ▶ Lennard-Jones potential is used.
- ▶ Forces exist between every pair of atoms
- ▶ Force depends only on the distance
- ▶ $E = A/r^{12} - B/r^6$

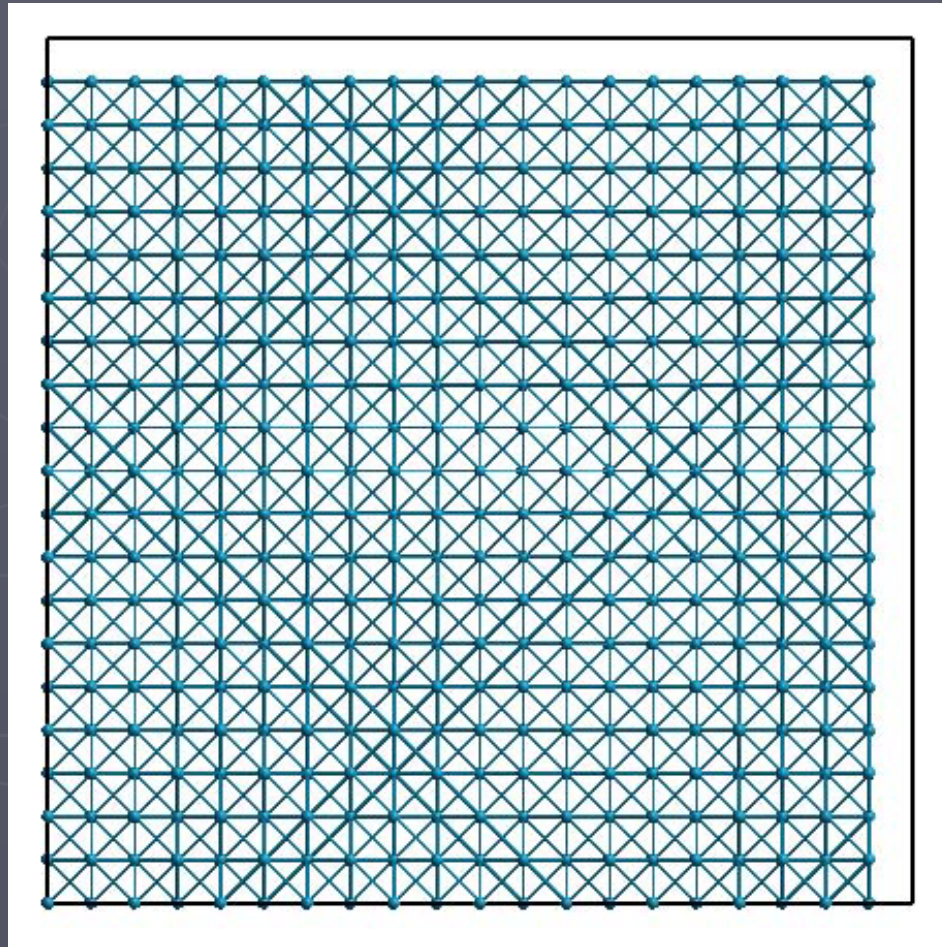
The system to simulate

- ▶ $E = A/r^{12} - B/r^6$
- ▶ $A = 158675.062040 \text{ (eV*angstrom}^{12}\text{)}$
- ▶ $B = 463.262145 \text{ (eV*angstrom}^6\text{)}$
- ▶ Put these potential parameters into GULP, we can calculate that:
 - The lattice constant: 4.078200 (Angstrom)
 - Bulk modulus: 219.99974 (GPa)
 - Close to experimental value

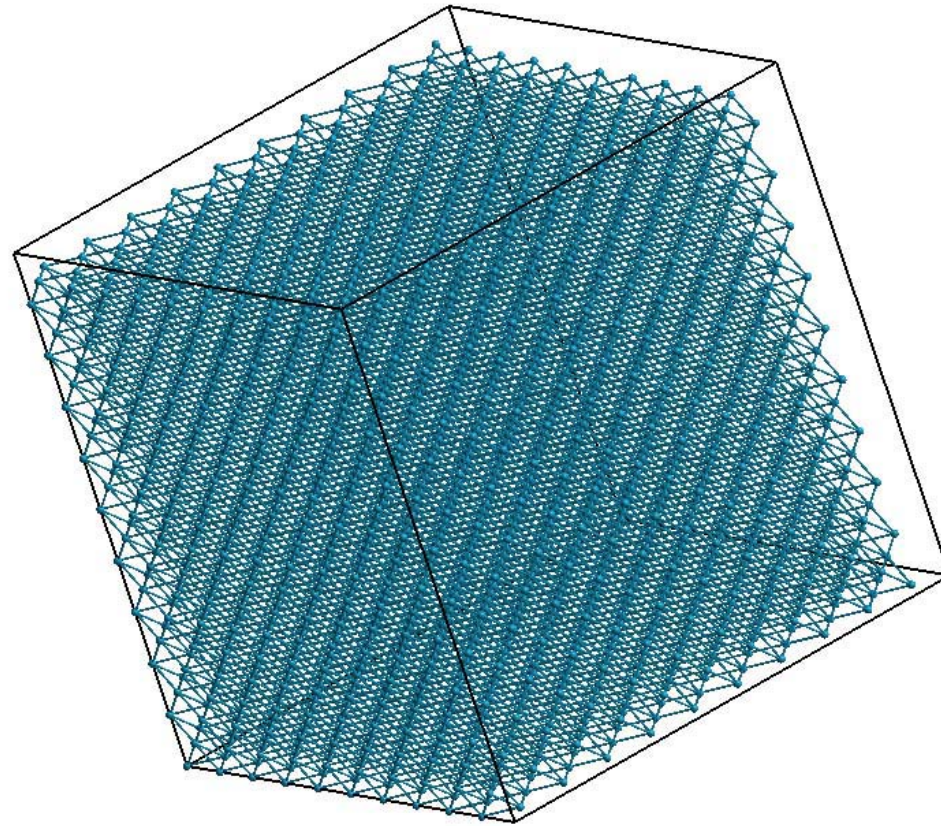
The system to simulate



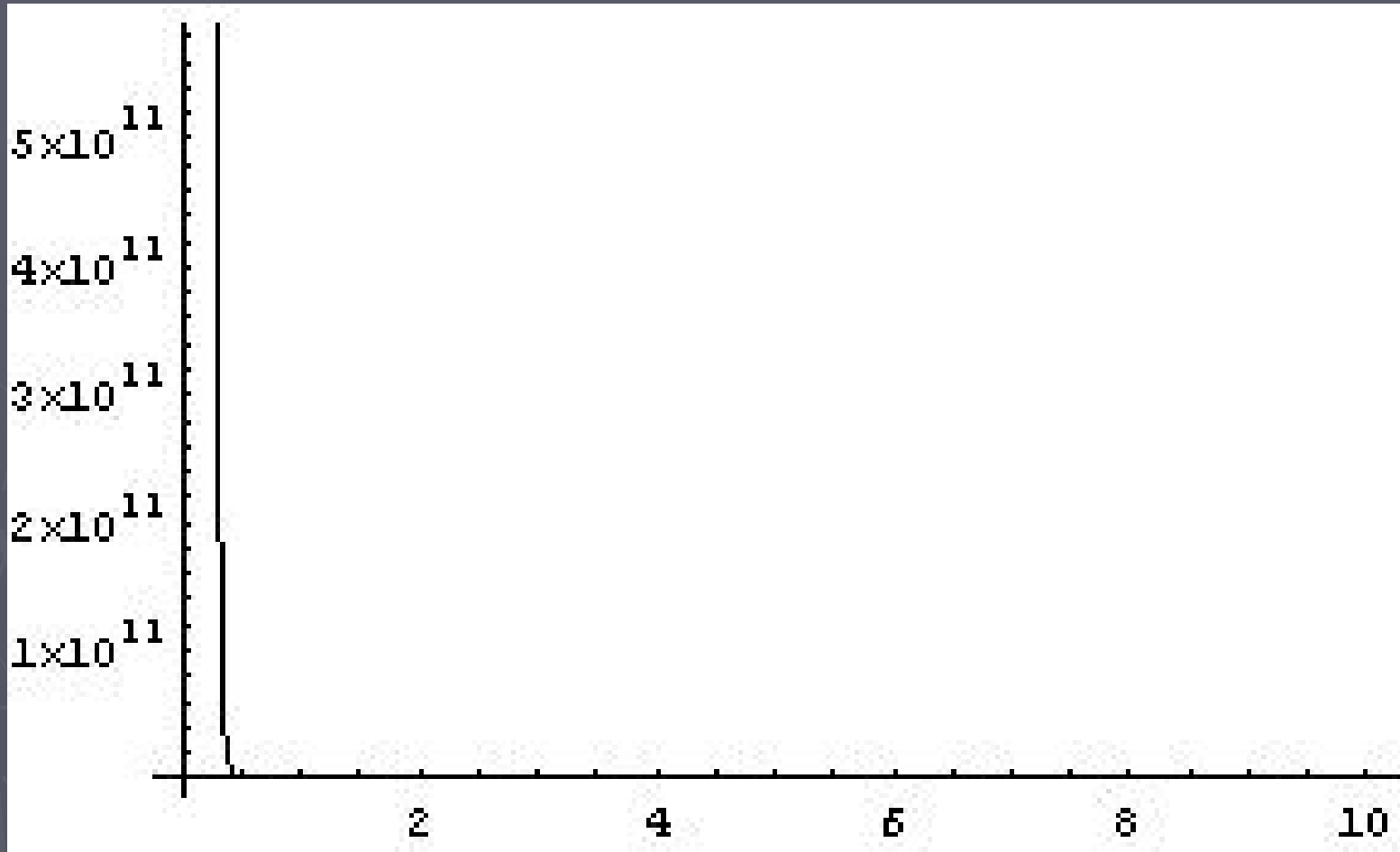
The system to simulate



The system to simulate



The system to simulate



The system to simulate

- ▶ Choose force cutoff to 6 angstroms.
- ▶ Included 3rd nearest neighbor.
($4.078/\sqrt{2}$, 4.078, $4.078*\sqrt{2}=5.767$)
- ▶ Use the formula $E = A/r^{12} - B/r^6$, $E = 0.01$ eV at $r = 6$ angstroms, which is indeed negligible)

The serial code

- ▶ 1st line: super cell size, 3 integers.
- ▶ 2nd line:
 - ▶ 1st: time interval per step, in units of ps (pico seconds, 10⁻¹² seconds), double type.
 - ▶ 2nd: equilibration time step, integer type.
 - ▶ 3rd: velocity scaling interval, integer type.
 - ▶ 4th: total time step, integer type.
- ▶ 3rd line:
 - ▶ 1st: temperature, in unit of Kelvin, double type.
 - ▶ 2nd: force cutoff value, in unit of angstrom, double type.
- ▶ 4th line:
 - ▶ 1st: start time step of dumping atom positions to output file, integer type.
 - ▶ 2nd: time step interval of dumping atom positions to output file, integer type.
- ▶ 5th line:
 - ▶ time step interval of updating neighbor list.

The serial code

- ▶ Super cell size
- ▶ Time interval per step
- ▶ Equilibration time step: scaling velocities
- ▶ Velocity scaling interval
- ▶ Total time step
- ▶ Output atom positions
- ▶ Neighbor list updating interval

The serial code

- Output file:
 - Echoing input
 - Output atom positions at specified time steps.

Generate initial positions

► Unit cell

- $(0, 0, 0)$
- $(0.5 * a, 0.5 * a, 0)$
- $(0.5 * a, 0, 0.5 * a)$
- $(0, 0.5 * a, 0.5 * a)$

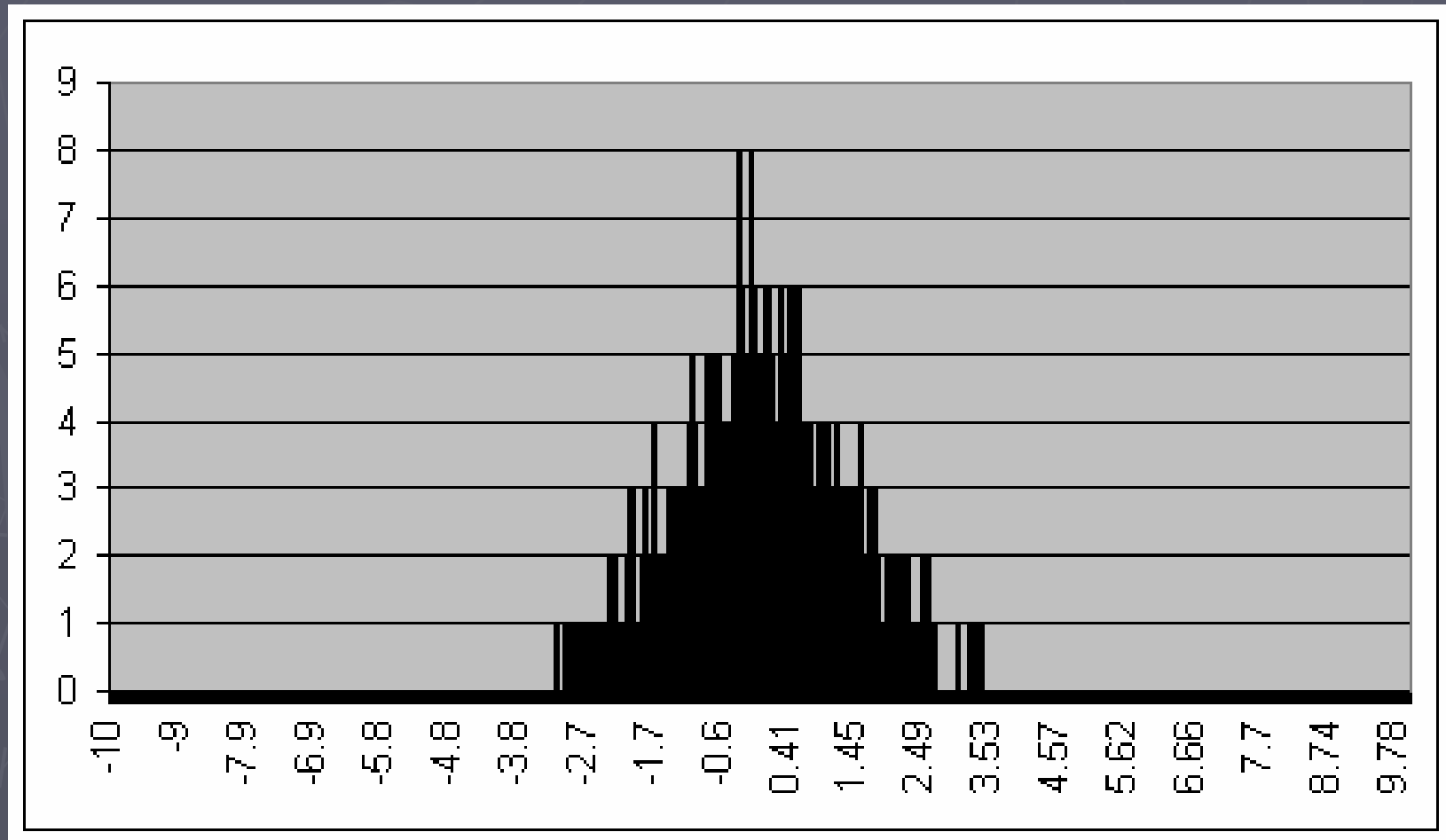
► Super cell

Generate the initial velocities

- ▶ According to Maxwellian distribution
- ▶ To generate Gaussian random numbers
 - The polar method of G. E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in The Art of Computer Programming, Volume 2: Seminumerical Algorithms, section 3.4.1, subsection C, algorithm P.
- ▶ Scale velocities

Gaussian random number generator

► 4000 numbers



Velocity Scaling

- ▶ From the equi-partition law: kinetic energy $\frac{3}{2} kT$, where k is the Boltzmann constant.
- ▶ N atoms, we have total kinetic energy:
 - $\text{targetE} = \frac{3}{2} kT * N$
- ▶ The real kinetic energy realE of the system: sum over all atoms.
- ▶ The scaling factor will be $\text{sqrt}(\text{targetE} / \text{realE})$
- ▶ Each velocity is then multiplied by this scaling factor.

Periodic boundary condition

- ▶ To calculate the distance between atom A and atom B, we consider the distances between all of A's images (including A) and all of B's images (including B), and the shortest distance of all these distances will be the distance we will use.
- ▶ This is based on the assumption that the effect of the second and higher order nearest images pairs can be neglected comparing the nearest pair of images.

Periodic boundary condition

- ▶ $xDis = x[j] - x[i]$
- ▶ $xDis - Len * \text{floor}(xDis / Len + 0.5)$
- ▶ Similarly for $yDis$ and $zDis$
- ▶ $distance = \sqrt{xDis * xDis + yDis * yDis + zDis * zDis}$

Periodic boundary condition

- ▶ if the positions go beyond the boundary, we need to change to the positions of the image that comes in from the other side
- ▶ $rx_new = rx + xLen * n$
- ▶ $0 \leq rx + xLen * n < xLen$
- ▶ $(-rx) / xLen \leq n < (xLen - rx) / xLen = (-rx) / xLen + 1$
- ▶ $n = \text{ceil}(-rx / xLen)$.
- ▶ Thus:
- ▶ $rx[i] = rx[i] + xLen * \text{ceil}(-rx / xLen)$

Velocity Verlet

$$r(t + dt) = r(t) + v(t)dt + \frac{1}{2}a(t)dt^2$$

$$v(t + dt) = v(t) + \frac{1}{2}[a(t) + a(t + dt)]dt$$

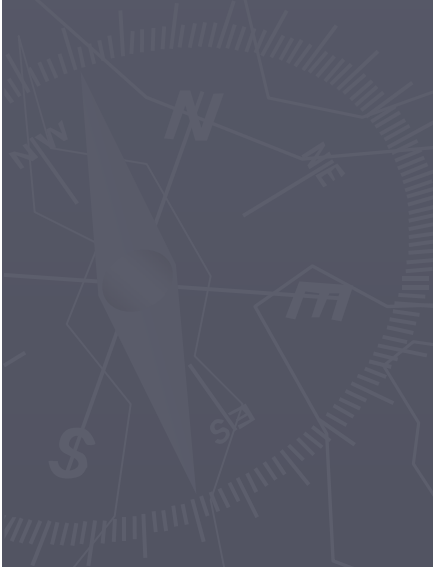
$$= v(t) + \frac{1}{2}a(t)dt + \frac{1}{2}a(t + dt)dt$$

The parallel code

- Geometry partition.
- For example, 8 processes:
 - Part 0: $0 \leq x < xLen/2$; $0 \leq y < yLen/2$; $0 \leq z < zLen/2$.
 - Part 1: $0 \leq x < xLen/2$; $0 \leq y < yLen/2$; $zLen/2 \leq z < zLen$.
 - Part 2: $0 \leq x < xLen/2$; $yLen/2 \leq y < yLen$; $0 \leq z < zLen/2$.
 - Part 3: $0 \leq x < xLen/2$; $yLen/2 \leq y < yLen$; $zLen/2 \leq z < zLen$.
 - Part 4: $xLen/2 \leq x < xLen$; $0 \leq y < yLen/2$; $0 \leq z < zLen/2$.
 - Part 5: $xLen/2 \leq x < xLen$; $0 \leq y < yLen/2$; $zLen/2 \leq z < zLen$.
 - Part 6: $xLen/2 \leq x < xLen$; $yLen/2 \leq y < yLen$; $0 \leq z < zLen/2$.
 - Part 7: $xLen/2 \leq x < xLen$; $yLen/2 \leq y < yLen$; $zLen/2 \leq z < zLen$.

Geometric partition

- ▶ Hard to keep track of atoms
- ▶ Hard to synchronize between processes
- ▶ Hard to update neighbor list



Partition based on index

- ▶ Positions are synchronized before updating neighbor list
- ▶ Velocities and accelerations are not synchronized

Parallel Code

- ▶ Old positions of neighbors are used
- ▶ Acceleration calculated based on old positions of atoms belong to other process is used
- ▶ Neighbor list updating interval
- ▶ Approximation won't be any worse than our previous many approximations made

Parallel Code

- ▶ Every process outputs its own output file: [outfile]_i for process i.
- ▶ Every process only output the positions of atoms that belong to it.
- ▶ Process 0 will be responsible to read the input file, generate the initial positions and velocities, and broadcast initial values.

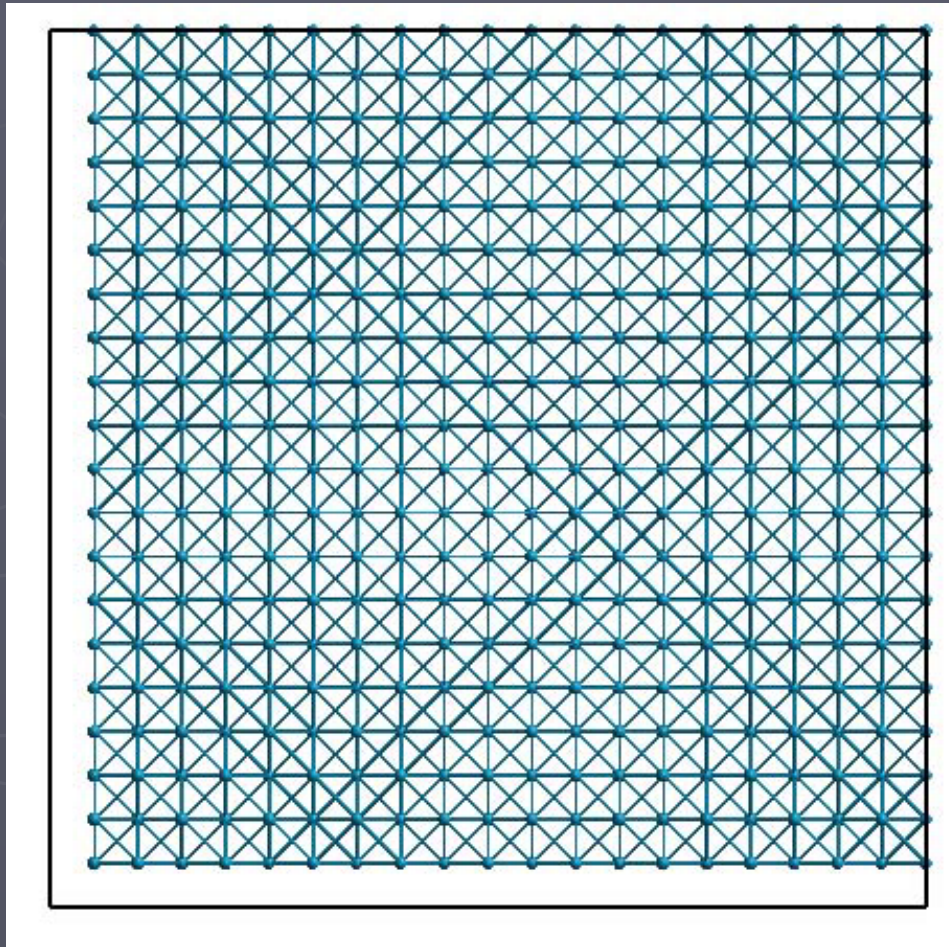
Parallel Velocity Scale

- ▶ Partial Kinetic Energy Sum
- ▶ “Reduce” to total energy.
- ▶ Calculate scaling factor
- ▶ Broadcast the factor
- ▶ Do the scaling locally

Accuracy of parallel code

- ▶ The accuracy of the parallel code is determined by comparing the output from the parallel code with that from the serial code.
- ▶ Normally the accuracy of the serial code should be tested by comparing results to physical reality.

After time step 1



Comparing with serial code

- ▶ Disable random number generator.
- ▶ After 1 time step: average distance 0.0
- ▶ After 100 time step: 0.1134226

Performance: system size

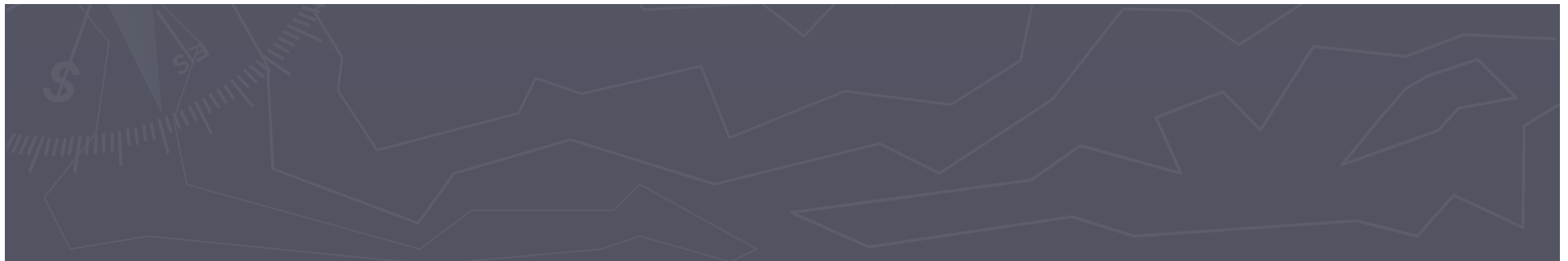
The system is run to 1000 time step; neighbor list is updated every 10 steps; 4 processes are used for parallel code.

System size	10*10*10	15*15*15	20*20*20
Serial code time (seconds)	243	1868	9256
Parallel code time (seconds)	96.8537	836.006	4243.07
Serial_time/parallel_time	2.51	2.23	2.18

Performance: # of processes

The system size is $15 \times 15 \times 15$; system is run to 1000 time step; neighbor list is updated every 10 steps

# of processes	serial code	2	4	8
Running time (seconds)	1868	1665.79	836.006	405.113



Performance: neighbor list updating interval

The system size is $15 \times 15 \times 15$; # of processes: 8; system is run to 1000 time step.

Neighbor list update interval	10	100	never
Running time (seconds)	405.113	90.2313	54.192

Conclusion

- ▶ In this specific scenario, it is worth the effort to develop the parallel code, because the speed-up is significant.
- ▶ Over-simplified
- ▶ None-the-less a good start

THE END

