Molecular Dynamics
Improving Neighbor Table

Michael Graziano
Sahan Bandara

Problem Statement

- Molecular dynamics systems = very common model
  - Two-body interatomic interactions (Lennard-Jones potential function)
  - Periodic boundaries to represent “infinite” space
- Brute Force -> $O(N^2)$ performance when comparing atom pairs
- Improvement: Verlet Table Algorithm / Cell Linked List Algorithm
- Further improvement: https://arxiv.org/pdf/physics/0311055.pdf
  - Combines above techniques
  - Uses additional techniques to improve table update frequency/memory organization
- How does parallelization of system work?

Tasks

1. Research/study molecular dynamics model and algorithms
2. Implement/source code that can be used to create model.
3. Make improvements to model by:
    a. Parallelizing
    b. Following methods described in https://arxiv.org/pdf/physics/0311055.pdf
4. Demonstrate improvements/errors in system

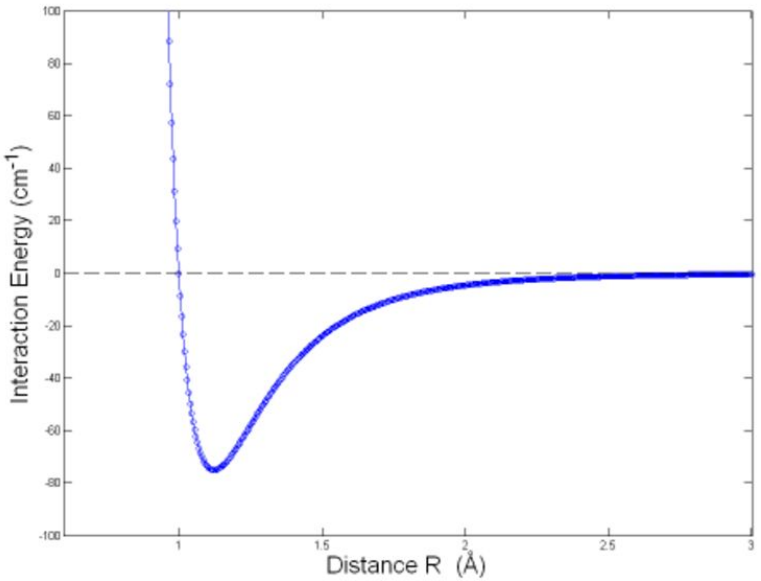
Goals

- Demonstrate parallelization of simulation
- Show the progressive improvement between conventional methods/combined methods
- Understand the use of graph algorithm in solving a complex problem

Molecular Dynamics Basics

- Interaction between particles in a “infinite” space
  - Utilize periodic boundaries
- Conservation of total energy
  - $E = E_{kin} + E_{pot}$
  - $E_{pot} = \sum_{i=1..N} \sum_{j>i} e_{pot}(r_{ij})$ where r_{ij} = distance between particles “i” and “j" (Equation 2.2)[1]
- Lennard-Jones Potential
  - Neutral particles
  - Repulsion Forces (no collisions)

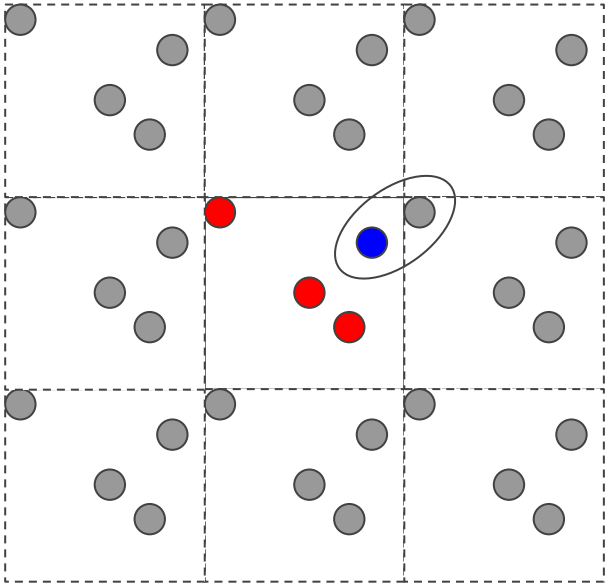
Simulation of Lennard-Jones potential [2]



[1] Basics of Molecular Dynamics. Available from: C2_for.pdf
[2] Six Degree-of-Freedom Haptic Rendering for Biomolecular Docking - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/The-simulation-of-Lennard-Jones-potential_fig1_220110295 [accessed 1 May, 2018]

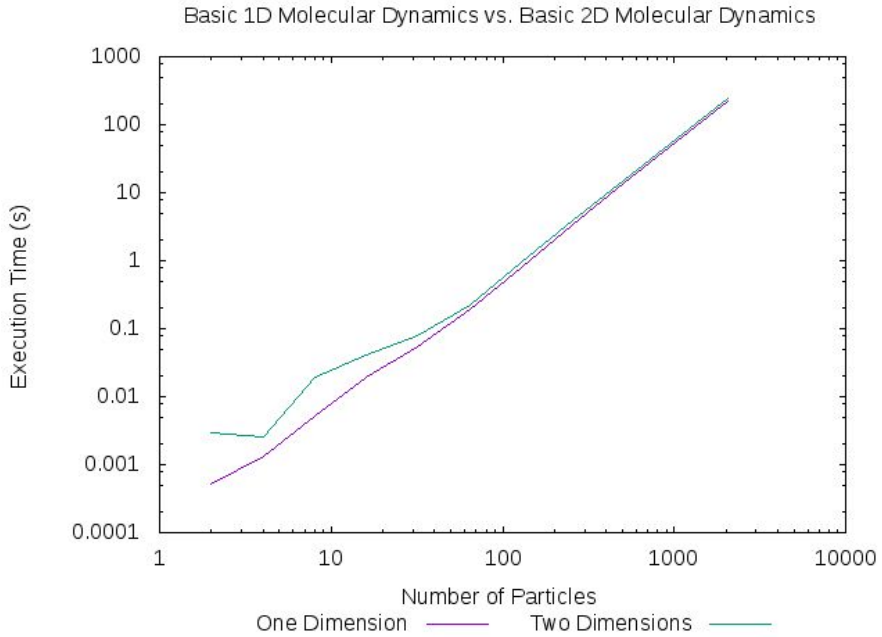
Periodic Boundaries

- Avoid surface effects
- Simulation particles see “images”
  - Image particles follow movement of simulation particle
- Pair interaction with closest particle
  - Minimum Image Criterion
- Constant number of particles
  - Simulation exit -> Image enter



Basic Method - 1D & 2D

- Time step range: 1E-5 to 1E-12
- Maximum Iterations: 5000
- 2D performance tracks with 1D
  - Particles vs Array Size
  - Additional dimension seen at low particle density
- 2048 particles require 226 seconds
  - Normal systems 1E6(+) particles!

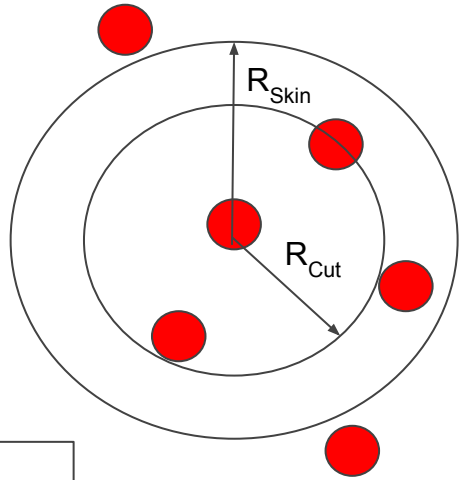


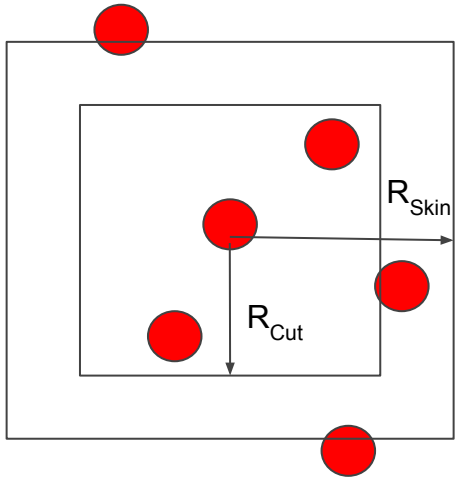
Acceleration - The Problem Child

- System update in distinct time steps
  - Step **SUFFICIENTLY** small for stability
- Update method - Verlet Velocity:
  - From *Basics of Molecular Dynamics* (C2_for.pdf):
    - $\mathbf{v}(t + 0.5*dt) = \mathbf{v}(t) + \mathbf{a}(t)*(0.5*dt)$
    - $\mathbf{r}(t + 0.5*dt) = \mathbf{r}(t) + \mathbf{v}(t + 0.5*dt)*dt$
    - $\mathbf{a}(t + dt) = \mathbf{F}(\mathbf{r}(t+dt))/m$
    - $\mathbf{v}(t + dt) = \mathbf{v}(t + 0.5*dt) + \mathbf{a}(t + dt)*(0.5*dt)$
- Acceleration calculation $\mathbf{O}(N^2)$ <- How to reduce impact?

Neighbor Table

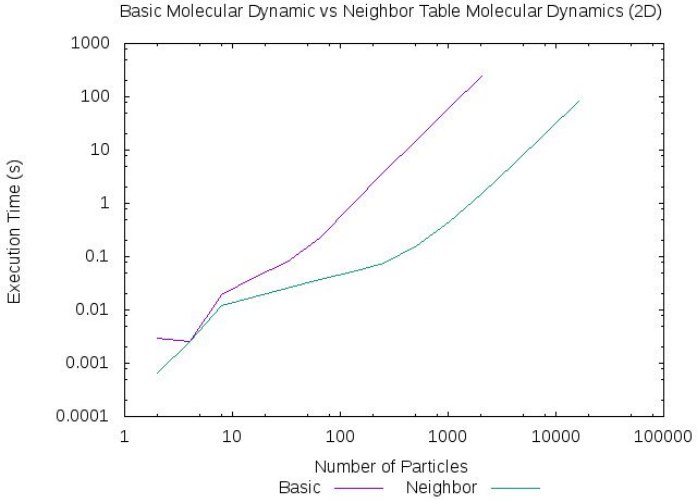
- R_{Cut} -> Potential Cutoff
- R_{Skin} -> Considered Particle
- Circular boundaries
  - Used Square boundaries
- Data Structure for N particles:
  - Array (size N) of...
    - Pointers
    - Linked-Lists
    - Vectors
- Vector chosen for simplicity
  - Not efficiency
- Table reconstructed a set number of iterations

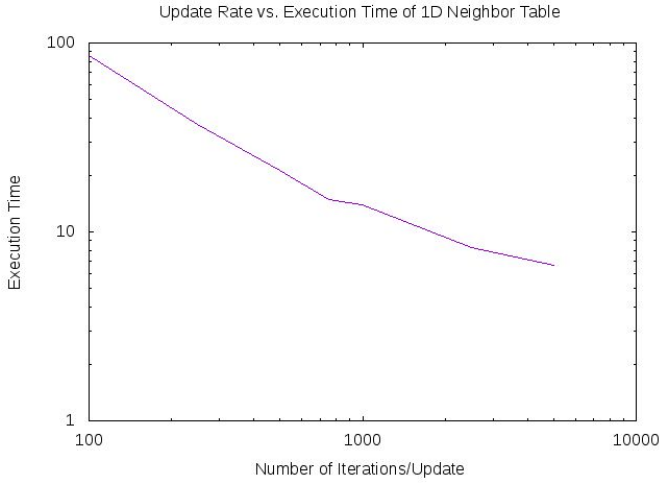




Basic Method vs. Neighbor Table

- Time step range: 1E-5 to 1E-12
- Maximum Iterations: 5000
- Table Update every 100 cycles
- Neighbor Table overkill at low particle density
- 16384 particles require 83 seconds
    - Improved with larger update interval
- Can we do anything else?





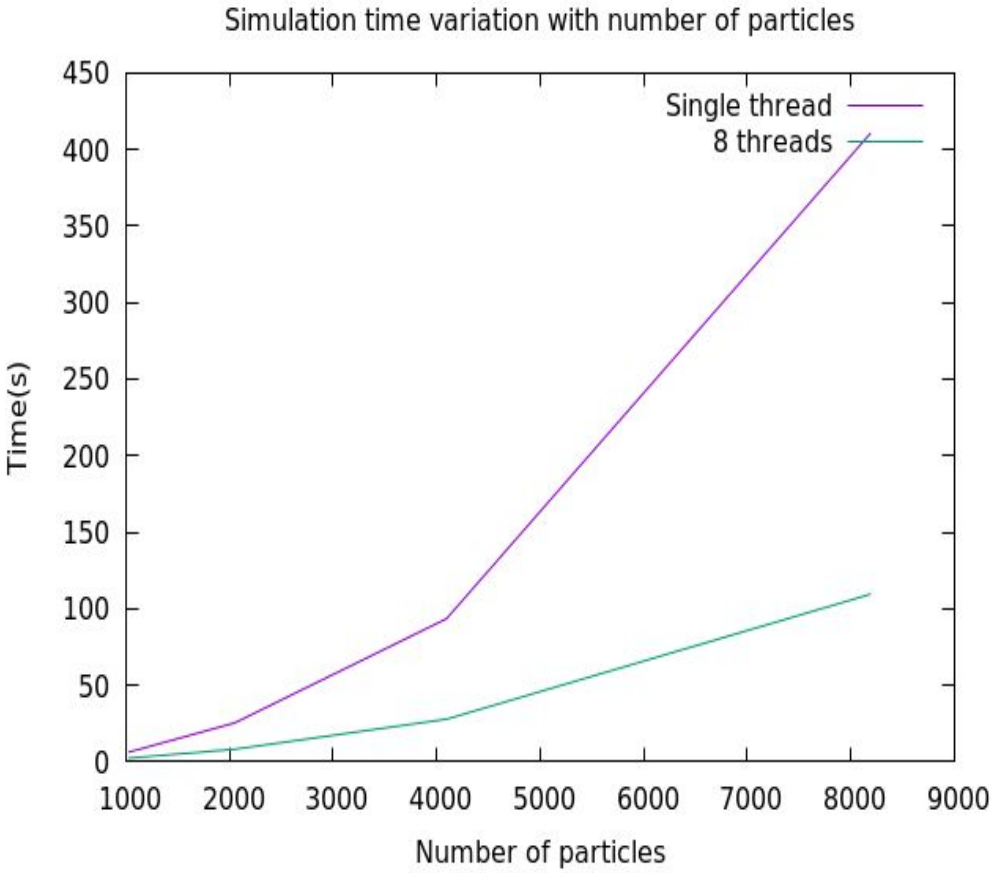
Improvements through parallelization

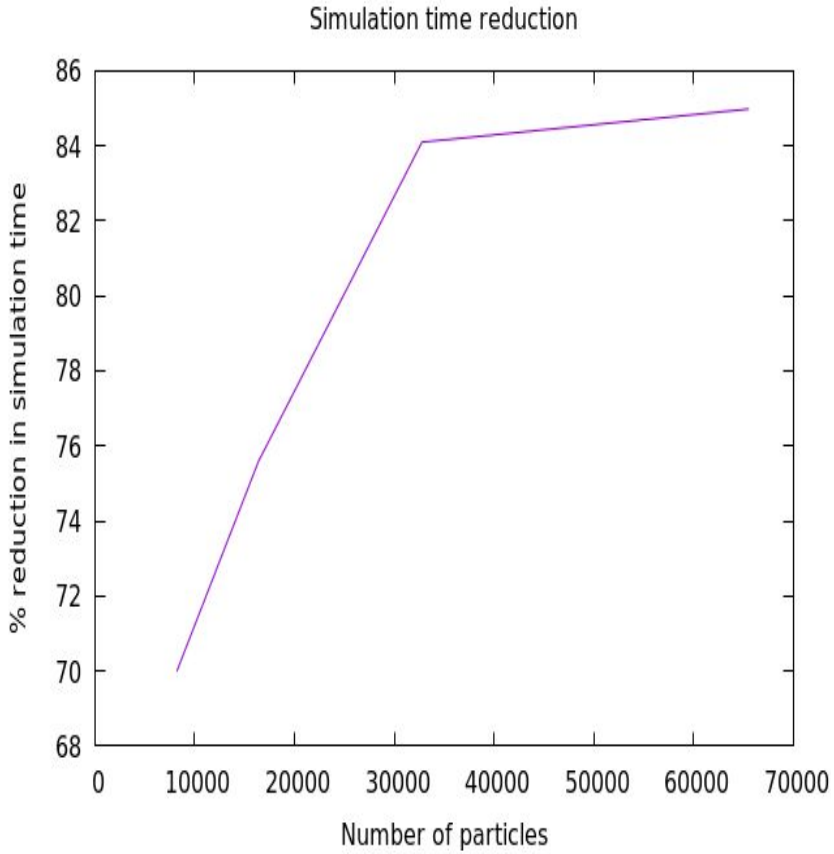
- $O(N^2)$ calculations result in lower performance.
- Force calculations in basic implementation.
- Neighbor map generation in neighbor map based implementation.
- Parallel execution using OpenMP.
- Parallelising nested ‘for’ loops which go through all the particles gives the largest performance improvement.

Force calculation in basic implementation

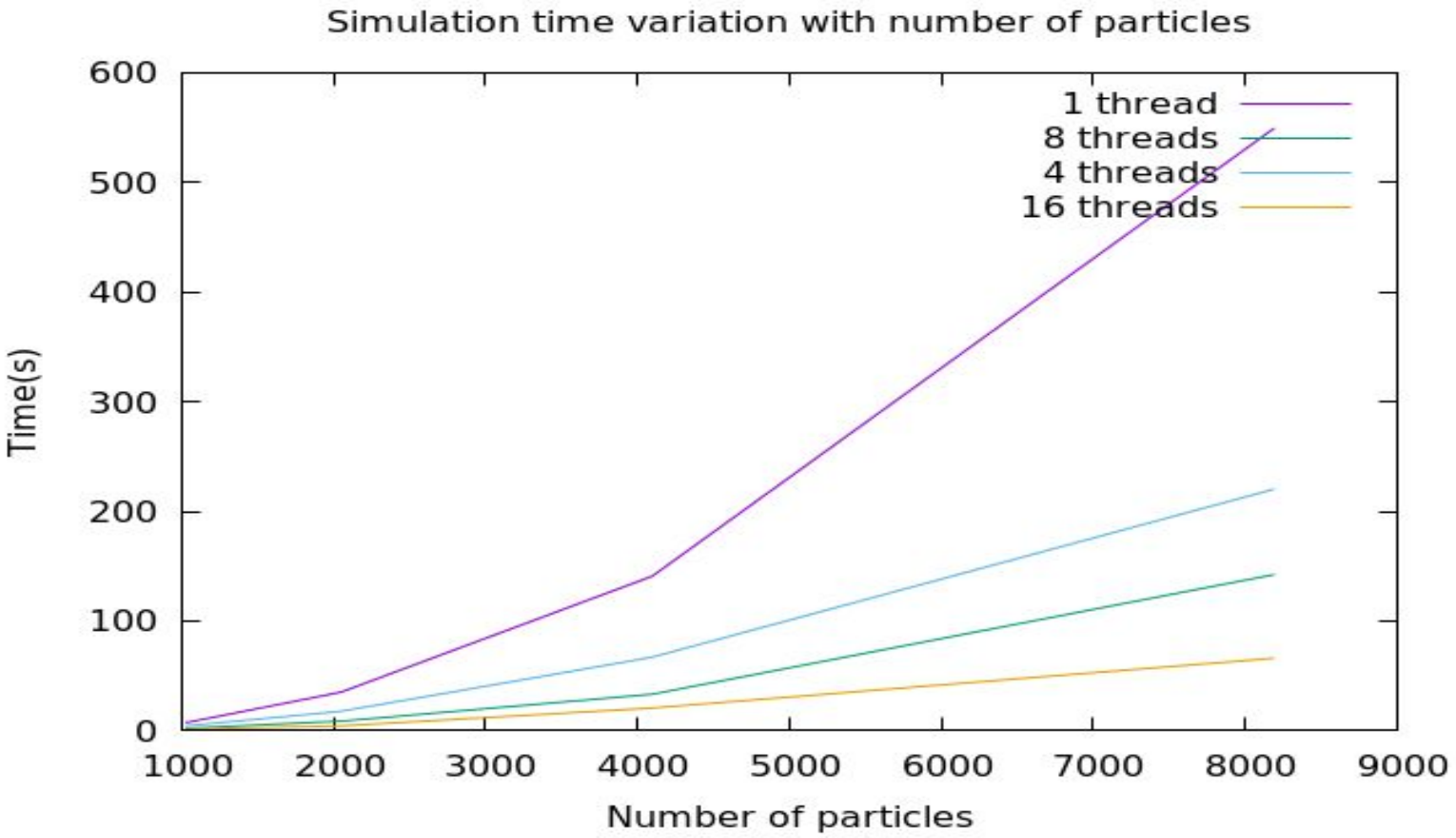
```
// Perform force & Epot calculation based on proximity
#pragma omp parallel for private(j,dim,r2, r2_over, r6_over,fp,finst,dist_diff,in_range) reduction(+:Epot)
for (i = 0; i < np; i++) {
    for (j = i+1; j < np; j++) {
        r2 = 0.0;
        in_range = true;
        for (dim = 0; dim < nd; dim++) {
            dist_diff[dim] = pos[dim+i*nd] - pos[dim+j*nd];
            // Minimum Image Criterion
            if (abs(dist_diff[dim]) > 0.5*L) { dist_diff[dim] -= sign(L, dist_diff[dim]); }
            if (dist_diff[dim] > R_cut) { in_range = false; }
            r2 += dist_diff[dim] * dist_diff[dim];
        }
        if (in_range) {
            r2_over = 1.0/r2;
            r6_over = r2_over*r2_over*r2_over;
            fp = 48.0*r6_over*(r6_over-0.5)*r2_over;
            for (dim = 0; dim < nd && in_range; dim++) {
                finst[dim] = fp * dist_diff[dim];
                acc[dim+i*nd] += finst[dim]/mass;
                acc[dim+j*nd] -= finst[dim]/mass;
            }
            Epot += 4.0*(r6_over*r6_over-r6_over);
        }
    }
}
```

Results (Basic implementation)





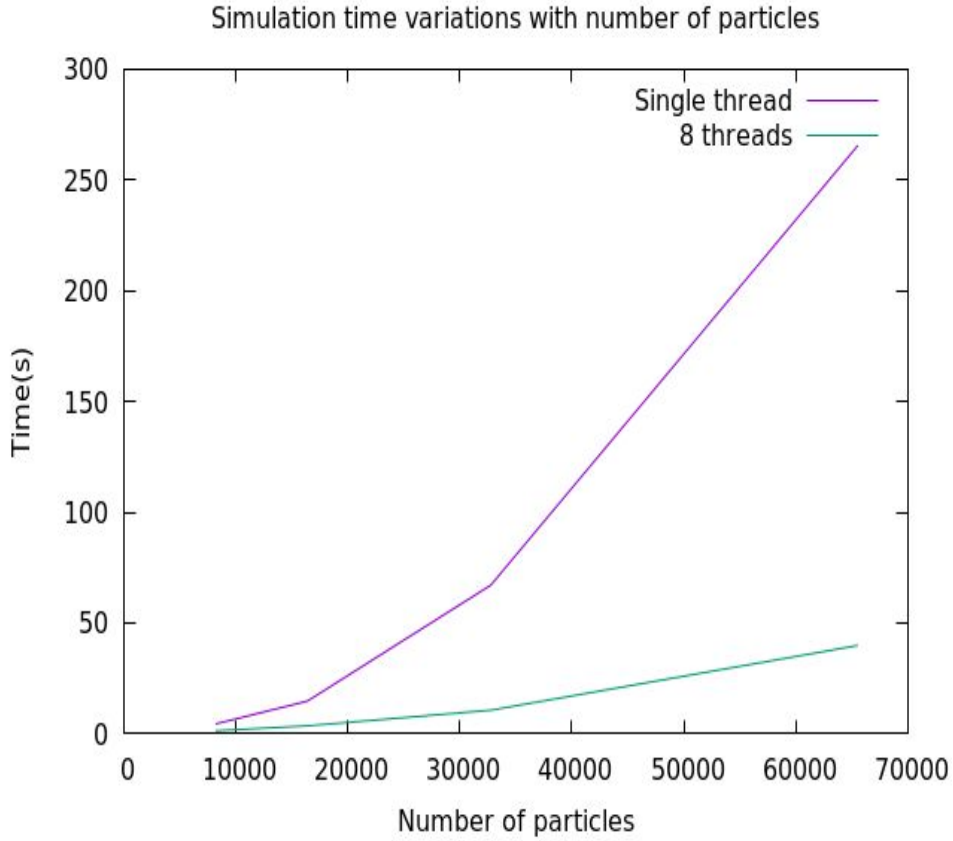
2D simulation

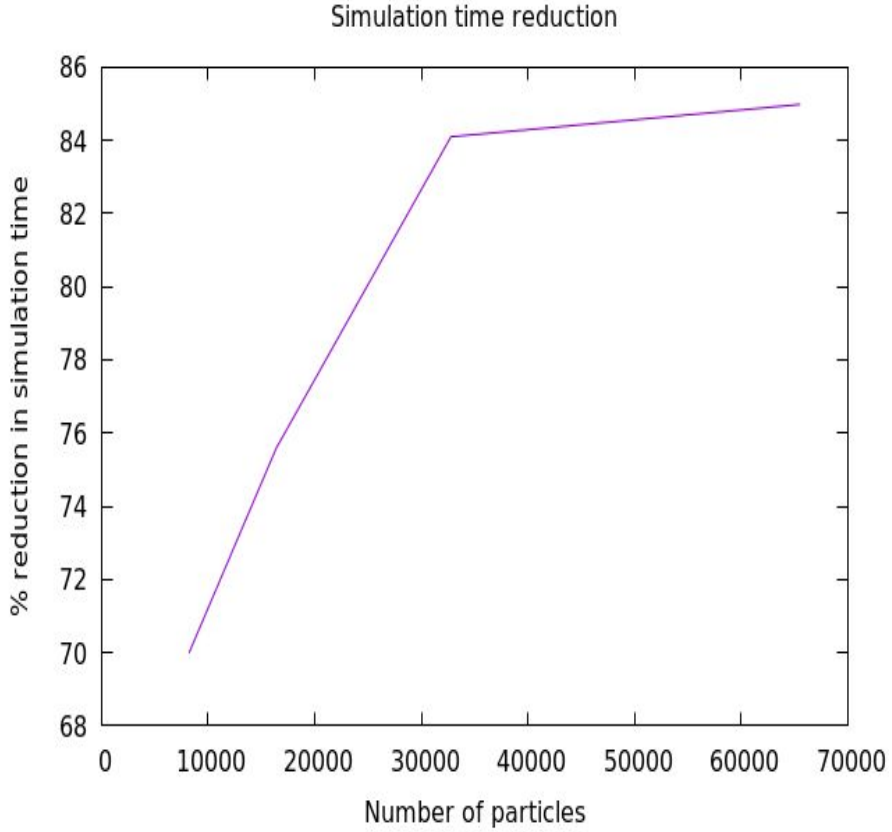


Neighbor map generation

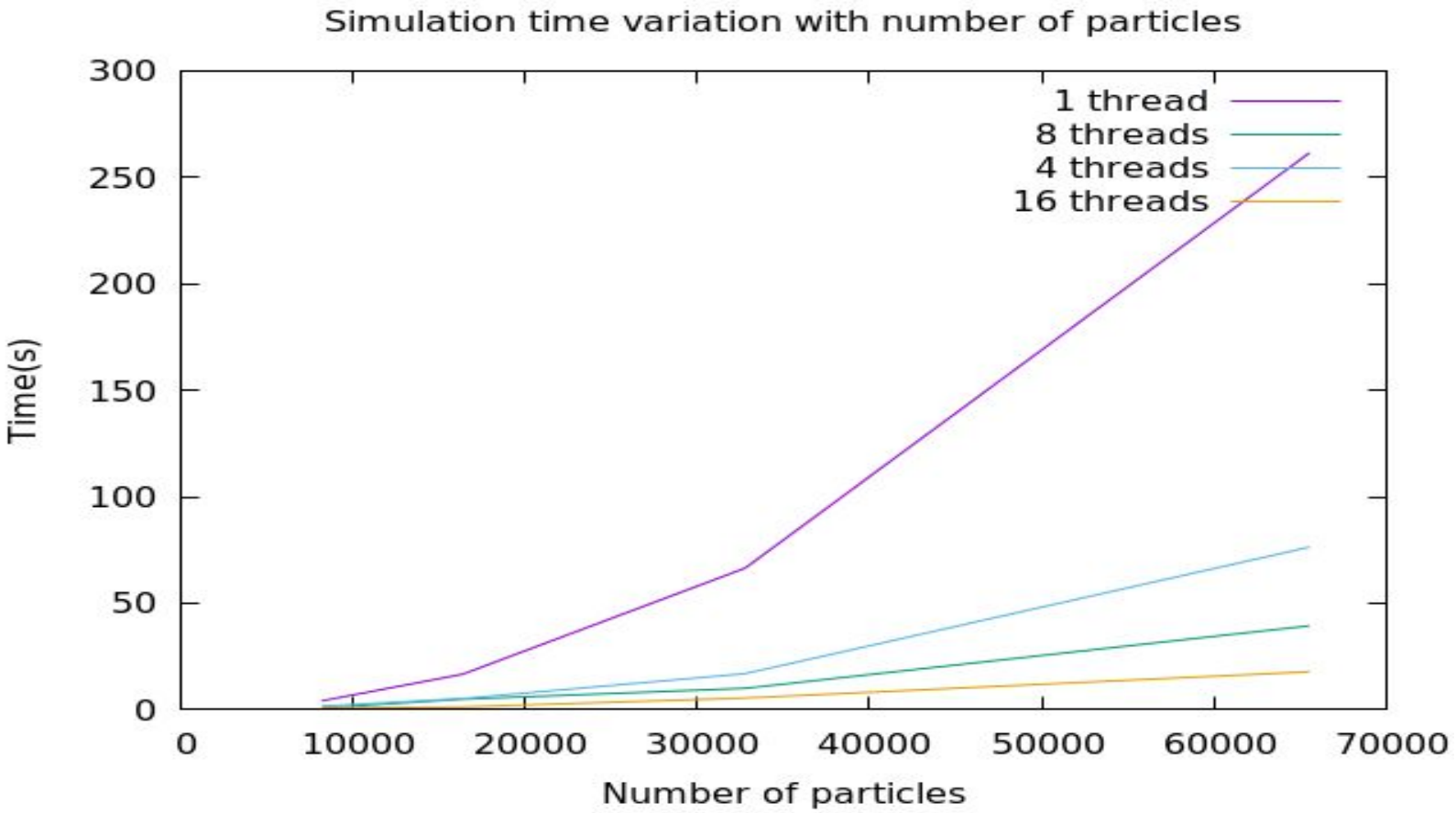
```
    #pragma omp parallel for private(j,dim,diff_dist,in_range)
    for (i = 0; i < np; i++) {
        for (j = 0; j < np; j++) {
//        for(j = i+1; j < np; j++) {
            if (j == i) { continue; } // Can't pair the particle with itself
            else { // Check distances
                for (dim = 0; dim < nd; dim++) {
                    diff_dist = pos[dim+i*nd] - pos[dim+j*nd];
                    if (abs(diff_dist) > 0.5*L) { diff_dist -= sign(L, diff_dist); }
                    in_range = (abs(diff_dist) <= R_skin);
                    if (!in_range) { break; }
                }
                if (in_range) { neighbor[i].push_back(j); }
            }
        }
    }
```

Neighbor map based implementation





2D simulation



Summary

- $O(N^2)$ calculation forces basic simulation size to less than ~500 particles to be effective
- Improvements can be made WITHOUT parallelization (but together would be the best)
  - Neighbor Table implementation and update interval
- Cell Linked List Algorithm analysis and integration next objective.