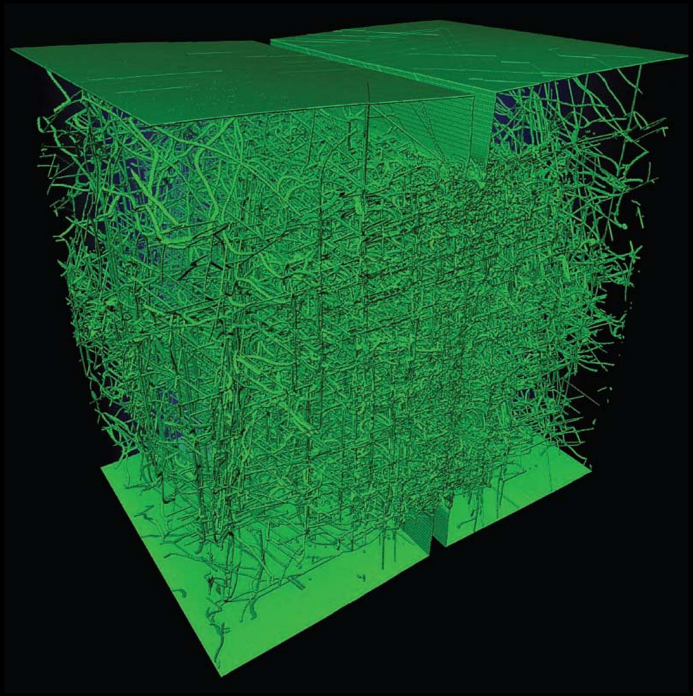
05/13/2009 Final Report

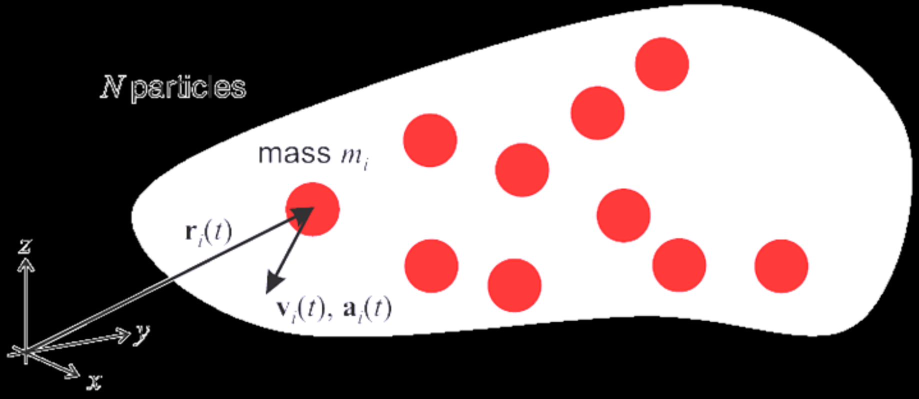
Parallel Molecular Dynamics Algorithms



Anna Nikiforova

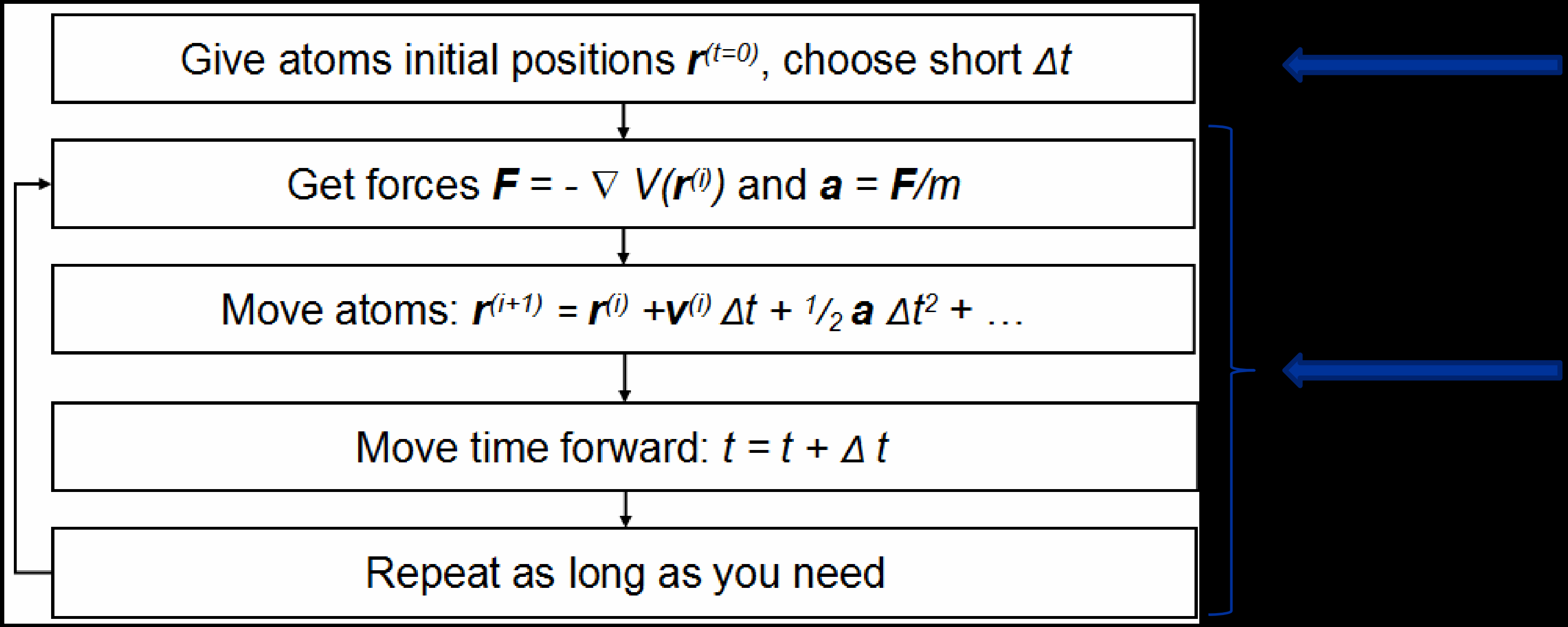
MD

- Atomistic simulation method where:
  - Each atom is treated as a point mass
  - Physics is described by a potential energy functional
  - Newton's equations are integrated to advance the atomic positions & velocities
  - Thermodynamic statistics are extracted from the motion of the atoms
- We follow the evolution of a system composed of many classical particles



What is MD?

- Computationally intensive in the number of atoms and number of time steps



Give atoms initial positions $\boldsymbol{r}^{(t=0)}$, choose short Δt

Get forces $\boldsymbol{F} = -\nabla V(\boldsymbol{r}^{(i)})$ and $\boldsymbol{a} = \boldsymbol{F}/m$

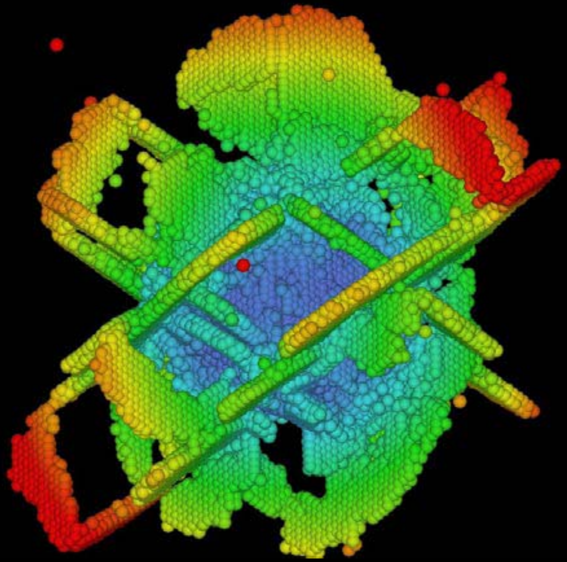
Move atoms: $\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} + \boldsymbol{v}^{(i)} \Delta t + {}^{1}/_{2}\, \boldsymbol{a}\, \Delta t^2 + \ldots$

Move time forward: $t = t + \Delta t$

Repeat as long as you need

Why go parallel?

- Large number of atoms is required in order to describe a system with sufficient correctness
  - Example: crack propagation in a material or grain boundary
  - Millions of atoms are needed to simulate even a micron scale
- Large number of time steps required
  - The time step size is generally limited to a femtosecond by atomic vibration frequency



J. Moriarty et al., *Journal of Physics: Condensed Matter*, **14**, pp. 2825-2857 (2002)

What’s used?

- Atom Decomposition Method
  - Split the N atoms into P processors
  - Each processor is responsible for its set of atoms
  - No atom “migration” between the processors happens
- Force Decomposition Method
  - computation of force loops rather than the atoms is performed in parallel
- Spatial Decomposition Method
  - The system is divided into the simulation domains
  - Each processor is assigned a domain and computes the forces and updates the positions and velocities of the atoms in its domain

Comparison

	Atom decomposition	Force decomposition	Spatial decomposition
Method overview	o Split N atoms into P processors permanently o Atom are not allowed to migrate between processors	o Split the matrix representing all of the combinations of atoms into blocks o Atoms are not tied to a particular domain	o Divide the system into the simulation domains o Atoms are allowed to move freely between the processors
Communication requirements	(1) Update of positions and velocities (2) Report of forces to obtain total force on each atom	(1) The MD computations are divided between the processors evenly (2) Block decomposition of the force matrix requires order of information (3) No geometric info is needed (not communicated)	(1) Communication of atoms' positions to processors with neighboring cells (2) Periodical update of atoms that left processor's box to the appropriate processor

Performance on 1 processor

- How does the method affect the computational time vs. system setup?
  - Number of atoms
  - Density
  - Initial temperature
- What effect does binning of atoms have on the computational speed?
  - System size
  - Choice of parallel algorithm

Case 1. System setup

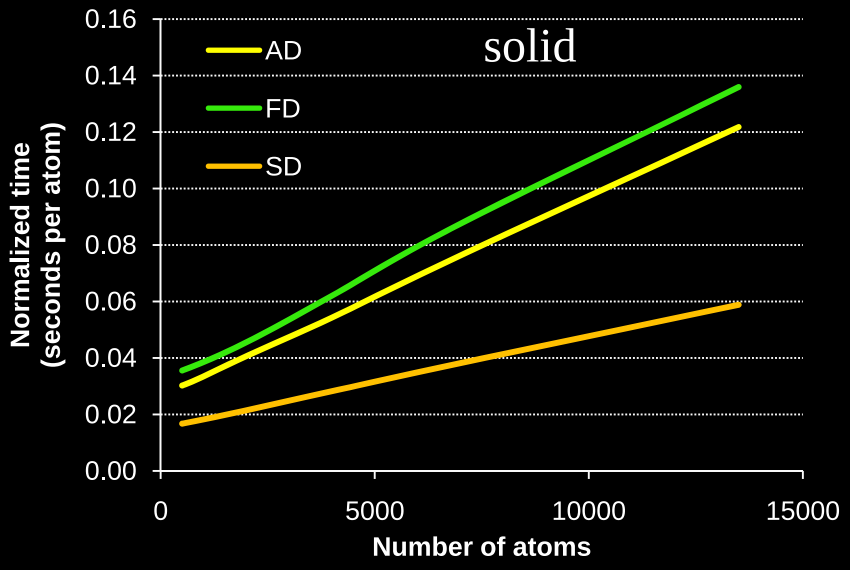
$$\left.\begin{array}{l}\blacksquare\ \text{TR} = 1.2 \\ \blacksquare\ \text{DR} = 0.6\end{array}\right\}\ \text{liquid}$$

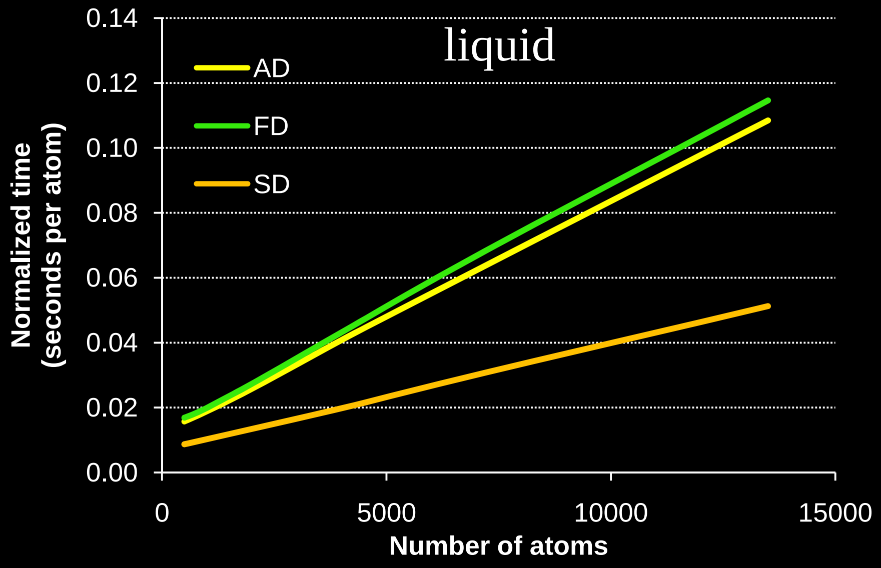
$$\left.\begin{array}{l}\blacksquare\ \text{TR} = 0.4 \\ \blacksquare\ \text{DR} = 1.2\end{array}\right\}\ \text{solid}$$

- Reduced time step = 0.00001
- Potential cutoff = 2.5
- # time steps = 10000
- FCC lattice
- How does the choice of method affect the computational time vs. number of atoms?

Case 1. Results

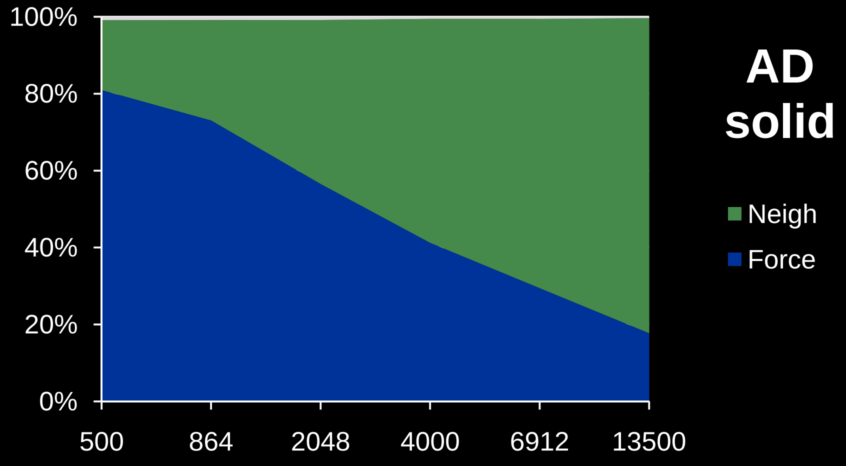
- Solid is slower than liquid system
- Possible explanation (next slide)

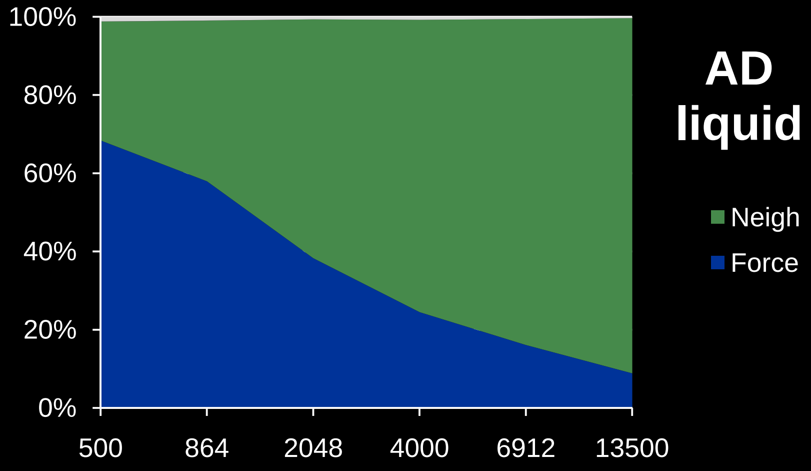




Case 1. Results

- Higher fraction of time spent on calculating forces
- Density of the solid system is higher -> larger numbers of atoms fall under the cutoff radius
- Neighboring for solid system is simpler since all of the atoms are less mobile than in liquid
- AD, FD, and SD produced nearly the same fraction plots

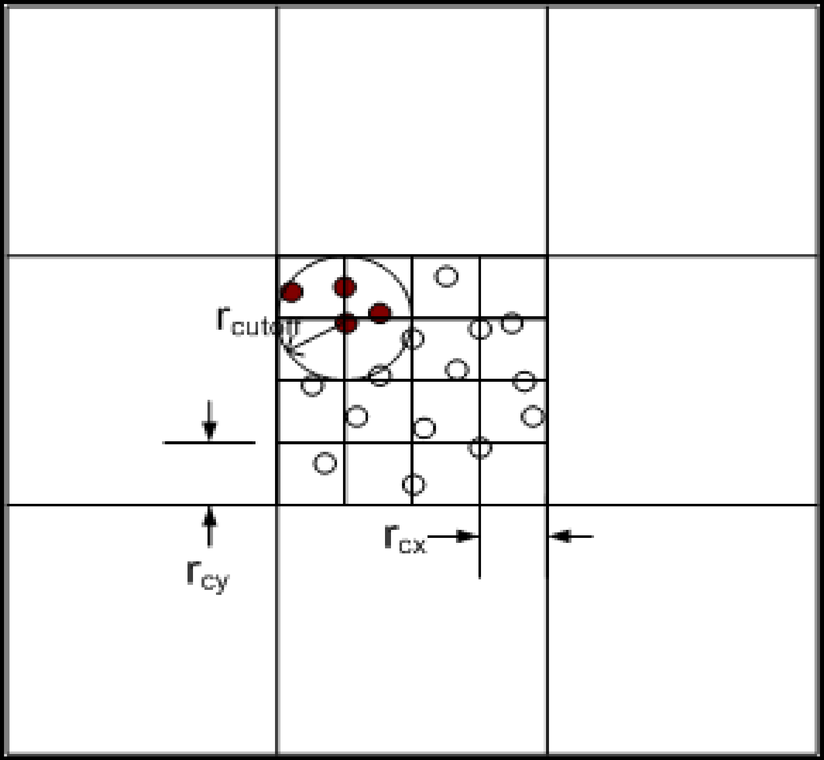




Case 2. Binning

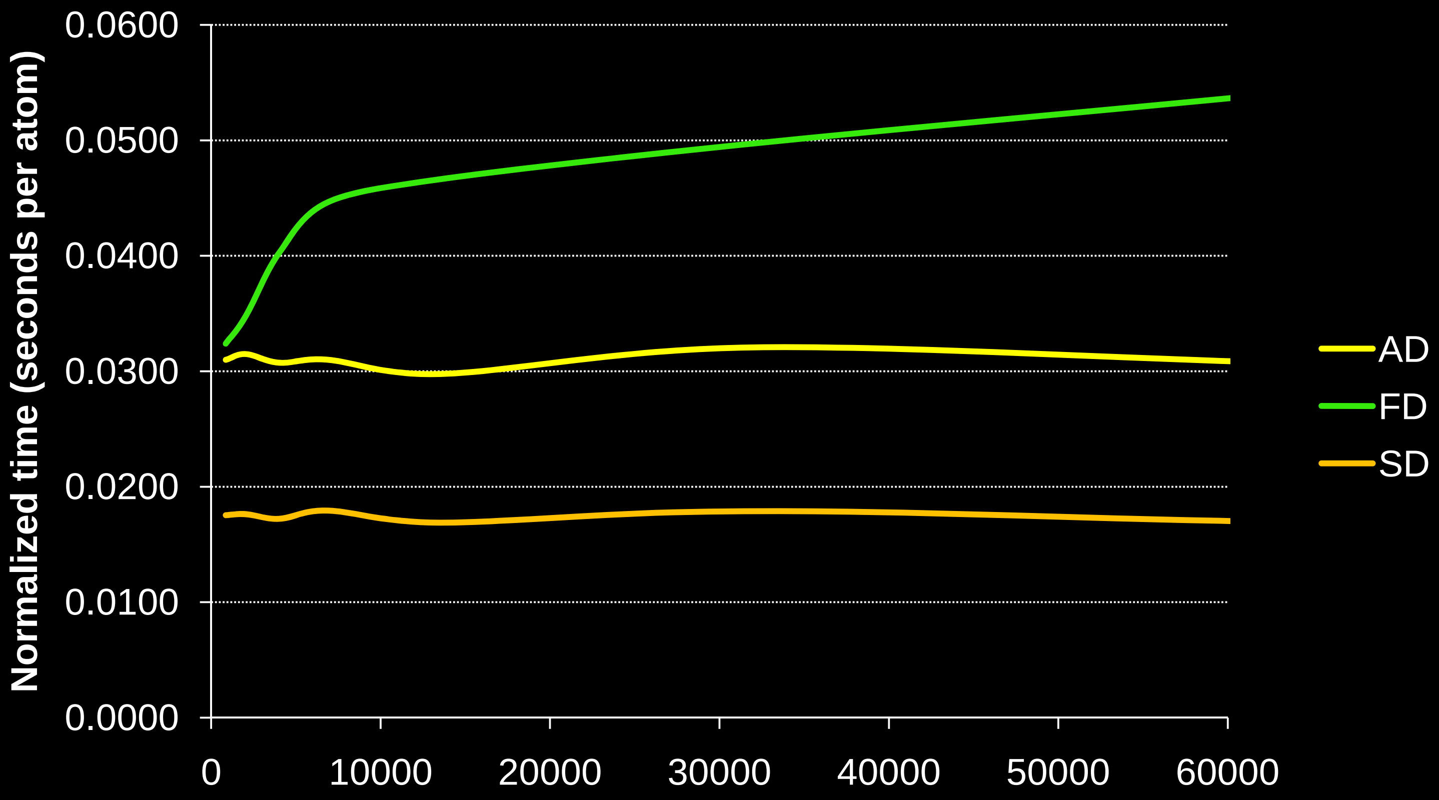
- Interactions between the particles = pairwise additive
- The time needed for the force evaluation scales as O(N^2)

- Binning = divide system into cells with size equal or smaller than r_{cutoff} then each particle will interact only with particles in the same cell, and the evaluation scaling will be reduced from O(N^2) to O(N)



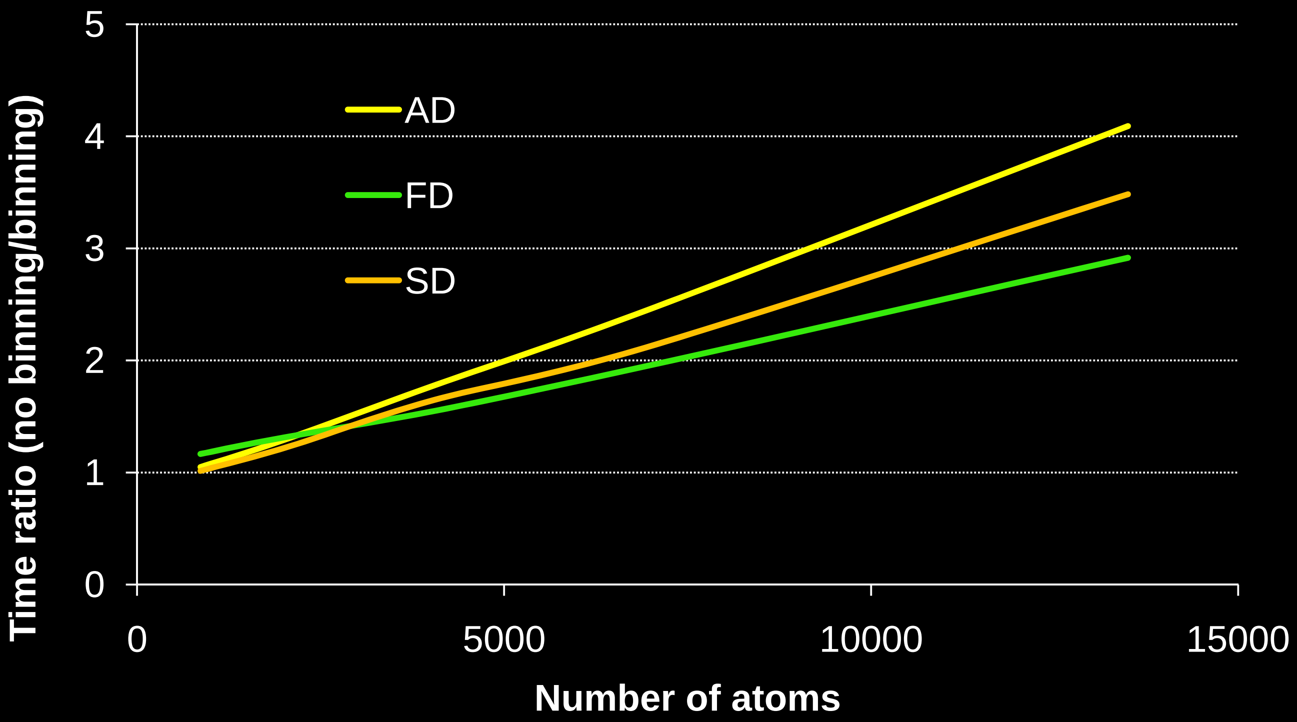
Case 2. Binning

- Normalized time is constant -> O(N) behavior



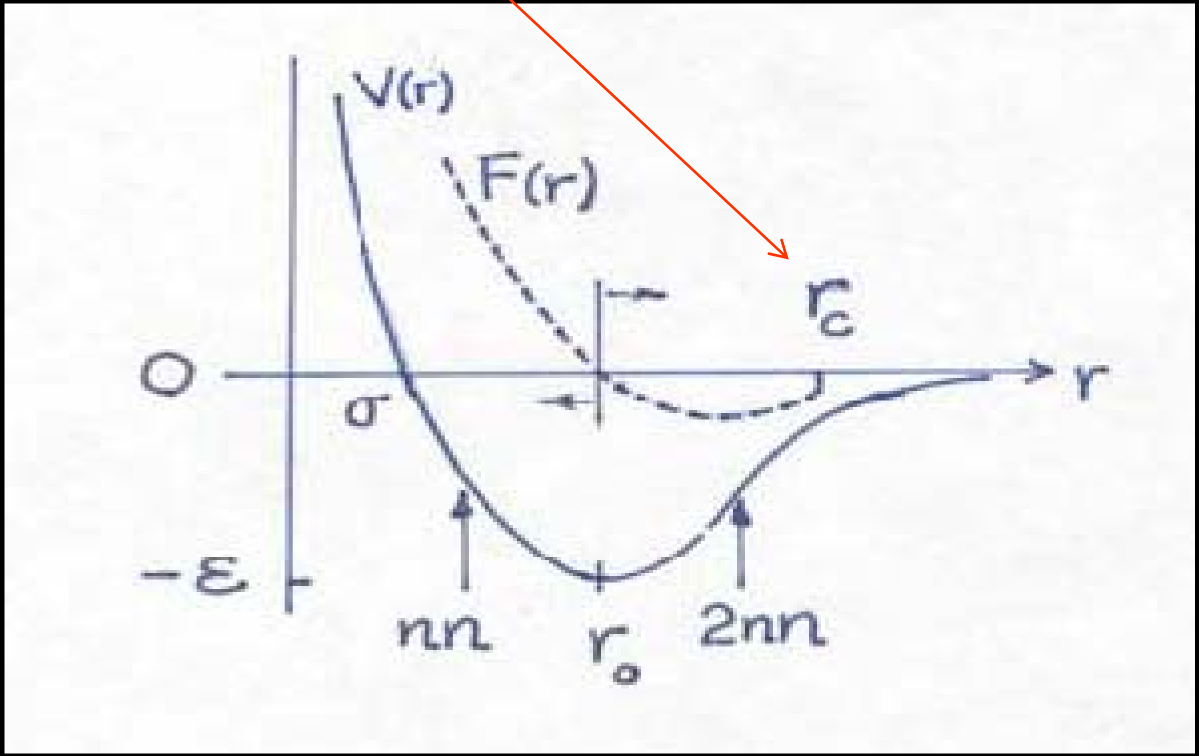
Case 2. Binning

- Binning can speed up the calculation especially as the system size increases
- Especially effecting in Atom Decomposition method
- Ratio of time spent per calculation of “no binning” to “binning”:



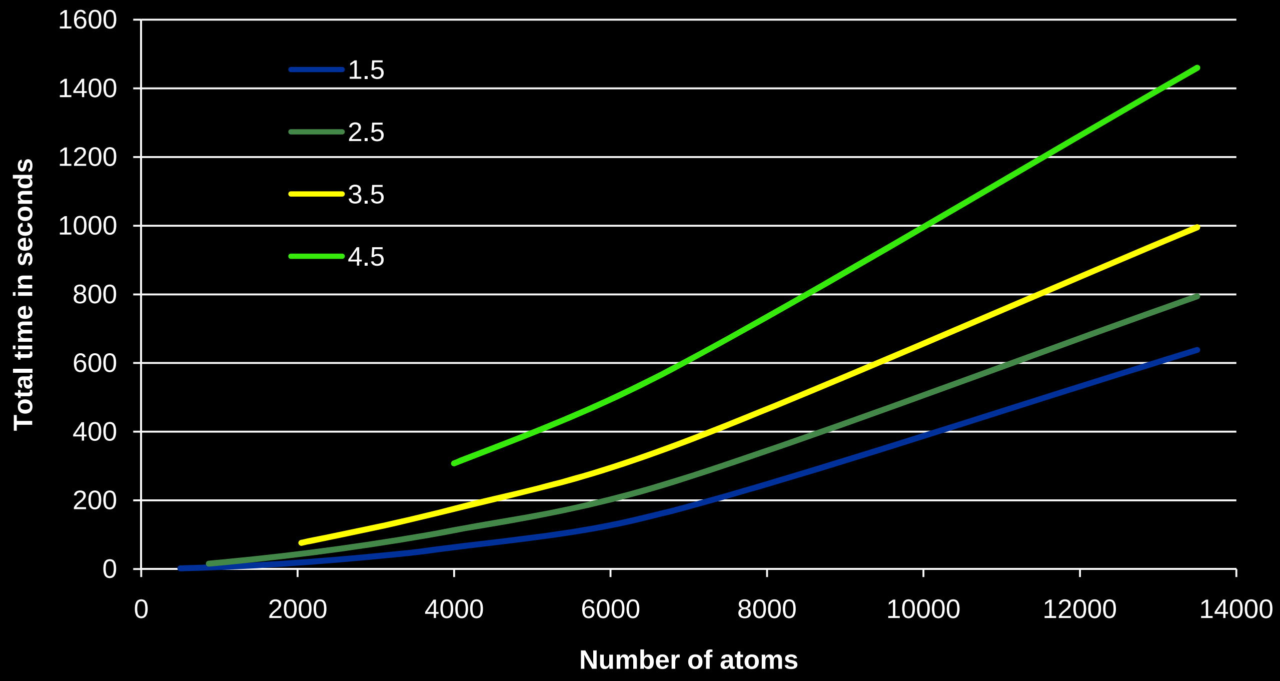
Case 3. Cutoff radius

- Cutoff distance r_{cutoff} beyond which the potential and force are approximated by zero



Case 3. Cutoff radius

- Choice of cutoff radius affects computational time
- Balance between how accurate the results vs. how long it takes to obtain them
- This results are for SD method applied to solid system



Performance on multi proc

- How do the methods behave on different number of processors?
  - 8 processors (2x Quad Core Intel® Xeon® Processor X5482 (3.20GHz,2X6M L2,1600))
  - 16GB, DDR2 SDRAM FBD Memory, 667MHz, ECC (4 DIMMS)
  - Red Hat Enterprise Linux 5 operating system

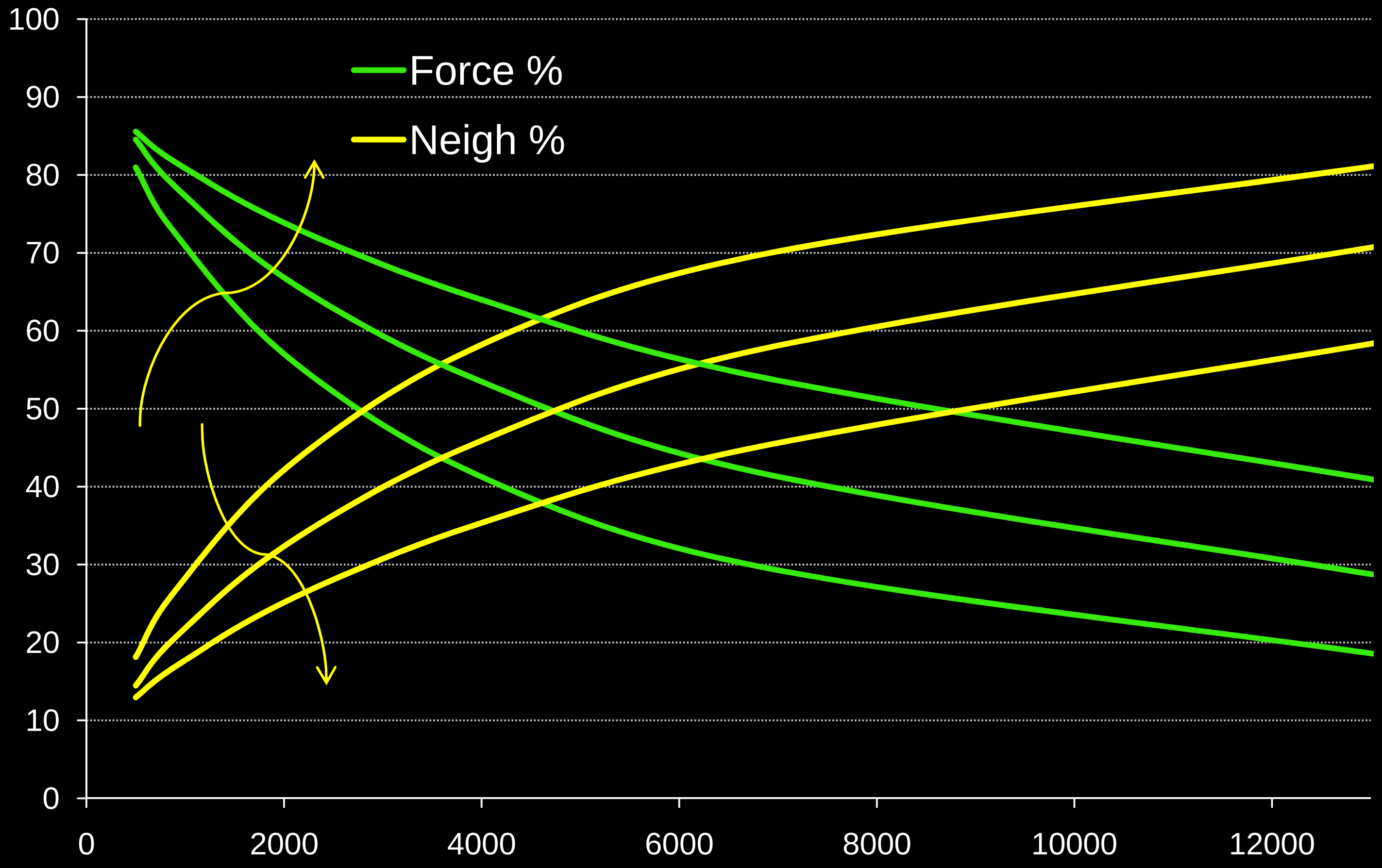
Total time

- Speed up = t_1/t_N
- More than ideal speed up? AD physics seems correct.

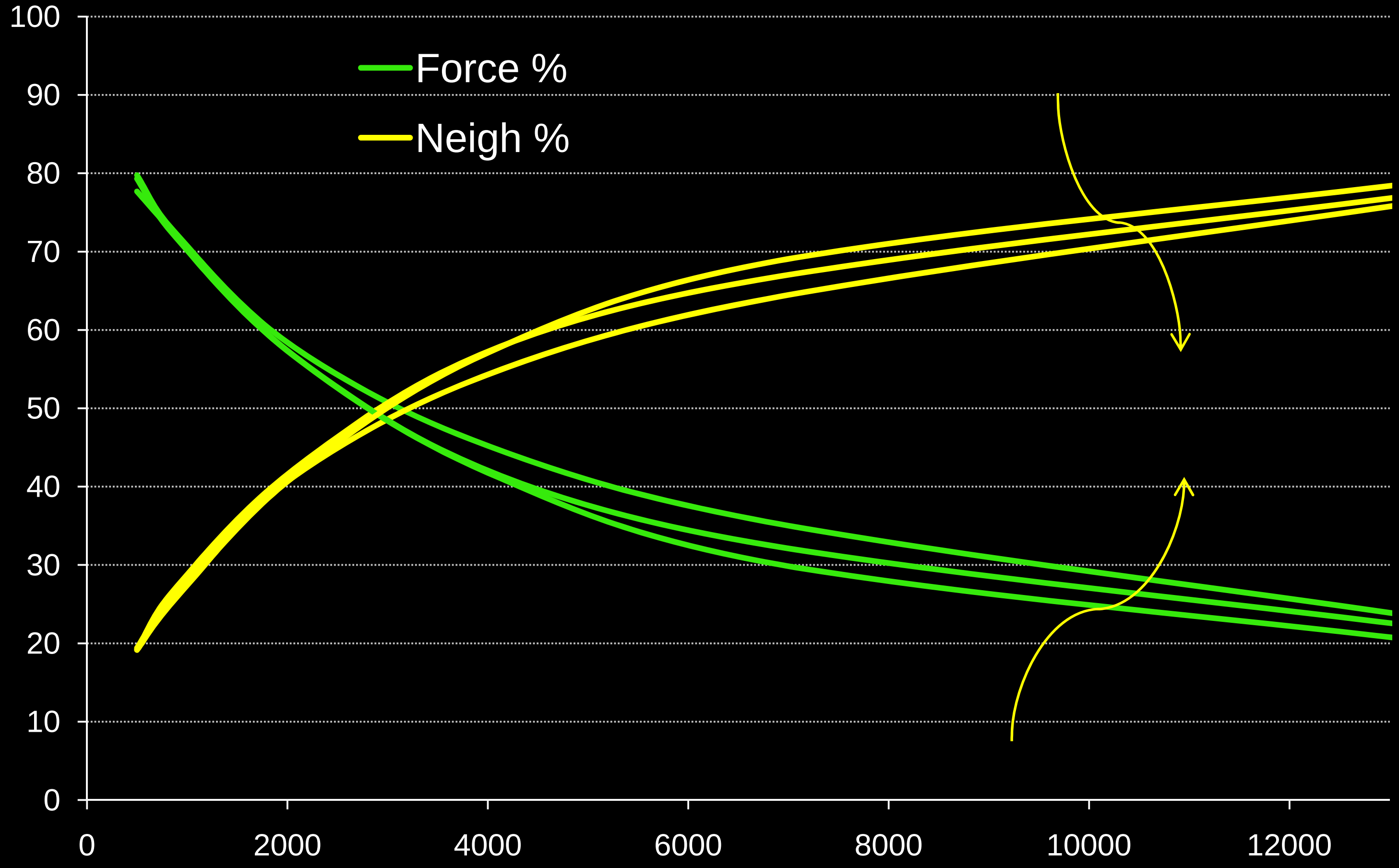
	1 proc			2 proc			4 proc		
	AD	FD	SD	AD	FD	SD	AD	FD	SD
500	1	1	1	2.9	4.0	3.7	8.8	16.6	13.3
864	1	1	1	2.8	4.1	3.4	8.2	16.3	10.9
2048	1	1	1	2.9	4.1	3.3	8.0	16.5	10.5
4000	1	1	1	3.1	4.3	3.3	8.6	16.9	10.4
6912	1	1	1	3.3	4.2	3.6	9.7	17.4	11.1

AD

- Arrow indicated the increase in # of processors (1, 2, 4)

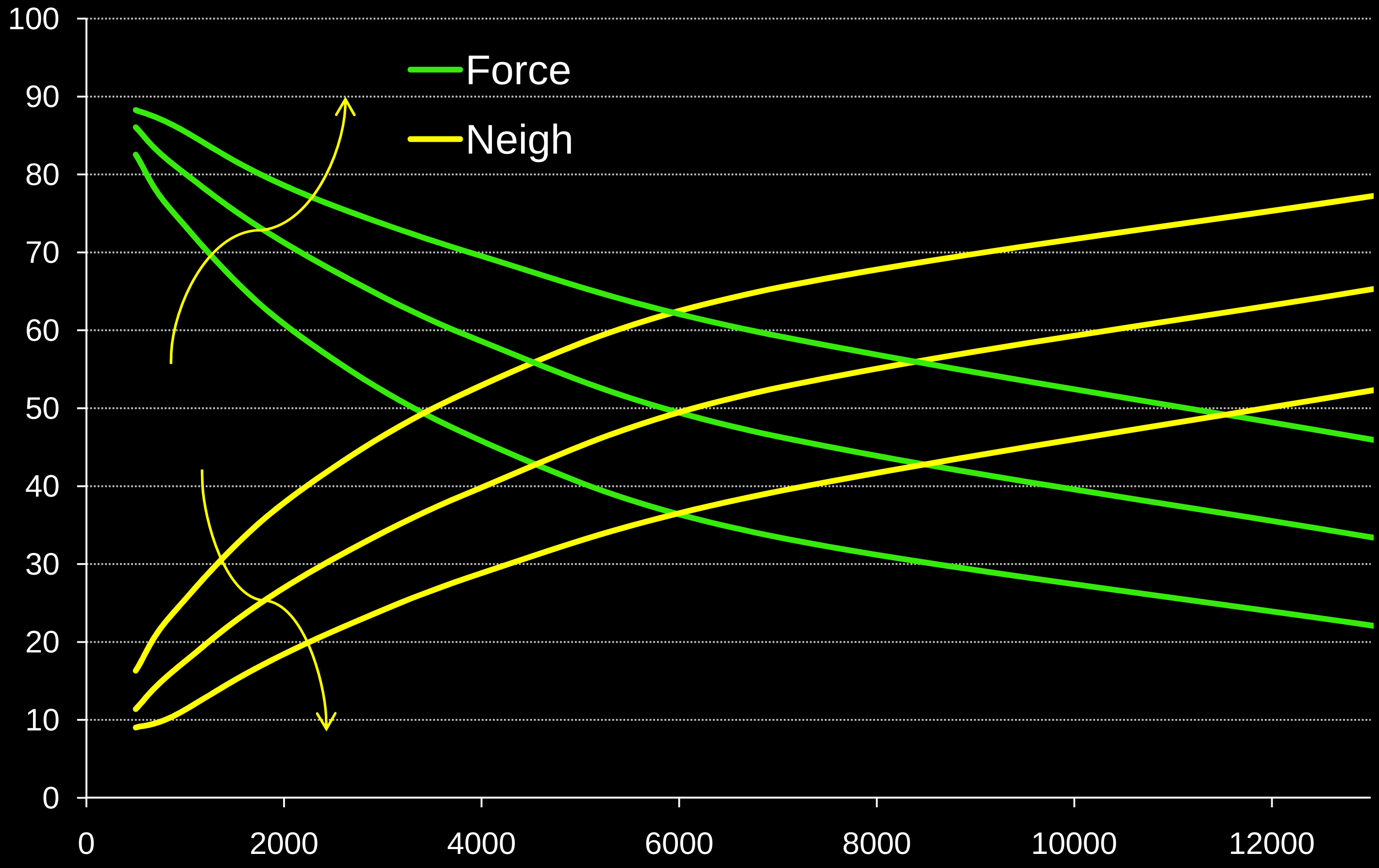


FD



100
90
80
70
60
50
40
30
20
10
0
0
2000
4000
6000
8000
10000
12000
Force %
Neigh %

SD


Force
Neigh
100
90
80
70
60
50
40
30
20
10
0
0
2000
4000
6000
8000
10000
12000

But physics for FD and SD is inconsistent!

Average values for solid system (1 proc)

(reduced temperature, reduced potential energy, reduced pressure)

	AD			FD			SD		
	T*	U*	P*	T*	U*	P*	T*	U*	P*
500	0.390	-8.236	10.943	0.390	-8.236	10.943	0.390	-8.236	10.943
864	0.390	-8.236	10.944	0.390	-8.236	10.944	0.390	-8.236	10.944
2048	0.390	-8.238	10.944	0.390	-8.238	10.944	0.390	-8.236	10.944
4000	0.390	-8.240	10.932	0.390	-8.240	10.932	0.390	-8.239	10.941
6912	0.390	-8.240	10.968	0.390	-8.235	10.977	0.390	-8.241	10.960
13500	0.390	-8.235	10.978	0.390	-8.235	10.977	0.390	-8.241	10.960

Average values for solid system (2 proc)

(reduced temperature, reduced potential energy, reduced pressure)

	AD			FD			SD		
	T*	U*	P*	T*	U*	P*	T*	U*	P*
500	0.392	-6.986	9.888	0.394	-2.541	1.952	0.356	-3.387	3.784
864	0.393	-7.197	10.045	0.398	-2.556	1.900	0.359	-3.588	4.033
2048	0.391	-7.454	10.298	0.396	-2.538	1.937	0.367	-3.801	4.256
4000	0.392	-7.614	10.412	0.396	-2.528	1.913	0.369	-3.912	4.369
6912	0.391	-7.717	10.509	0.396	-2.544	1.981	0.372	-3.999	4.471
13500	0.391	-7.824	10.610	0.396	-2.537	1.941	0.380	-4.124	4.638

What happened to FD and SD?

- AD shows consistent behavior
- FD and SD are off
  - Error in parallelization?
  - Improper atom migration from processor to processors?
  - Future work

Conclusions

- Three methods for parallelization of MD
  - atom decomposition (atoms are separated into different processors and fixed),
  - force decomposition (block decomposition of the force matrix),
  - spatial decomposition (domains are fixed)
- AD method does not perform well when communication is significant (recall that neighboring time increases as the number of processors increases).
- FD method is slower than AD when executed on one processor, but faster if more than 1 processor is used.
- SD is always the fastest algorithm, but does not scale with the number of processors as well as FD. From this, we can conclude that FD will become better than SD once certain number of processors is used.

Conclusions

- Dependence on what system type is used as well as if computational time savers are used:
  - Binning
  - force cutoff.
- SD algorithm always showed the best performance for testing on one processor.
- Binning was found to be very effective in reducing the total computational time and making system to behave as O(N) rather than O(N^2).