NAMD - Scalable Molecular Dynamics

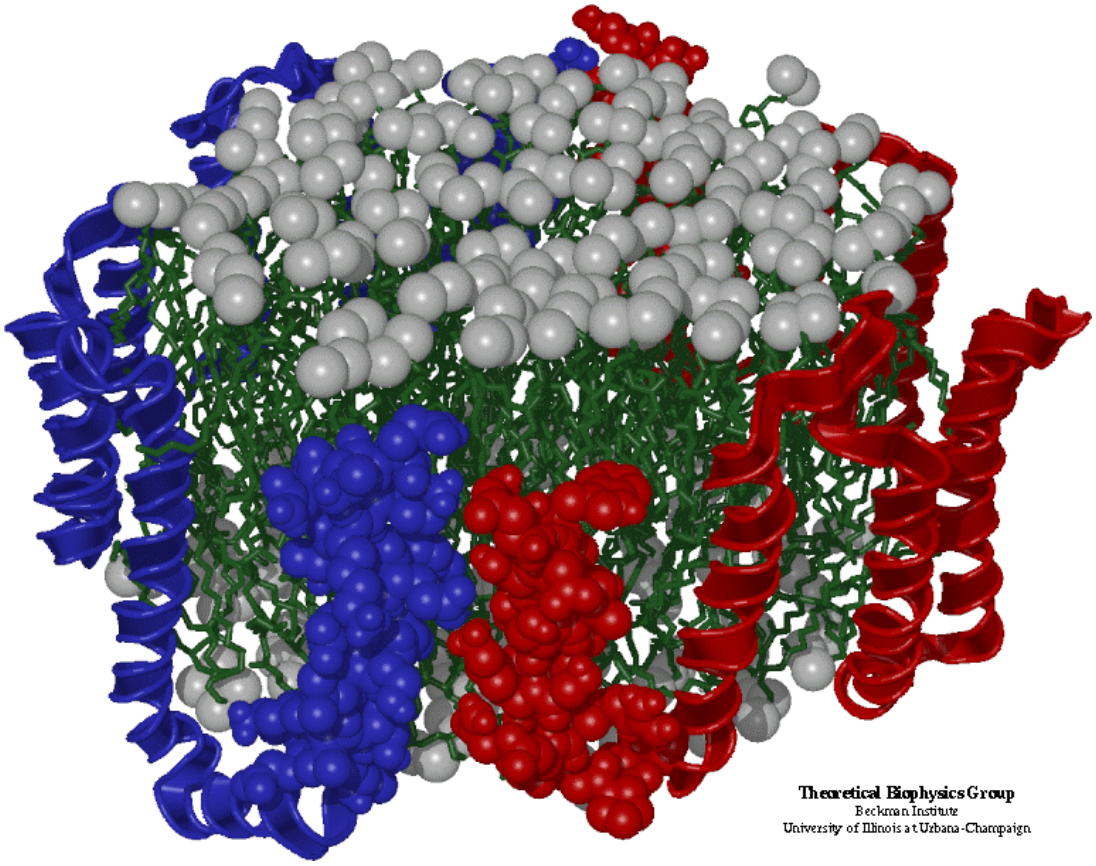
Gengbin Zheng

9/1/01

Molecular dynamics and NAMD

- MD to understand the structure and function of biomolecules
  - proteins, DNA, membranes
- NAMD is a production quality MD program
  - Active use by biophysicists (science publications)
  - 50,000+ lines of C++ code
  - 1000+ registered users
  - Features and “accessories” such as
    - VMD: visualization and analysis
    - BioCoRE: collaboratory
    - Steered and Interactive Molecular Dynamics

Molecular Dynamics



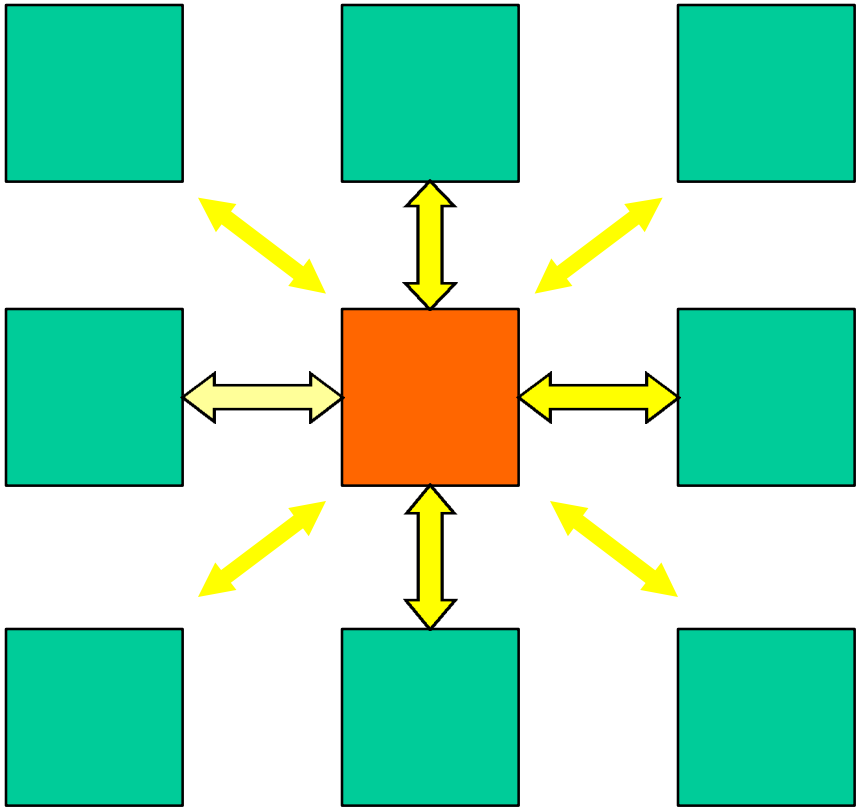
Molecular Dynamics

- Collection of [charged] atoms, with bonds
- Like N-Body problem, but much complicated.
- At each time-step
  - Calculate forces on each atom
    - non-bonded: electrostatic and van der Waal's
    - Bonds(2), angle(3) and dihedral(4)
  - Integration: calculate velocities and advance positions
- 1 femtosecond time-step, millions needed!
- Thousands of atoms (1,000 - 100,000)

Cut-off radius

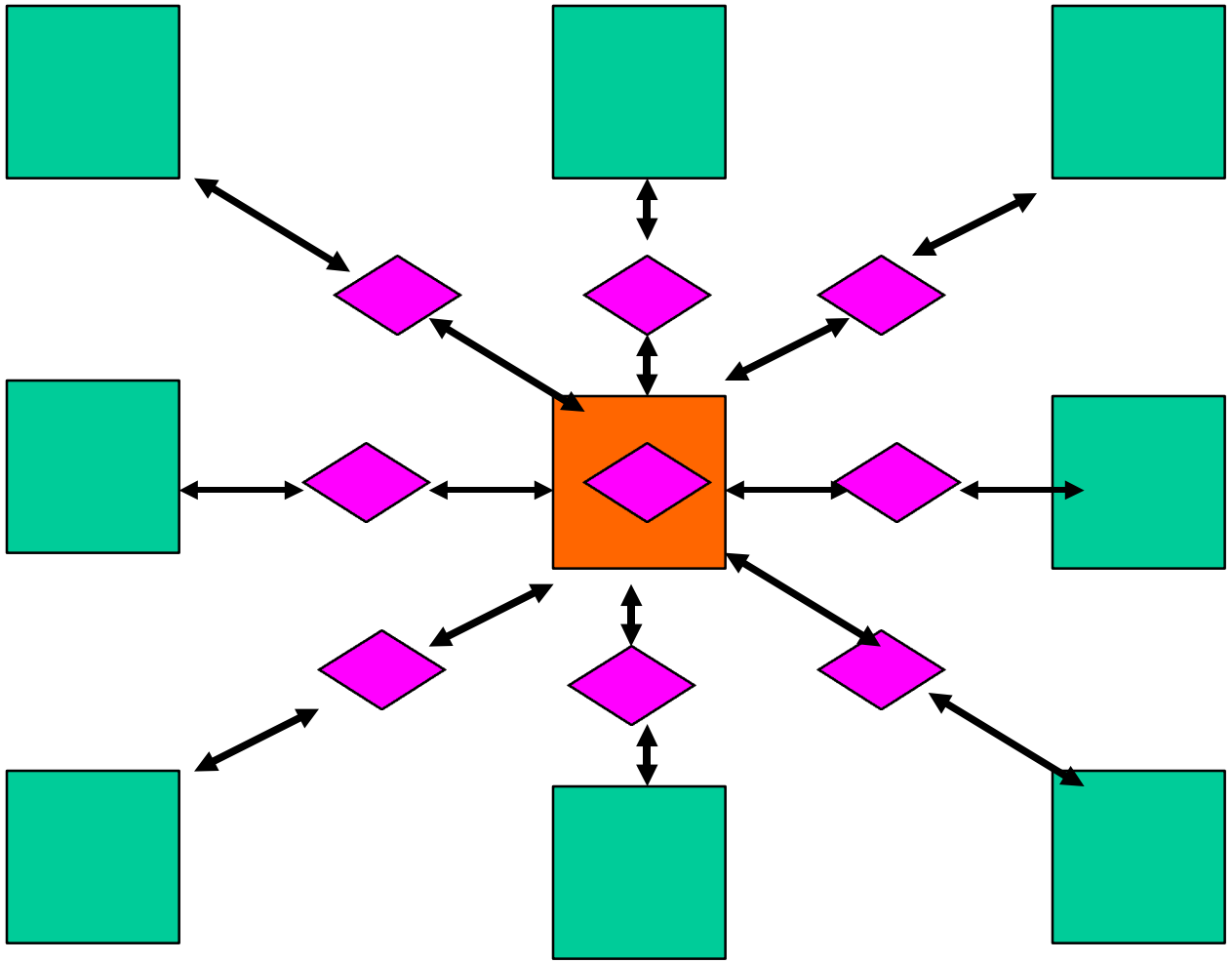
- Use of cut-off radius to reduce work
  - 8 - 14 Å
  - Far away charges ignored!
- 80-95 % work is non-bonded force computations
- Some simulations need far away contributions
  - Periodic systems: Ewald, Particle-Mesh Ewald
  - Aperiodic systems: FMA
- Even so, cut-off based computations are important:
  - near-atom calculations are part of the above
  - Cycles: multiple time-stepping is used: k cut-off steps, 1 PME/FMA

Spatial Decomposition



Patch

But the load balancing problems are still severe:



Patch

Compute

Proxy

FD + SD

- Now, we have many more objects to load balance:
  - Each diamond can be assigned to any processor
  - Number of diamonds (3D):
    - 14·Number of Patches

Load Balancing

- Is a major challenge for this application
  - especially for a large number of processors
- Unpredictable workloads
  - Each diamond (force object) and patch encapsulate variable amount of work
  - Static estimates are inaccurate
- Measurement based Load Balancing Framework
  - Robert Brunner's recent Ph.D. thesis
  - Very slow variations across timesteps

Load Balancing

- Based on migratable objects
- Collect timing data for several cycles
- Run heuristic load balancer
  - Several alternative ones:
    - Alg7 - Greedy
    - Refinement
- Re-map and migrate objects accordingly
  - Registration mechanisms facilitate migration

Load balancing strategy

Greedy variant (simplified):

Sort compute objects (diamonds)

Repeat (until all assigned)

S = set of all processors that:

-- are not overloaded

-- generate least new commun.

P = least loaded {S}

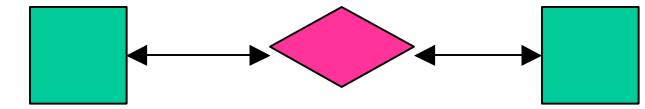
Assign heaviest compute to P

Refinement:

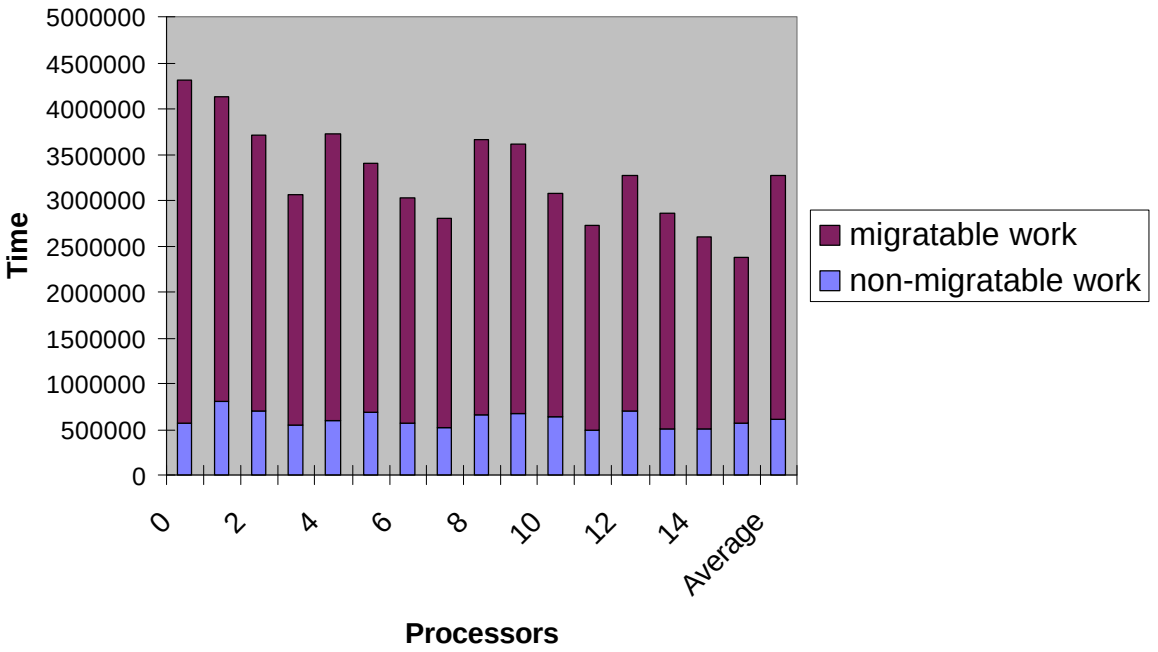
Repeat

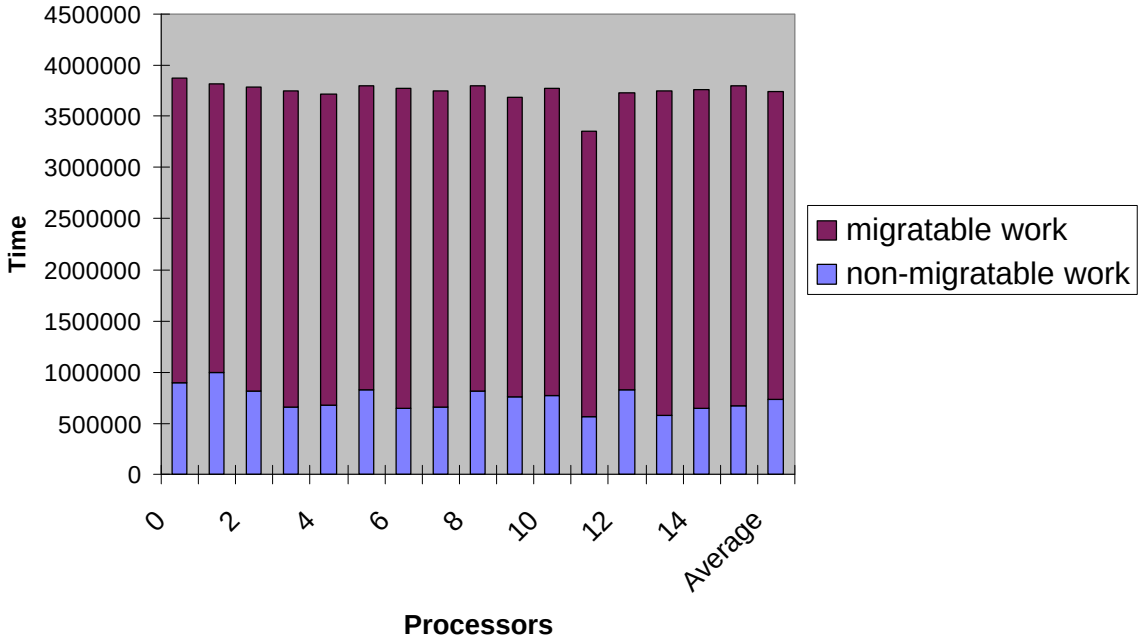
- Pick a compute from the most overloaded PE
- Assign it to a suitable underloaded PE

Until (No movement)

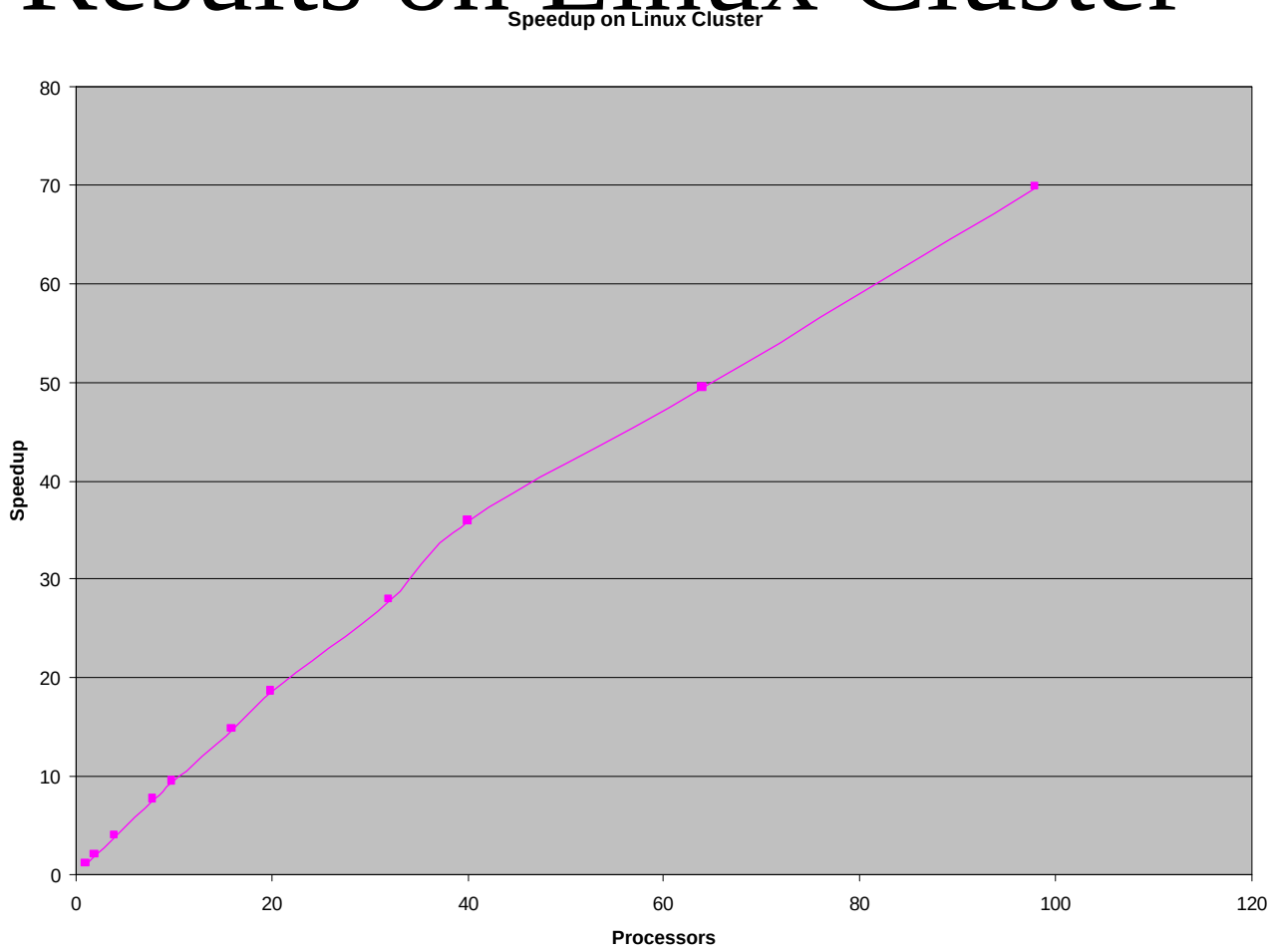


Cell Compute Cell

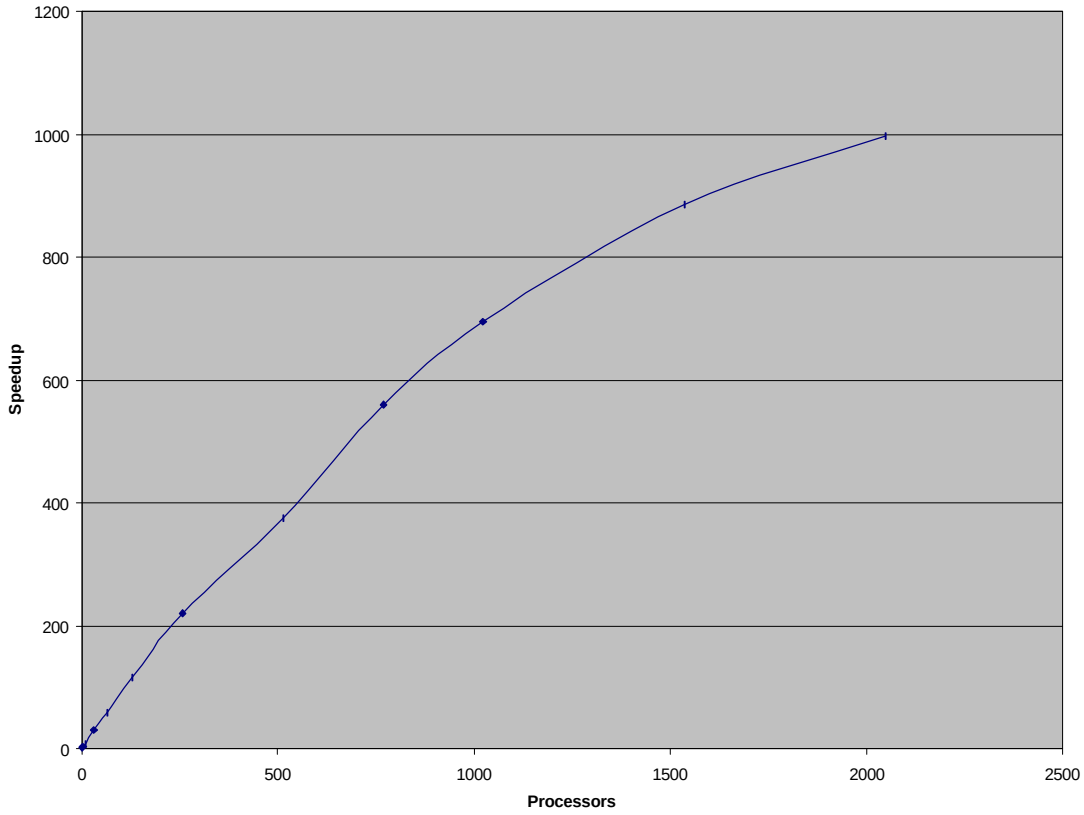

5000000
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
Time
0
2
4
6
8
10
12
14
Average
Processors
migratable work
non-migratable work


4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
Time
0
2
4
6
8
10
12
14
Average
Processors
migratable work
non-migratable work

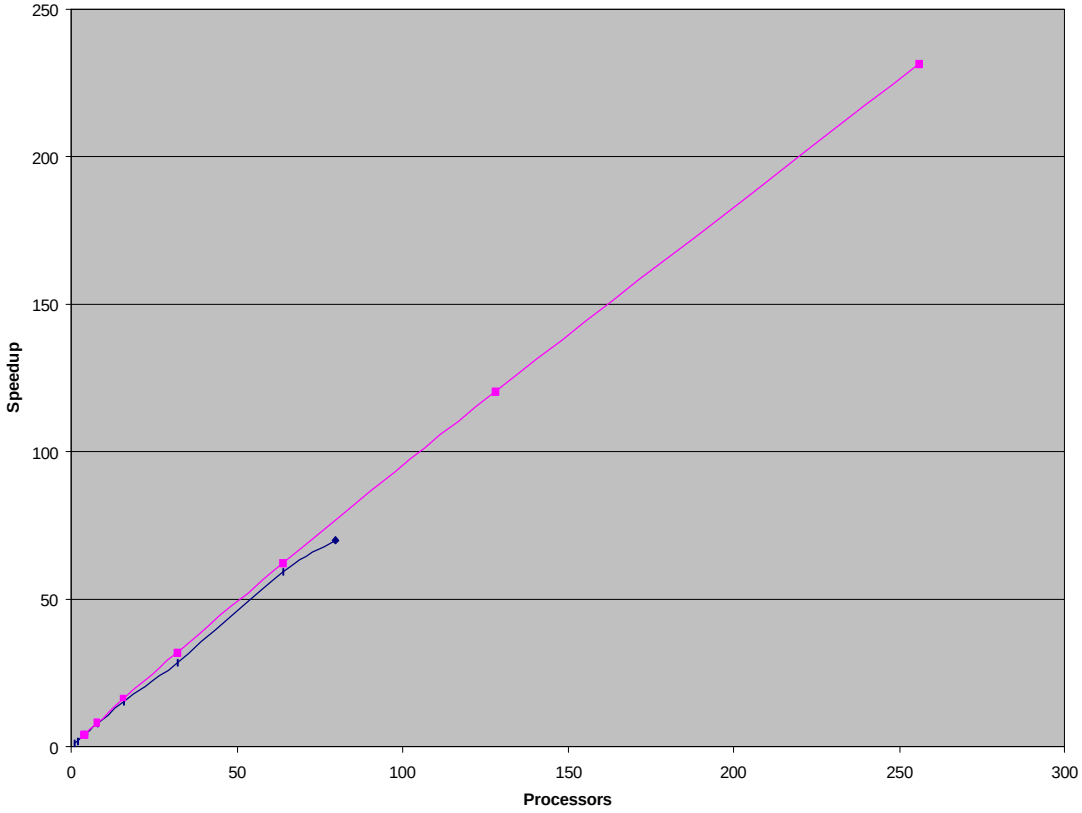
Results on Linux Cluster



Performance of Apo-A1 on Asci Red



Performance of Apo-A1 on O2k and T3E



Future and Planned work

- Increased speedups on 2k-10k processors
  - Smaller grainsizes
  - New algorithms for reducing communication impact
  - New load balancing strategies
- Further performance improvements for PME/FMA
  - With multiple timestepping
  - Needs multi-phase load balancing

Steered MD: example picture

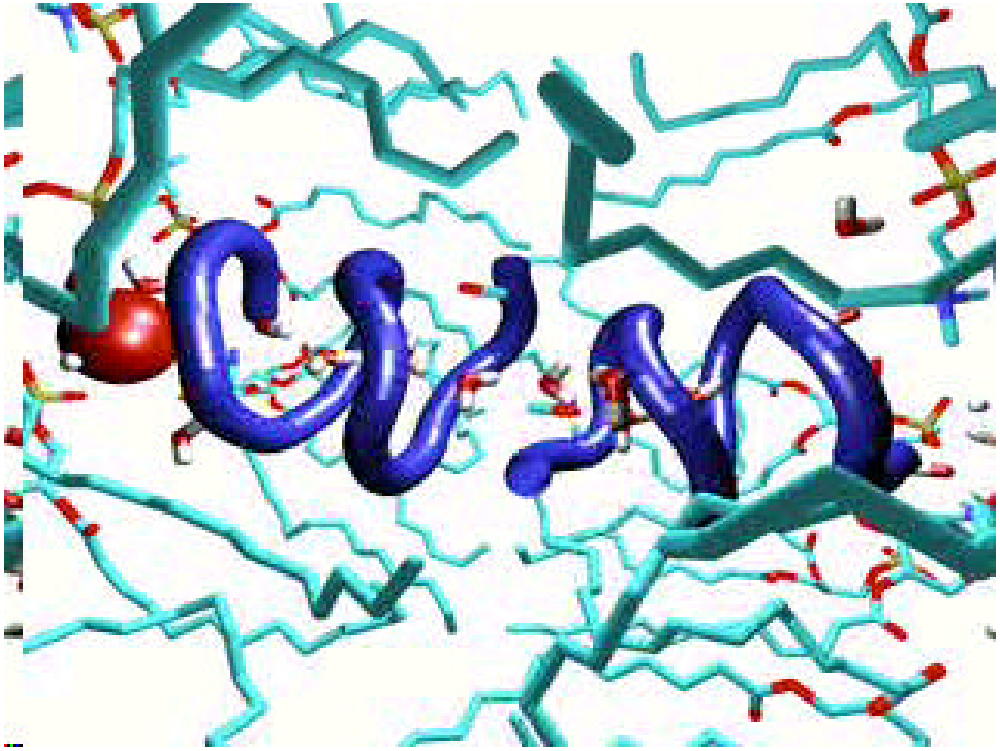


Image and Simulation by the theoretical biophysics group, Beckman Institute, UIUC