NAMD: Biomolecular Simulation on Thousands of Processors

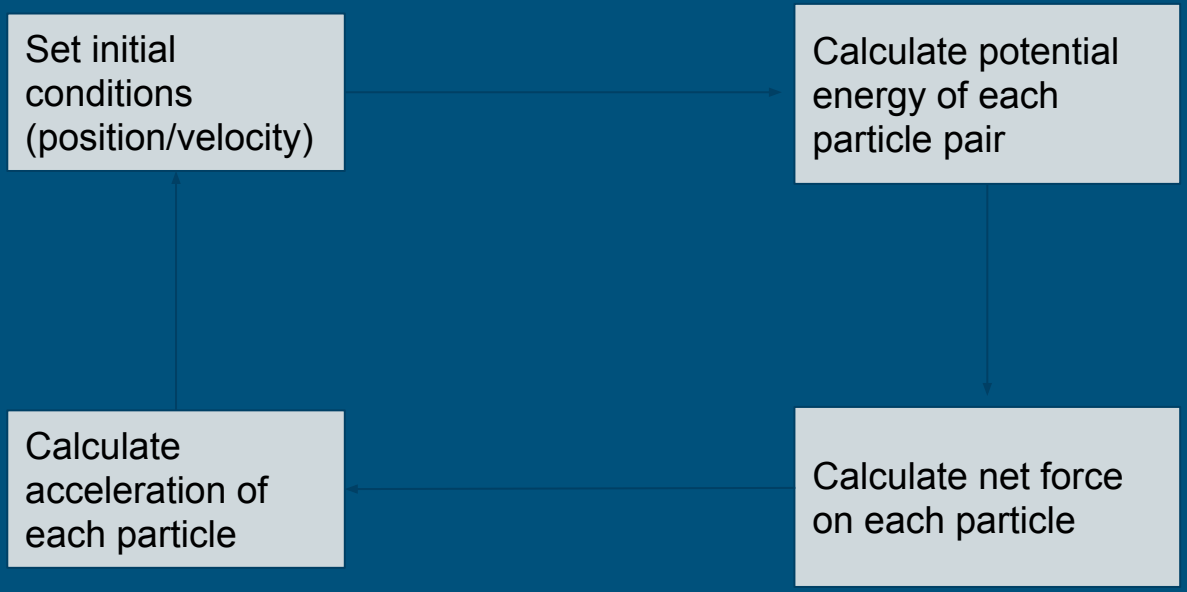
James C. Phillips* Gengbin Zheng† Sameer Kumar† Laxmikant V. Kalé†

*Beckman Institute, University of Illinois at Urbana-Champaign.
†Department of Computer Science and Beckman Institute, University of Illinois at Urbana-Champaign.

Molecular Dynamics (MD)

- Predict material properties.
- Drug discovery
- Model biomolecular interaction
- Used when real world experiments are too expensive / time consuming.
- Cheaper/quicker alternative to buying lab equipment and manufacturing a material/drug/biomolecule.

The Molecular Dynamics Cycle

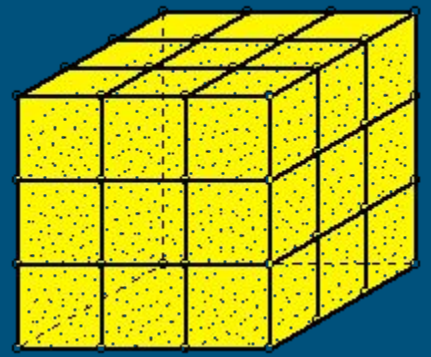


Accurate MD simulations - factors?

1. Setting good initial conditions using some well known distribution in Physics like the Boltzmann distribution.
2. Having good approximations of potential energy for every pair of particles.
3. Having a small time quanta for the integrating step that calculates position and velocity from acceleration. Usually, kept as 1 femtosecond (10^-15 second).

Parallelization Strategy

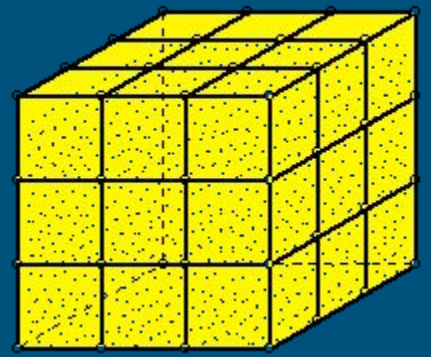
3D Space is divided into equal sized cubes called home patches. Each home patch has 26 neighbours.



1 home patch with it’s 26 neighbours

Parallelization Strategy

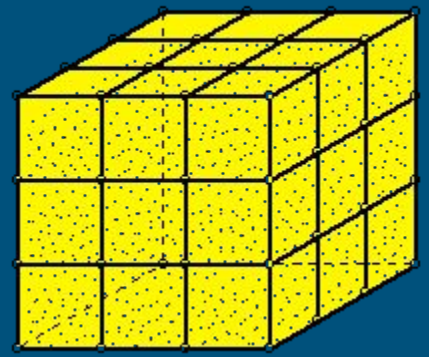
Size of each home patch is set to the cutoff radius (12Å). Thus, forces between two molecules are calculated only if they are in the same or neighbouring home patches.



1 home patch with it’s 26 neighbours

Parallelization Strategy

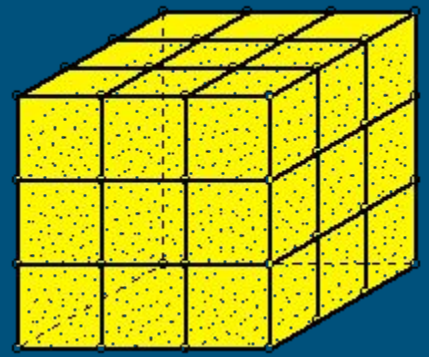
However, the fixed size of each cube prevents over-decomposition. For eg: with ATPase (a very large MD simulation) we can just have 704 patches.



1 home patch with it’s 26 neighbours

Parallelization Strategy

NAMD thus employs force decomposition as well. For each pair of neighbouring cubes, a non-bonded force computation object is created. There are 26/2=13 such objects per home patch. Therefore we have a total of 13+1=14 charm++ chares per home patch.

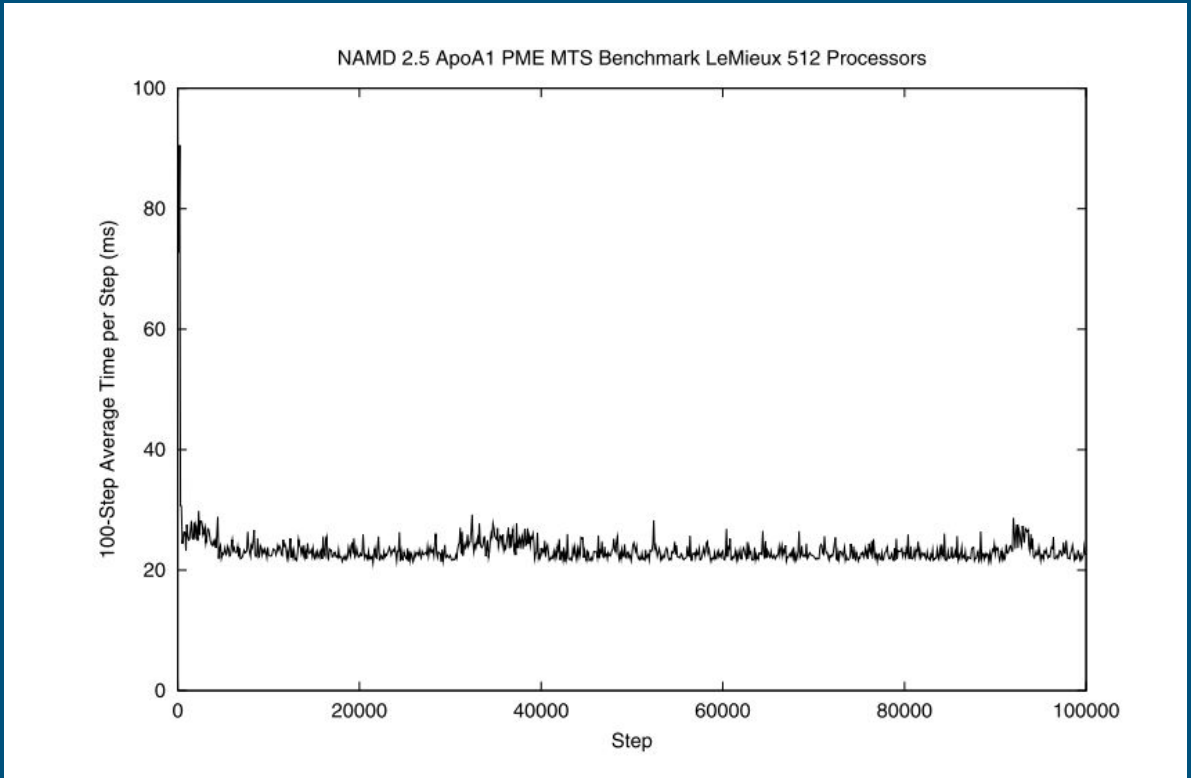


1 home patch with it’s 26 neighbours

Load Balancing

- Particles are moving, thus they can change their home patches with time.
- NAMD implements dynamic load balancing.
- Initially patches are distributed according to a recursive coordinate bisection scheme.
- Aggressive load balancing in the first 500 steps (frequency = 100 steps)
- Then (frequency = 4000 steps).

Load Balancing in practice



PME (Particle Mesh Ewald) in NAMD

- PME is an algorithm for full electrostatics calculation.
- This means calculation of forces between molecules not in neighbouring home patches is also done.
- NAMD includes PME in it’s codebase. The paper talks about the optimizations they made for PME calculation in NAMD.
- NAMD can thus operate in three modes, Cutoff (without PME), PME (PME every step), MTS (PME every 4 steps).

Communication Libraries

- On the PSC Leimux cluster, the authors noted that Charm++ with native MPI was very slow due to expensive MPI_Iprobe calls.
- They implemented Charm++ on top of the native Elan communication library on this cluster and observed a considerable amount of speedup.

Processors		Time/step		Speedup		GFLOPS	
Total	Per Node	MPI	Elan	MPI	Elan	MPI	Elan
1	1	28.08 s	28.08 s	1	1	0.480	0.480
128	4	248.3 ms	234.6 ms	113	119	54	57
256	4	135.2 ms	121.9 ms	207	230	99	110
512	4	65.8 ms	63.8 ms	426	440	204	211
510	3	65.7 ms	63.0 ms	427	445	205	213
1024	4	41.9 ms	36.1 ms	670	778	322	373
1023	3	35.1 ms	33.9 ms	799	829	383	397
1536	4	35.4 ms	32.9 ms	792	854	380	410
1536	3	26.7 ms	24.7 ms	1050	1137	504	545
2048	4	31.8 ms	25.9 ms	883	1083	423	520
1800	3	25.8 ms	22.3 ms	1087	1261	521	605
2250	3	19.7 ms	18.4 ms	1425	1527	684	733
2400	4	32.4 ms	27.2 ms	866	1032	416	495
2800	4	32.3 ms	32.1 ms	869	873	417	419
3000	4	32.5 ms	28.8 ms	862	973	414	467

From the charm++ github repo.

1. The way a parallel program written in Charm++ will communicate:
    - `netlrts-` : Charm++ communicates using the regular TCP/IP stack (UDP packets), which works everywhere but is fairly slow. Use this option for networks of workstations, clusters, or single-machine development and testing.
    - `gni-` , `pamilrts-` , `verbs-` , `ofi-` , `ucx-` : Charm++ communicates using direct calls to the machine's communication primitives. Use these versions on machines that support them for best performance.
    - `mpi-` : Charm++ communicates using MPI calls. This will work on almost every distributed machine, but performance is often worse than using the machine's direct calls referenced above.
    - `multicore-` : Charm++ communicates using shared memory within a single node. A version of Charm++ built with this option will not run on more than a single node.

Performance results

Processors		Time/step			Speedup			GFLOPS		
Total	Per Node	Cut	PME	MTS	Cut	PME	MTS	Cut	PME	MTS
1	1	24.89 s	29.49 s	28.08 s	1	1	1	0.494	0.434	0.480
128	4	207.4 ms	249.3 ms	234.6 ms	119	118	119	59	51	57
256	4	105.5 ms	135.5 ms	121.9 ms	236	217	230	116	94	110
512	4	55.4 ms	72.9 ms	63.8 ms	448	404	440	221	175	211
510	3	54.8 ms	69.5 ms	63.0 ms	454	424	445	224	184	213
1024	4	33.4 ms	45.1 ms	36.1 ms	745	653	778	368	283	373
1023	3	29.8 ms	38.7 ms	33.9 ms	835	762	829	412	331	397
1536	4	25.7 ms	44.7 ms	32.9 ms	968	660	854	477	286	410
1536	3	21.2 ms	28.2 ms	24.7 ms	1175	1047	1137	580	454	545
2048	4	25.8 ms	46.7 ms	25.9 ms	963	631	1083	475	274	520
1800	3	18.6 ms	25.8 ms	22.3 ms	1340	1141	1261	661	495	605
2250	3	15.6 ms	23.5 ms	18.4 ms	1599	1256	1527	789	545	733
2400	4	22.6 ms	44.6 ms	27.2 ms	1099	661	1032	542	286	495
2800	4	22.1 ms	43.6 ms	32.1 ms	1127	676	873	556	293	419
3000	4	22.6 ms	39.6 ms	28.8 ms	1102	743	973	544	322	467

NAMD performance on 327K atom ATPase benchmark system for Charm++ on ELAN

Questions?