Simulating Biomolecules on GPUs with the Multilevel Summation Method

David J. Hardy

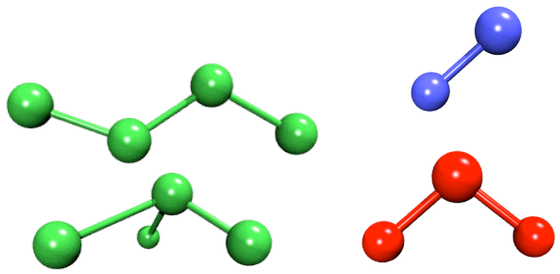
Theoretical and Computational Biophysics Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
http://www.ks.uiuc.edu/Research/gpu/

Bio-molecular Simulations on Future Computing Architectures,
Oak Ridge, Tennessee, Sept. 16-17, 2010.

Our Software Tools

- VMD - setup, analysis, visualization
- NAMD - molecular dynamics of biomolecules

$$m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i = -\vec{\nabla} U(\vec{R})$$ ⟵ integrate for 1 billion time steps



$$U(\vec{R}) = \underbrace{\sum_{bonds} k_i^{bond}(r_i - r_0)^2}_{U_{bond}} + \underbrace{\sum_{angles} k_i^{angle}(\theta_i - \theta_0)^2}_{U_{angle}} +$$

$$\underbrace{\sum_{dihedrals} k_i^{dihe}[1 + \cos(n_i\phi_i + \delta_i)]}_{U_{dihedral}} +$$

computational bottleneck ⟶

$$\underbrace{\sum_i \sum_{j \neq i} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right]}_{\text{(van der Waals)}} + \underbrace{\sum_i \sum_{j \neq i} \frac{q_i q_j}{\epsilon r_{ij}}}_{U_{nonbond}}$$ (electrostatics)

Multilevel Summation Method

R. Skeel, I. Tezcan, D. Hardy. *J. Comp. Chem.* 23:673-684, 2002.

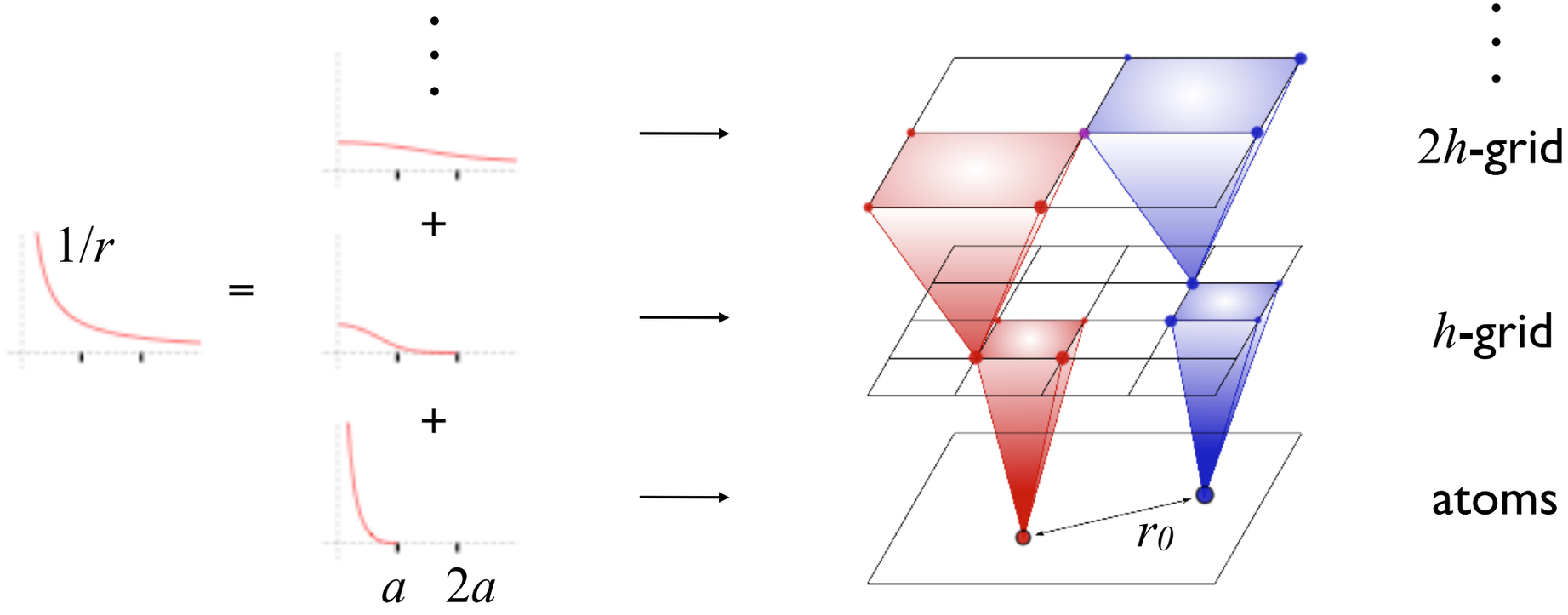
- Fast algorithm for N-body electrostatics
- Calculates sum of smoothed pairwise potentials interpolated from a hierarchal nesting of grids
- Advantages over PME (particle-mesh Ewald) and/or FMM (fast multipole method):
  - Algorithm has linear time complexity
  - Allows non-periodic or periodic boundaries
  - Produces continuous forces for dynamics (advantage over FMM)
  - Avoids 3D FFTs for better parallel scaling (advantage over PME)
  - Permits polynomial splittings (no *erfc()* evaluation, as used by PME)
  - Spatial separation allows use of multiple time steps
  - Can be extended to other types of pairwise interactions (e.g., vdW)

MSM Main Ideas

- Split the $1/r$ potential into a short-range cutoff part plus smoothed parts that are successively more slowly varying. All but the top level potential are cut off.
- Smoothed potentials are interpolated from successively coarser grids.
- Finest grid spacing h and smallest cutoff distance a are doubled at each successive level.

Split the $1/r$ potential

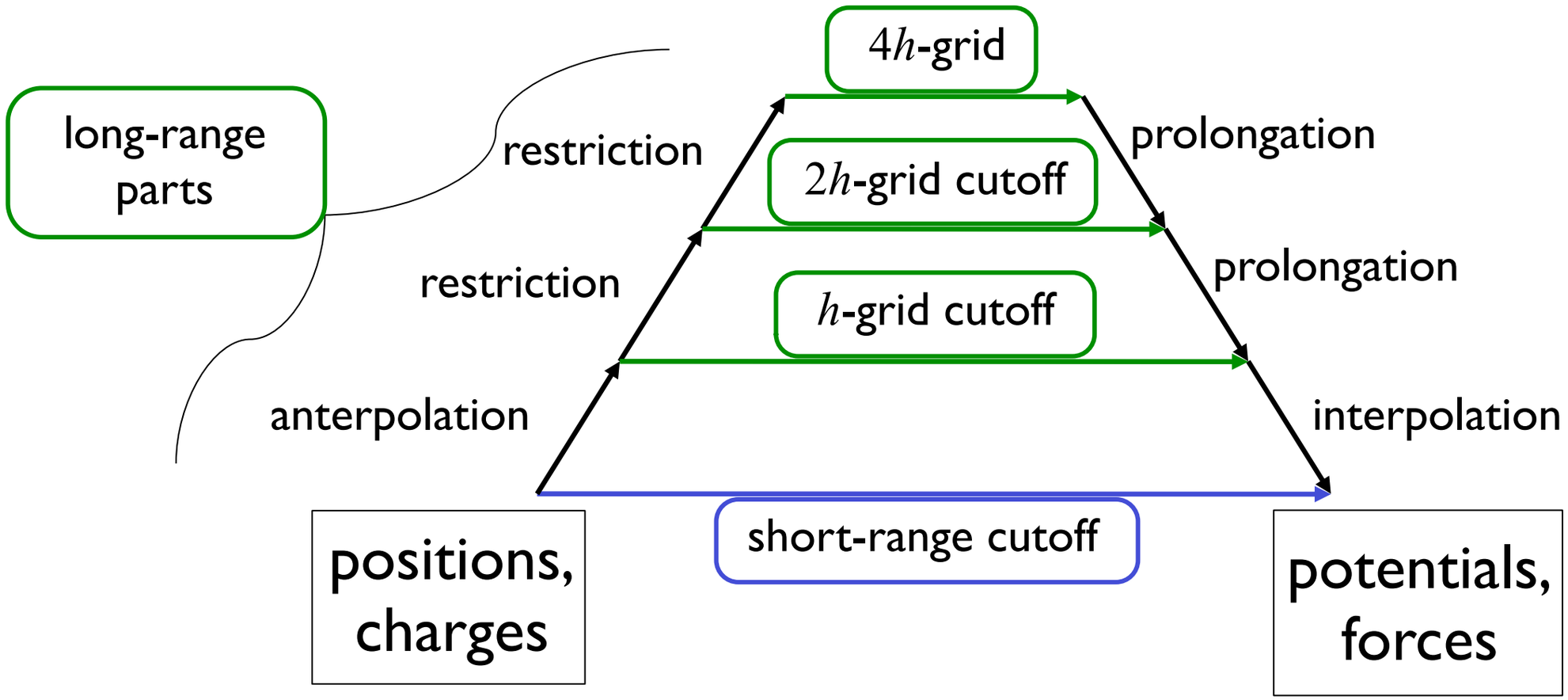
Interpolate the smoothed potentials



MSM Calculation

force	=	exact short-range part	+	interpolated long-range part

Computational Steps

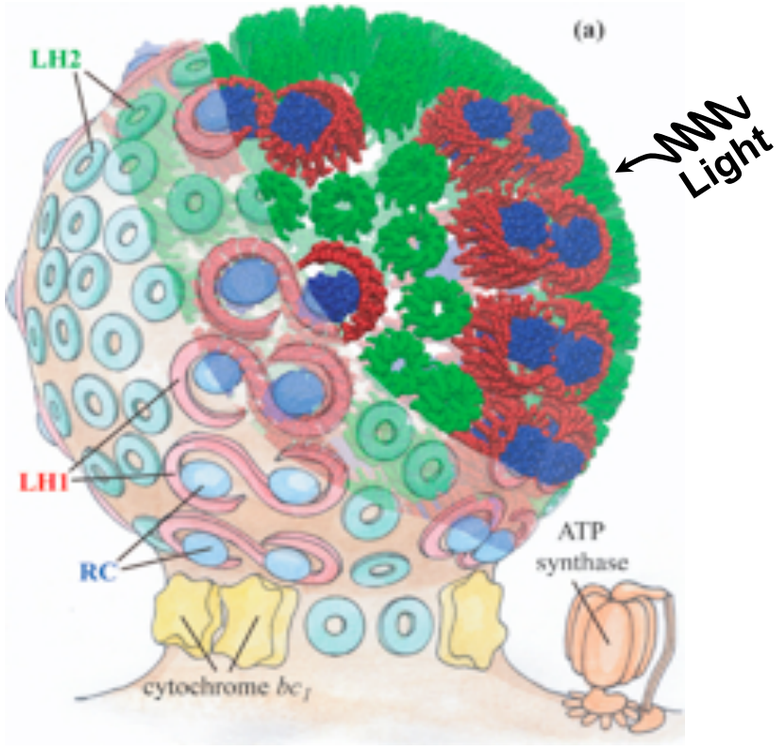


Using MSM to calculate...

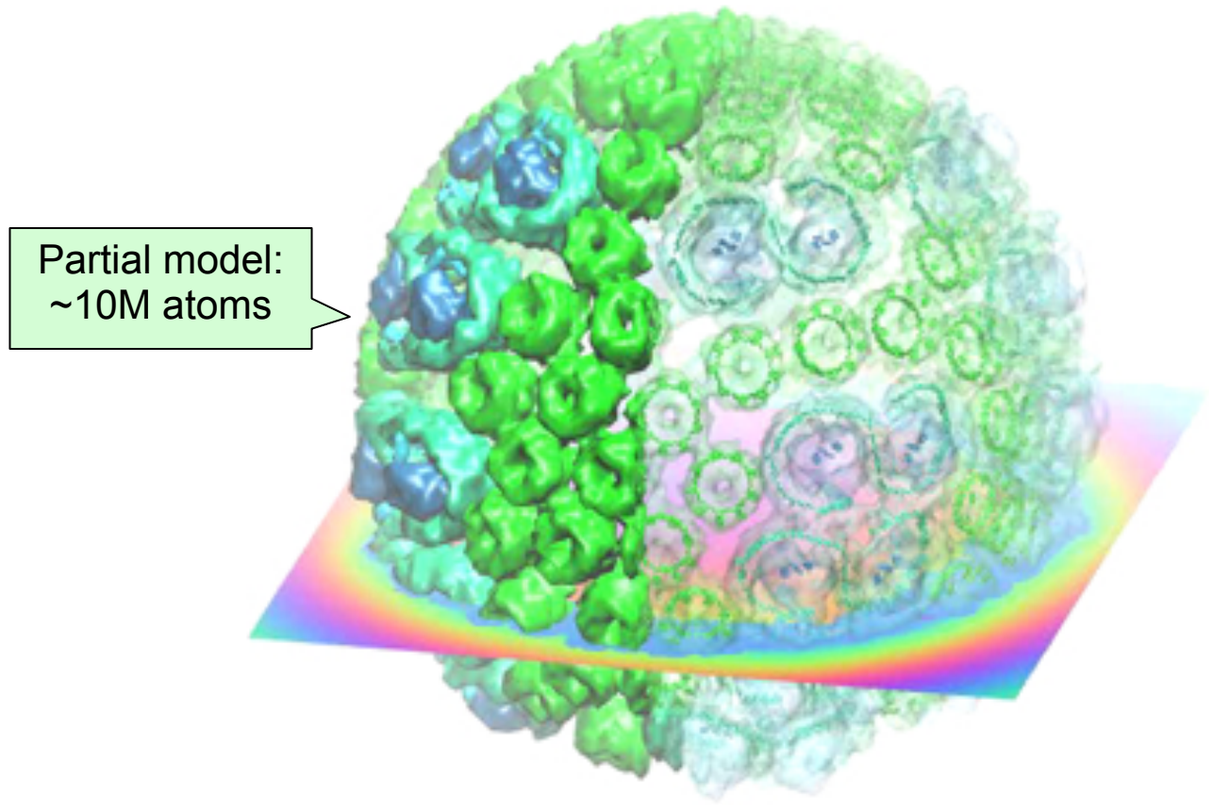
...electrostatic potential maps

Application of MSM in VMD to Photosynthesis

Investigations of the chromatophore, a photosynthetic organelle



Electrostatics needed to build full structural model, place ions, study macroscopic properties

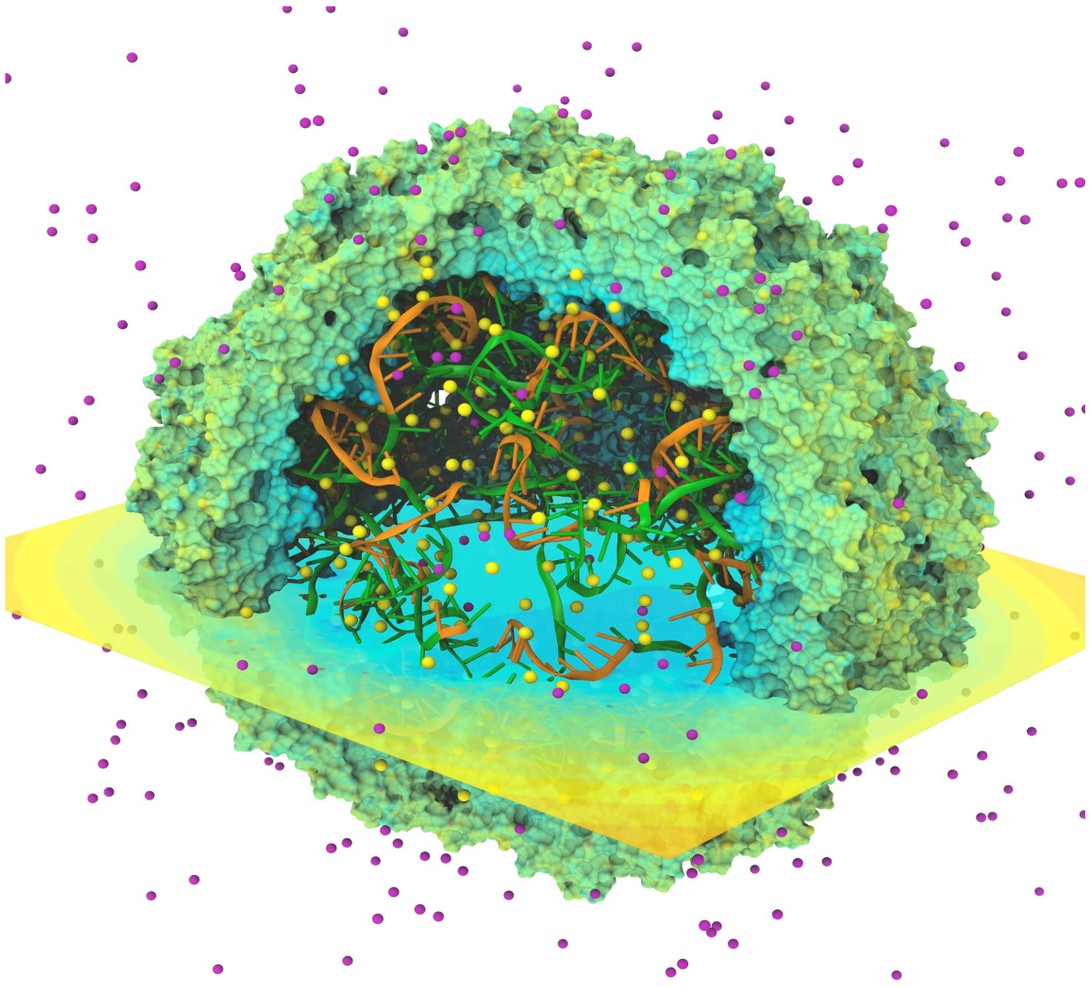


Electrostatic field of chromatophore model from **multilevel summation method**: computed with 3 GPUs (G80) in ~90 seconds, 46x faster than single CPU core in 1 hr, 10 min

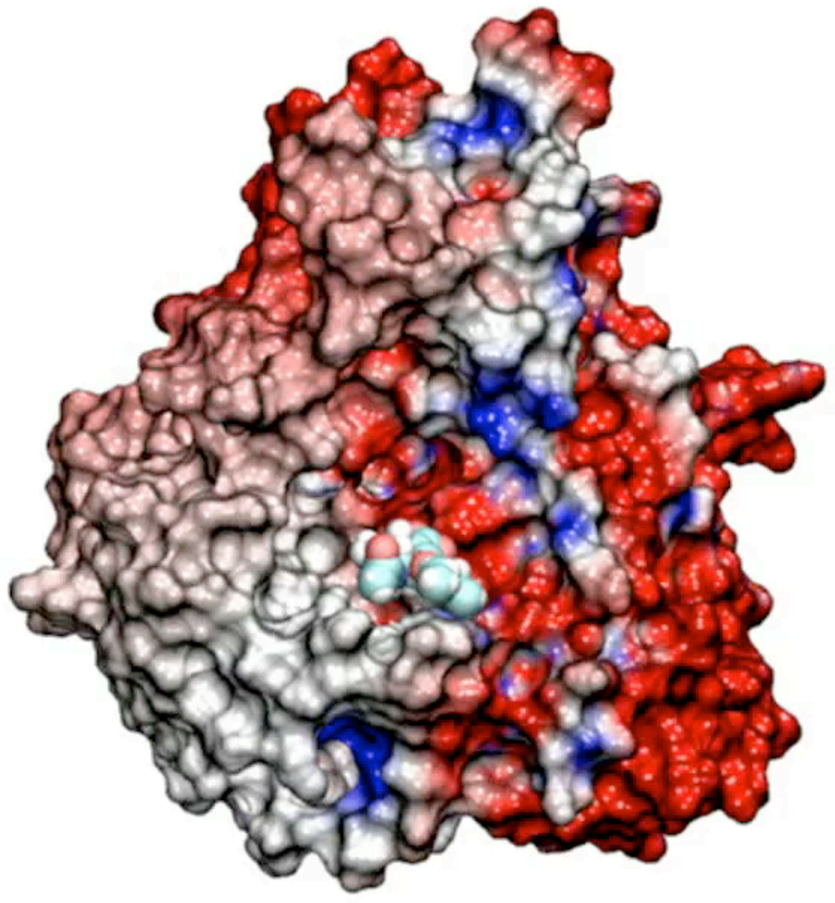
Full chromatophore model will permit structural, chemical and kinetic investigations at a structural systems biology level

More Applications of MSM in VMD

Investigations of Satellite Tobacco Mosaic Virus (STMV) and "swine" flu virus



Time averaged potential maps:
calculating electrostatics for
thousands of trajectory frames,
1.5 hour job reduced to 3 minutes
(NCSA "AC" cluster)



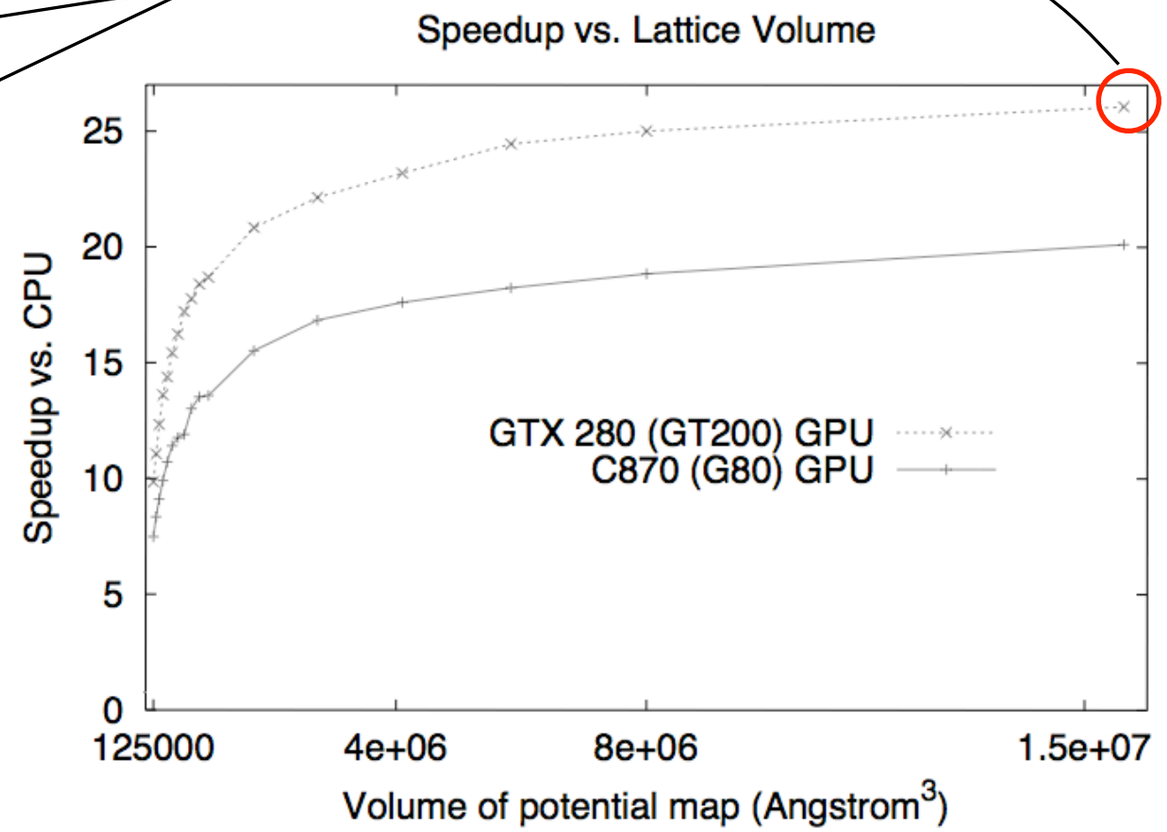
Investigation of drug (Tamiflu) resistance of the "swine" flu virus demanded **fast response!**
Calculating electrostatics for 20,000 trajectory frames, 27.8 hour job reduced to 1.1 hours
(Linux workstation with Quadro 5800)

MSM Potentials on the GPU

Accelerate **short-range cutoff** and **lattice cutoff** parts

Performance profile for 0.5 Å map of potential for 1.5 M atoms.
Hardware platform is Intel QX6700 CPU and NVIDIA GTX 280.

Computational steps	CPU (s)	w/ GPU (s)	Speedup
Short-range cutoff	480.07	14.87	32.3
Long-range anterpolation	0.18		
restriction	0.16		
lattice cutoff	49.47	1.36	36.4
prolongation	0.17		
interpolation	3.47		
Total	533.52	20.21	26.4

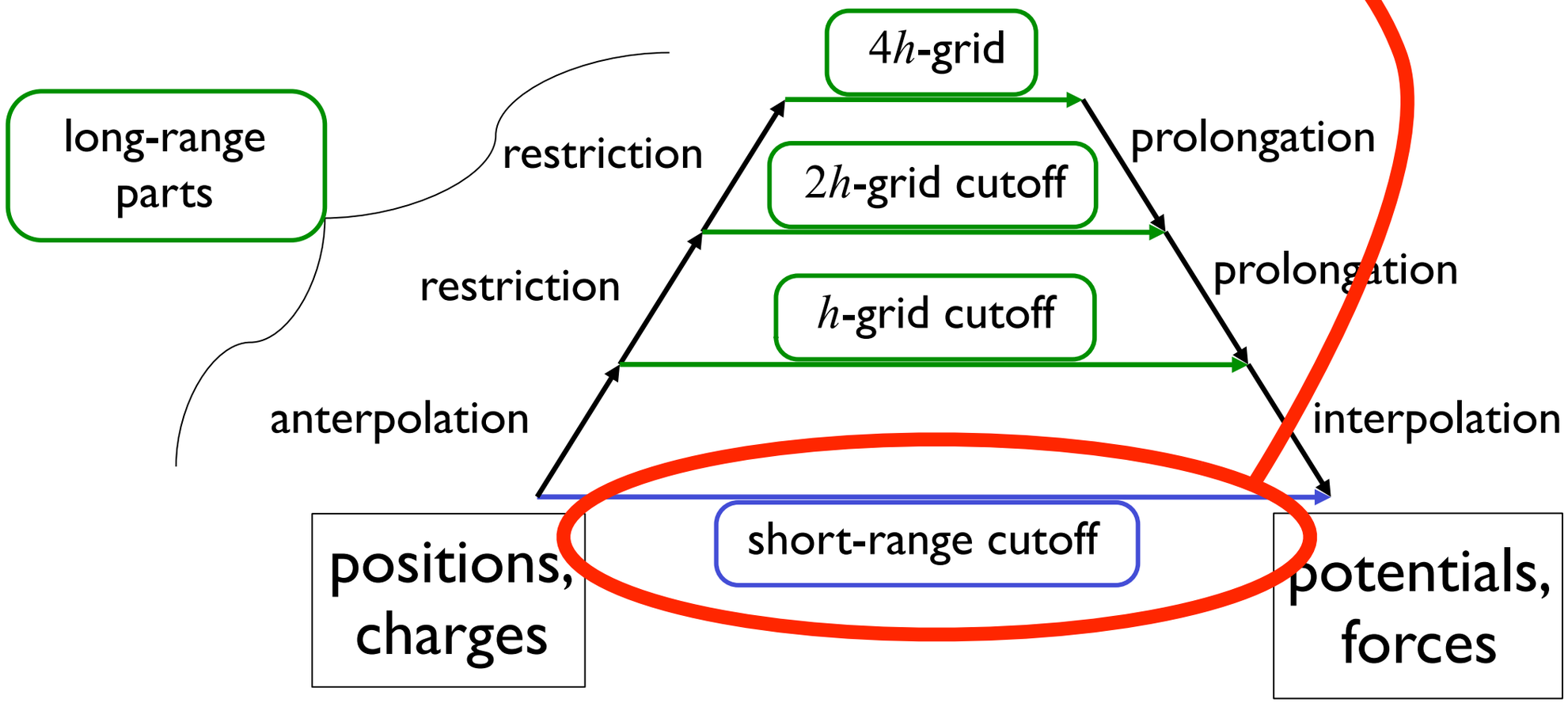


Multilevel summation of electrostatic potentials using graphics processing units.
D. Hardy, J. Stone, K. Schulten. *J. Parallel Computing*, 35:164-177, 2009.

MSM Calculation

force = exact short-range part + interpolated long-range part

Computational Steps

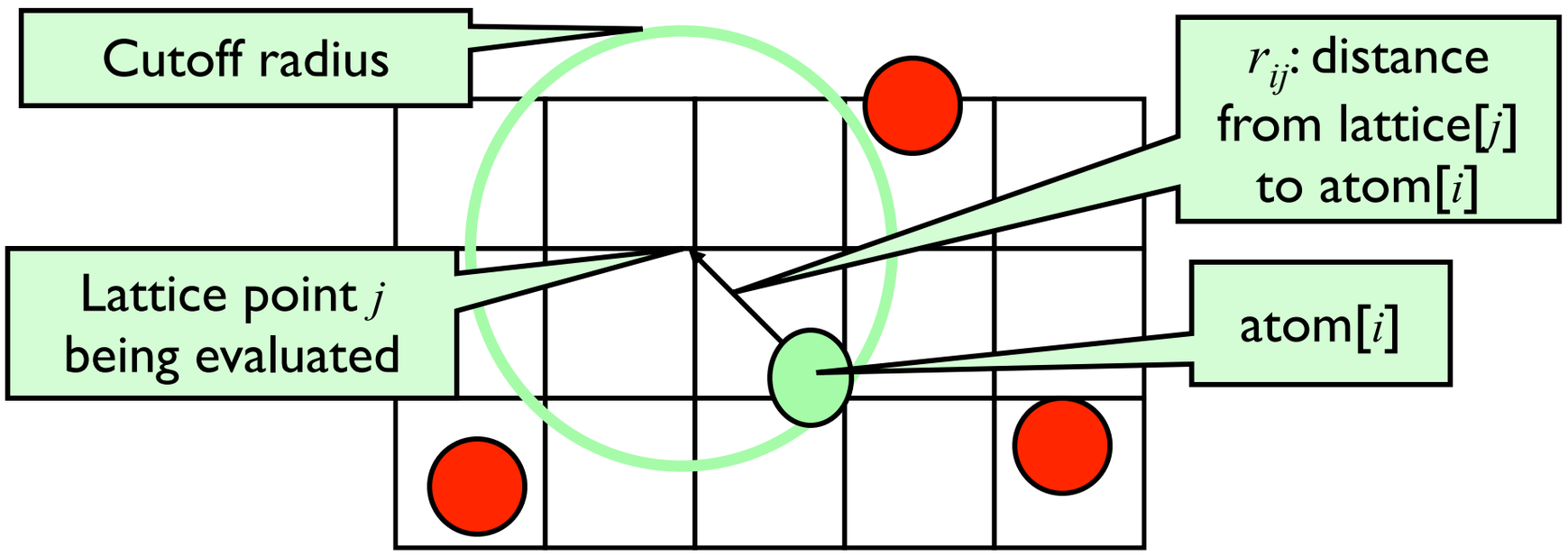


Short-range Cutoff Summation

- Each lattice point accumulates electrostatic potential contribution from atoms within cutoff distance:

  if (r_{ij} < cutoff)
    potential[j] += (charge[i] / r_{ij}) * $s(r_{ij})$

- Smoothing function $s(r)$ is algorithm dependent



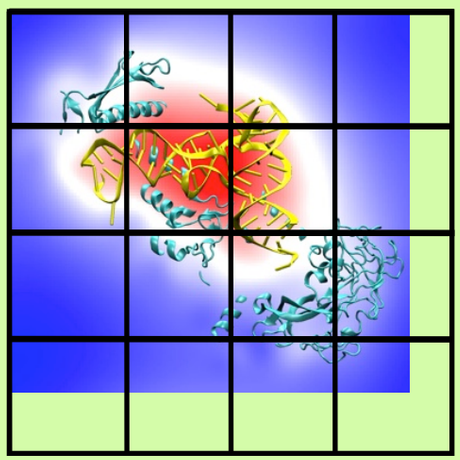
Short-range Cutoff Summation on GPU

- Atoms are spatially hashed into fixed-size bins (8 deep, stored x/y/z/q) to obtain **memory coalesced reads**
- CPU handles overflowed bins, so GPU kernel can be **aggressive** (choosing 4Å bin length works well in practice)
- GPU thread block calculates its respective region of the potential map
- Solve costly bin/region neighbor checks with **lookup table of offsets**

Each thread block cooperatively loads atom bins from surrounding neighborhood into shared memory for evaluation

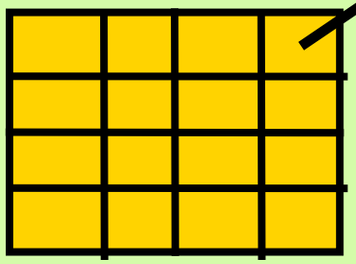
Shared memory

atom bin



Global memory

Potential map regions

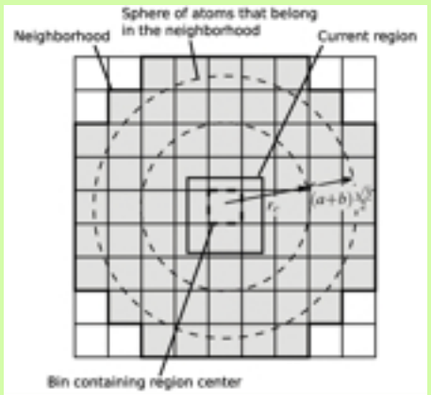


Bins of atoms

Constant memory

Offsets for bin neighborhood

Lookup table encodes "logic" of spatial geometry



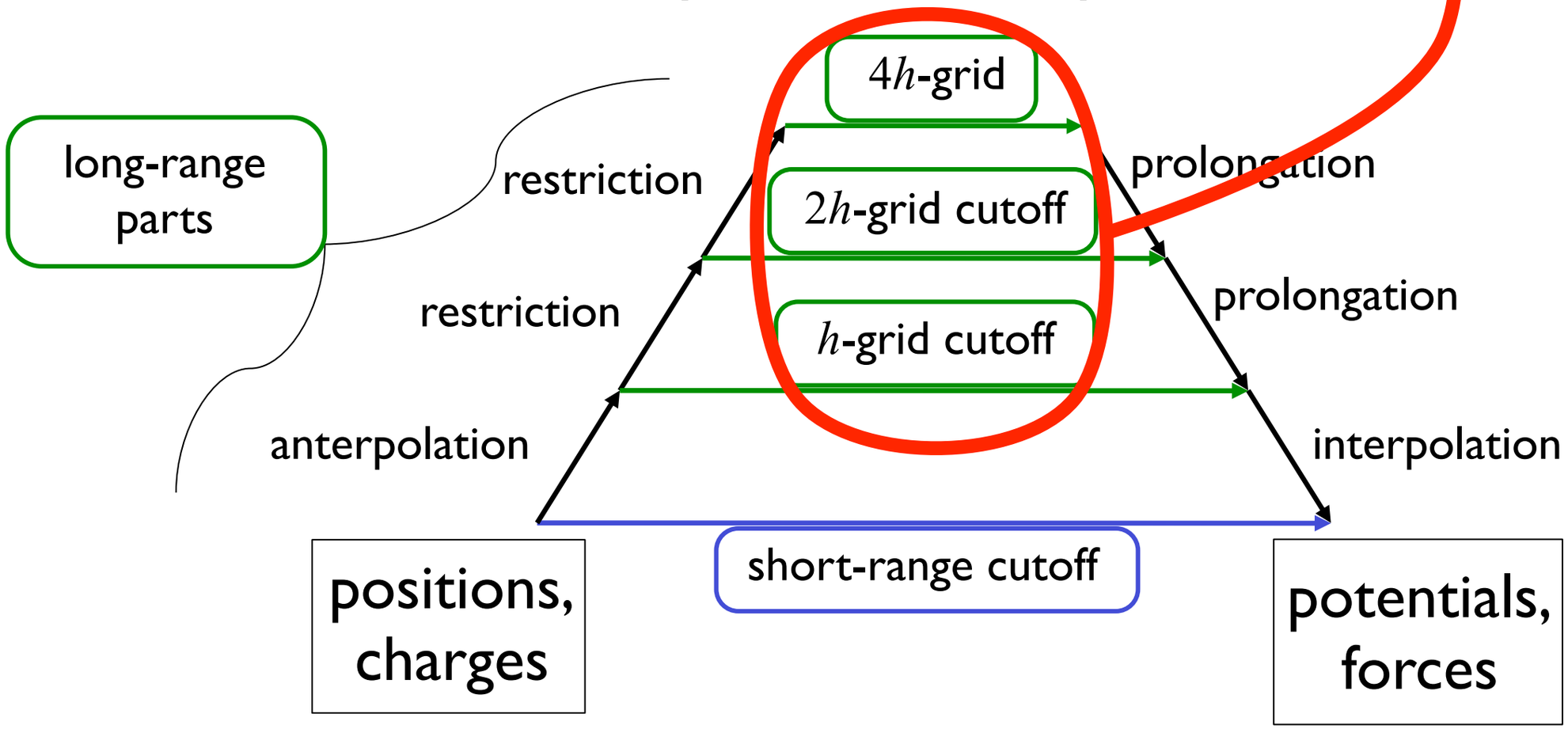
Using CPU to Improve GPU Performance

- GPU performs best when the work evenly divides into the number of threads / processing units
- Optimization strategy:
  - Use the CPU to "regularize" the GPU workload
  - Use fixed size bin data structures, with "empty" slots skipped or producing zeroed out results
  - Handle exceptional or irregular work units on the CPU while the GPU processes the bulk of the work
  - On average, the GPU is kept highly occupied to attain good fraction of peak performance

MSM Calculation

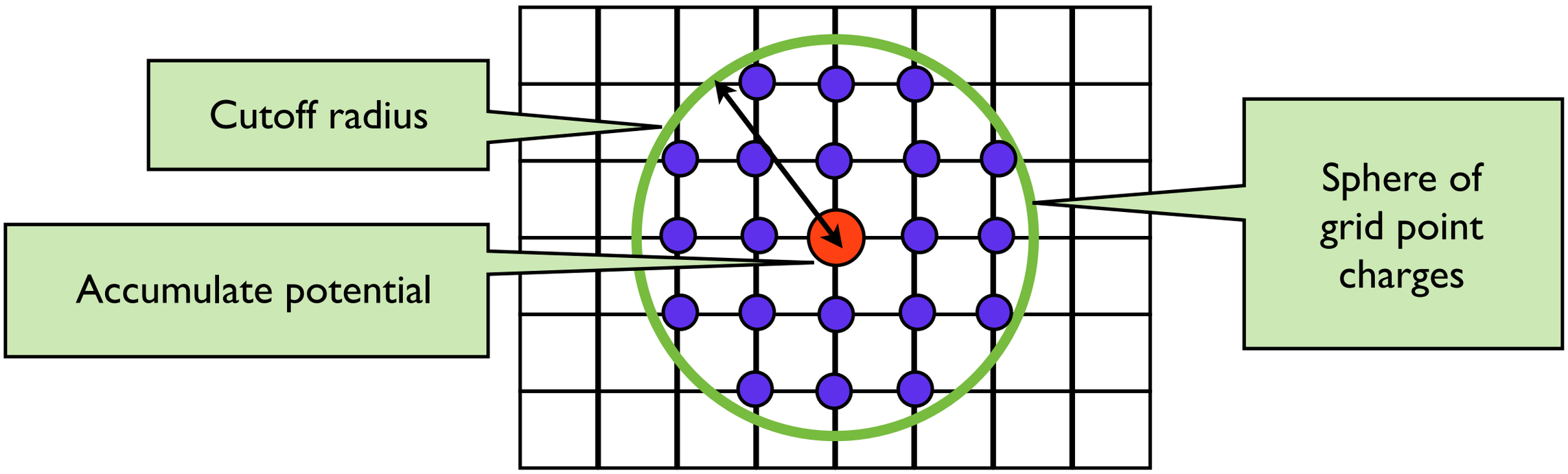
force = **exact short-range part** + **interpolated long-range part**

Computational Steps



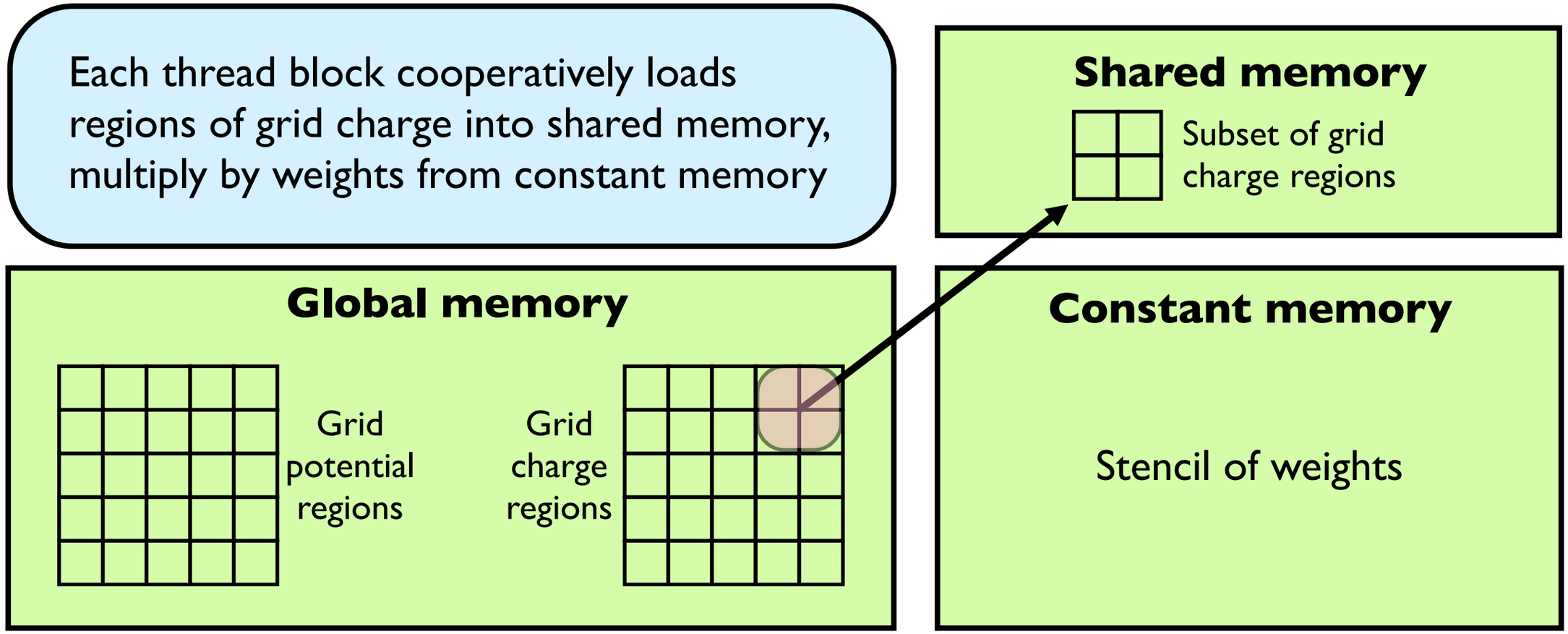
Lattice Cutoff Summation

- Potential summed from grid point charges within cutoff
- Uniform spacing enables distance-based interactions to be precomputed as stencil of "weights"
- Weights at each level are identical up to scaling factor (!)
- Calculate as 3D convolution of weights
  - stencil sizes range from 9x9x9 up to 23x23x23



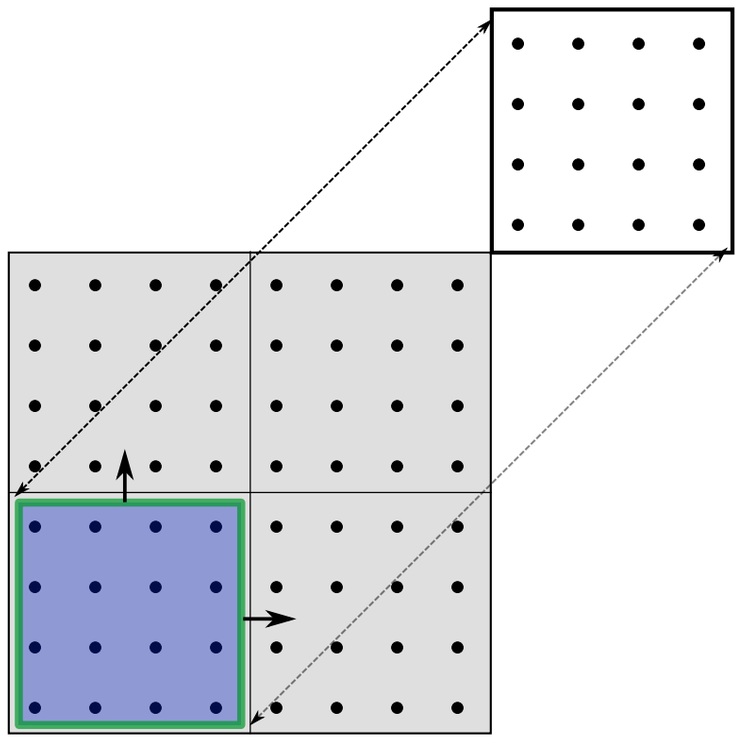
Lattice Cutoff Summation on GPU

- Store weights in constant memory (padded up to next multiple of 4)
- Thread block calculates 4x4x4 region of potentials, stored contiguously for **memory coalesced reads**
- Pack all regions over all levels into 1D array (each level padded with zero-charge region)
- Store **map of level array offsets** in constant memory
- Kernel has thread block loop over surrounding regions of charge (load into shared memory)
- All **grid levels are calculated concurrently**, scaled by level factor (keeps GPU from running out of work at upper grid levels)



Apply Weights Using Sliding Window

- Constant memory offers best performance when thread block collectively accesses the same location
- Read 8x8x8 grid charges (8 regions) into shared memory
- Window of size 4x4x4 maintains same relative distances
- Slide window by 4 shifts along each dimension



Using MSM to calculate...

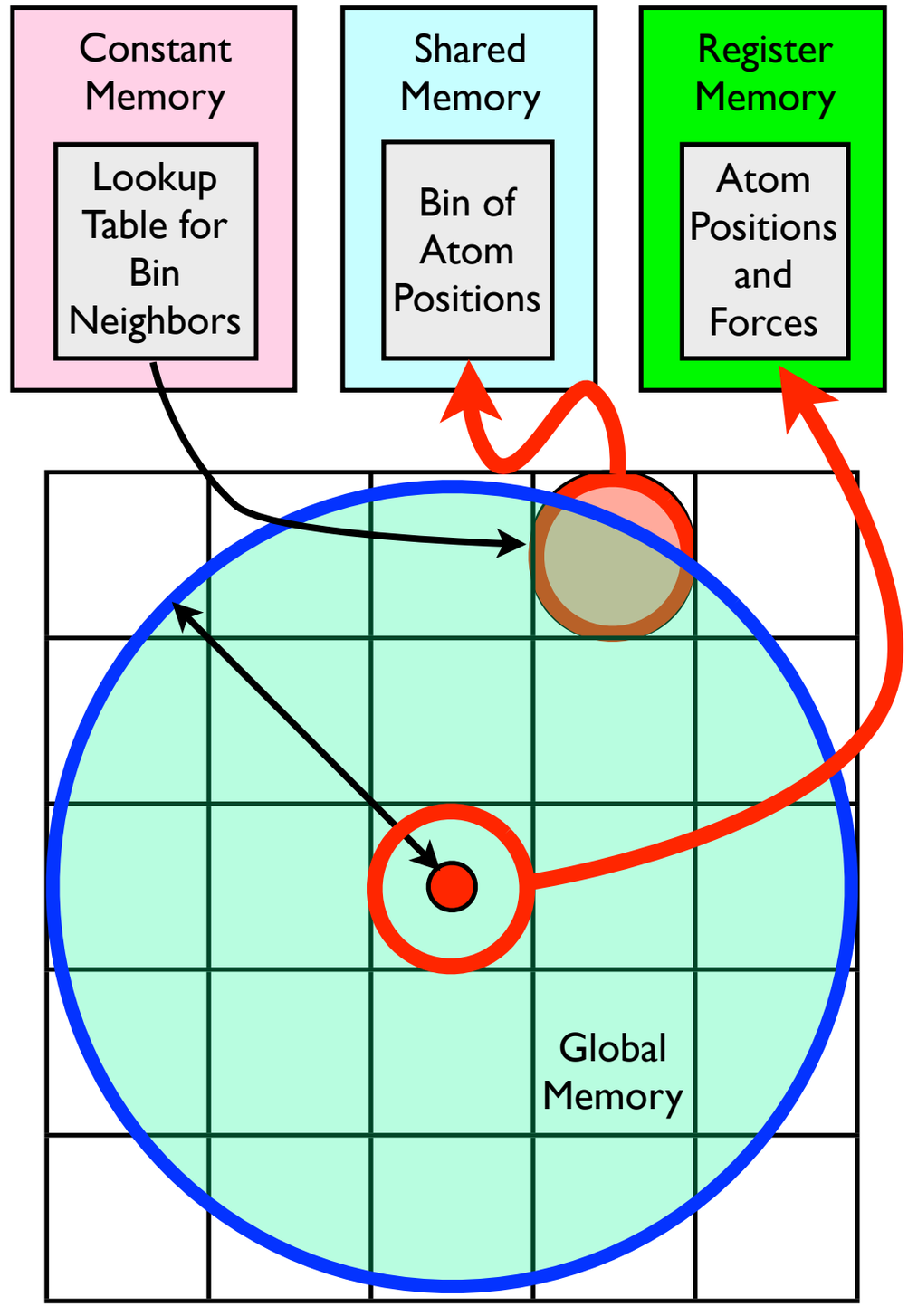
...electrostatic forces for MD

Designing GPU Kernels for Short-range Non-bonded Forces

- Calculate both electrostatics and van der Waals interactions (need atom coordinates and parameters)
  - supporting "NBFix" parameters for van der Waals
- Should we use pairlists?
  - Reduces computation, increases and delocalizes memory access
- Is single precision enough? Do we need double precision?
- How should we handle non-bonded exclusions?
  - Detect and omit excluded pairs (use bit masks)
  - Ignore, fix with CPU (use force clamping)
- How do we calculate expensive functional forms?
  - PME requires erfc(): is it faster to use an interpolation table?
  - Better if we don't have any expensive functions!

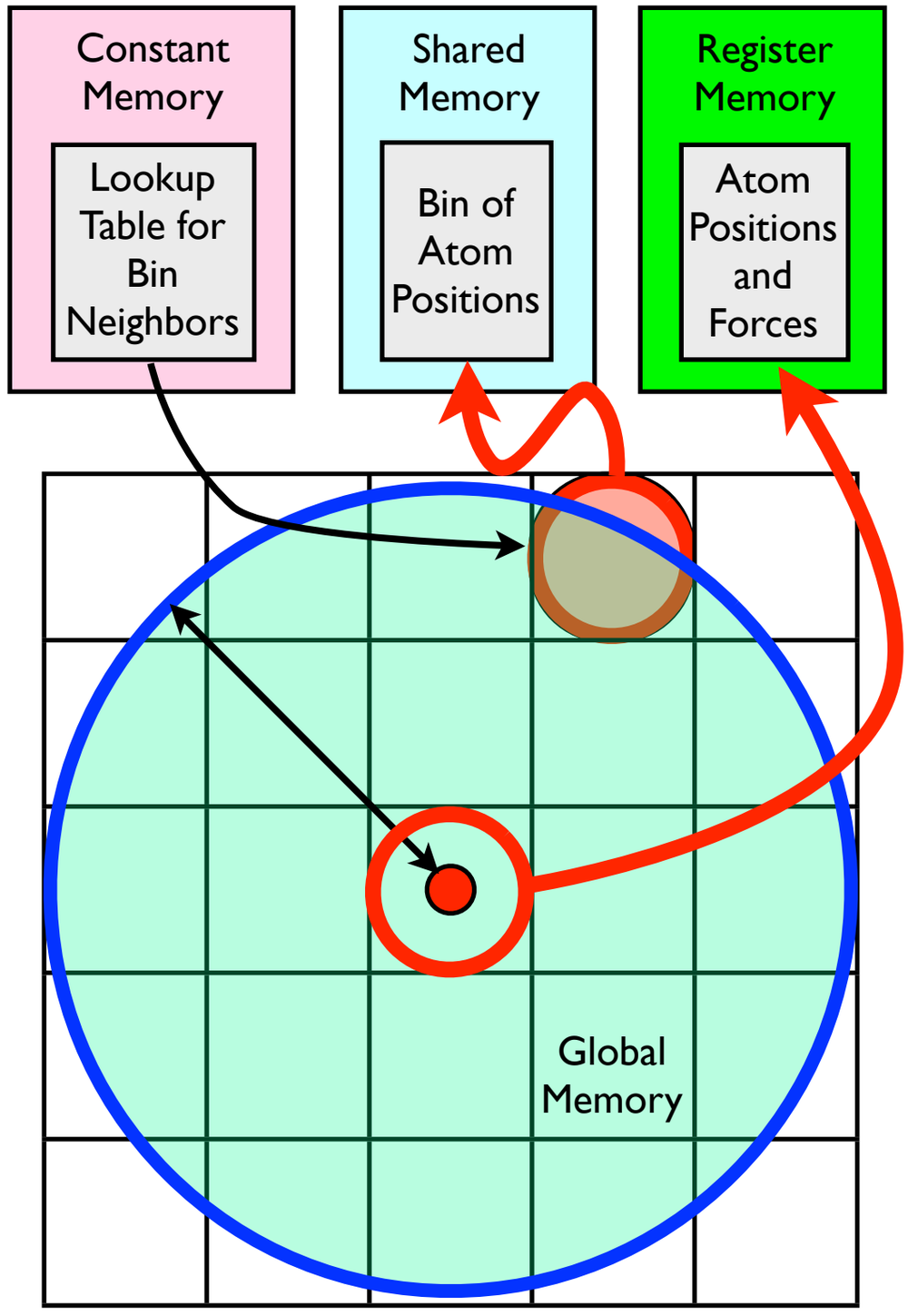
GPU Kernel for Short-range MSM (1)

- CPU sorts atoms into bins, copies bins to GPU global memory
- Each bin is assigned to a thread block
- Threads are assigned to individual atoms
- Loop over surrounding neighborhood of bins, summing forces and energies from their atoms
- Calculation for MSM involves rsqrt() plus several multiplies and adds
- CPU copies forces and energies back from GPU global memory



GPU Kernel for Short-range MSM (2)

- Each thread accumulates atom force and energies in registers
- Bin neighborhood index offsets stored in constant memory
- Load atom bin data into shared memory; atom data and bin "depth" are carefully chosen to permit coalesced reads from global memory
- Check for and omit excluded pairs
- Thread block performs sum reduction of energies
- Coalesced writing of forces and energies (with padding) to GPU global memory
- CPU sums energies from bins



Initial Results

(GPU: NVIDIA GTX-285, using CUDA 3.0; CPU: 2.4 GHz Intel Core 2 Q6600 quad core)

Box of 21,950 flexible waters, 12 A cutoff, 1ps, 1fs time step	CPU only	with GPU	Speedup vs. NAMD/CPU
NAMD* with PME	1199.8 s	210.5 s	5.7 x
NAMD-Lite** with MSM	5183.3 s (4598.6 short, 572.23 long)	176.6 s (93.9 short, 63.1 long)	6.8 x (19% over NAMD/GPU)

* using original NAMD/GPU

** NAMD-Lite timings do not overlap GPU with CPU, whereas NAMD does

Looking Ahead

- Parallelize MSM for NAMD
- Tune GPU-accelerated MSM for MD forces and bring it into NAMD-Lite
- Provide a NAMD-Lite plugin into VMD for energy evaluation, minimization, and short time-scale simulations (for setup and analysis)

Acknowledgments

- Bob Skeel (Purdue University)
- John Stone (University of Illinois at Urbana-Champaign)
- Jim Phillips (University of Illinois at Urbana-Champaign)
- Klaus Schulten (University of Illinois at Urbana-Champaign)
- Funding from NSF and NIH