Molecular Dynamics Simulations of Biomolecules on GPUs Using the Multilevel Summation Method

David J. Hardy

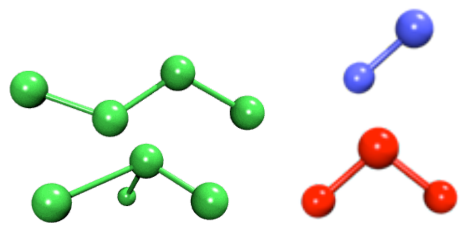
Theoretical and Computational Biophysics Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
http://www.ks.uiuc.edu/Research/gpu/

SIAM Conference on Computational Science and Engineering
Reno, Nevada, February 28, 2011

Our Software Tools

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

$$m_i \frac{d^2\vec{r}_i}{dt^2} = \vec{F}_i = -\vec{\nabla} U(\vec{R}) \longleftarrow$$ 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 $\longrightarrow$

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

(Lennard-Jones) (electrostatics)

Multilevel Summation Method

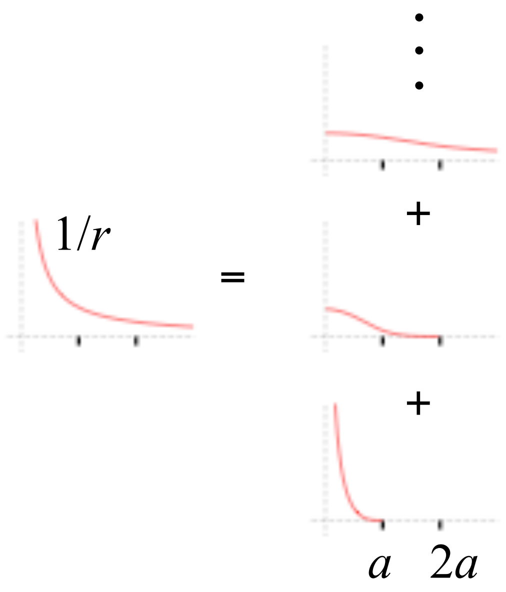
R. Skeel, I. Tezcan, D. Hardy. *J. Comp. Chem.* 23:673-684, 2002.

- 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)

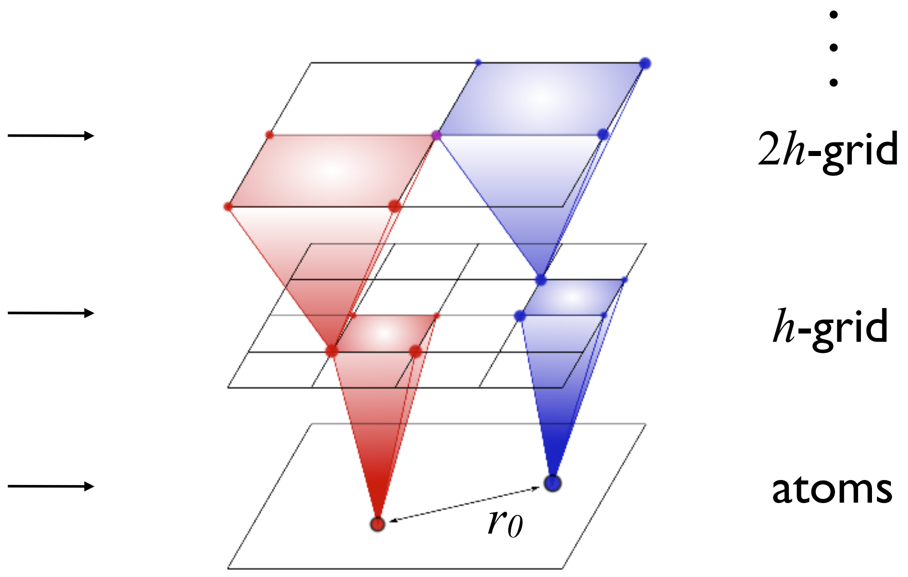
MSM Main Ideas

- 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



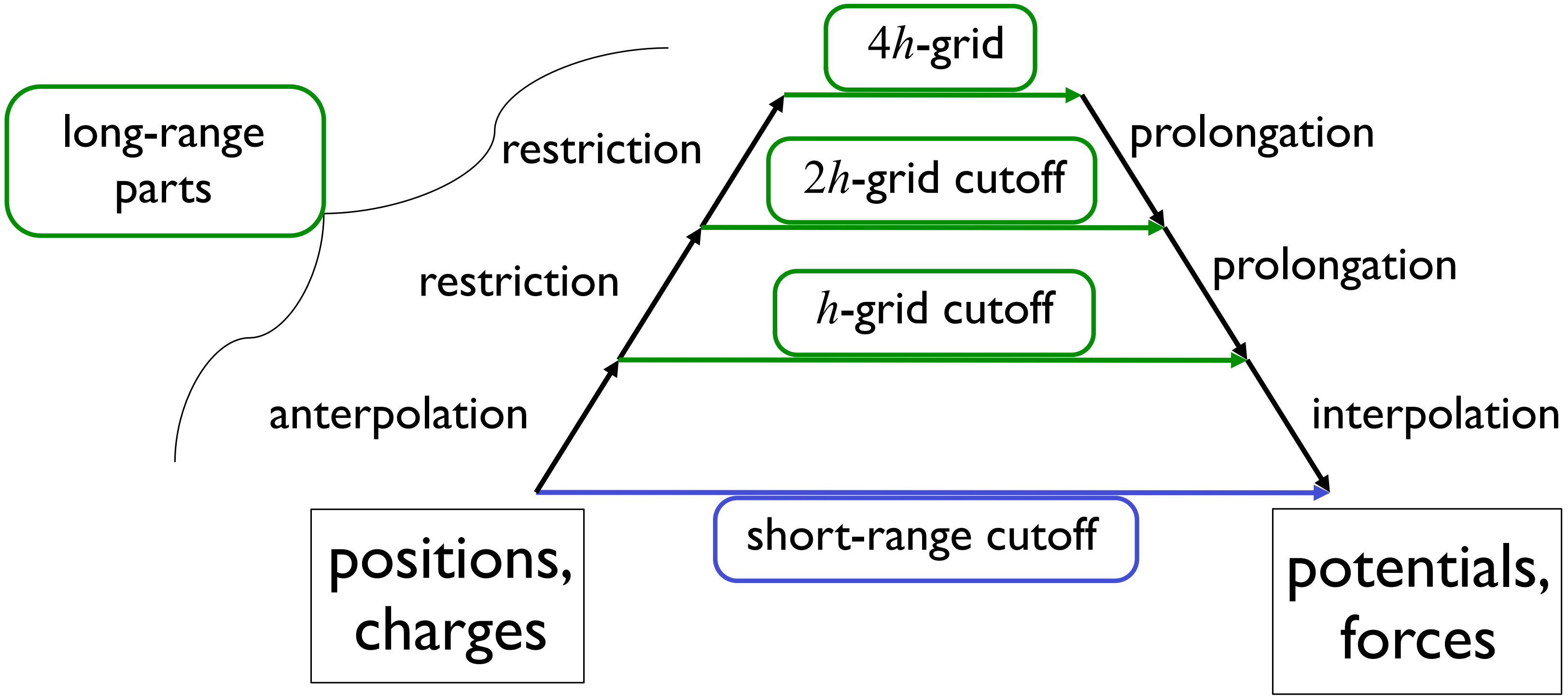
Interpolate the smoothed potentials



MSM Calculation

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

Computational Steps

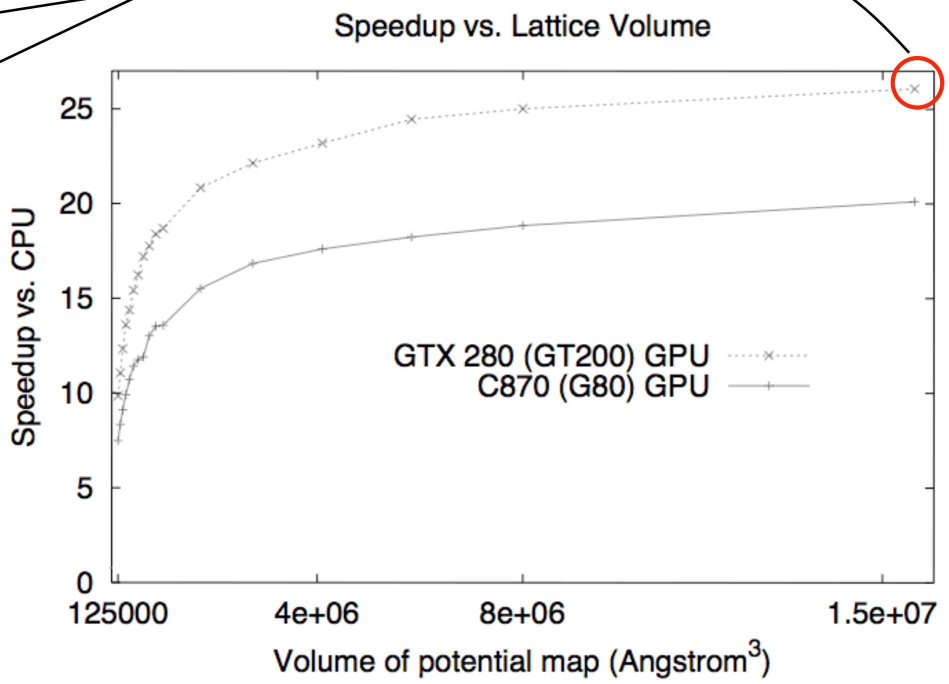


Calculating Electrostatic Potential Maps with MSM on GPUs

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.

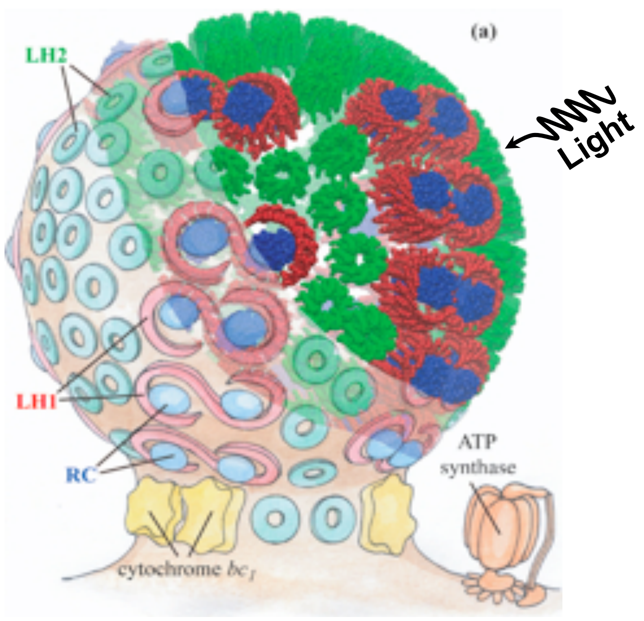
Computational steps	CPU (s)	w/ GPU (s)	Speedup
Short-range cutoff	480.07	14.87	32.3
Long-range anterpolation	0.18		
restriction	0.16		
lattice cutoff	49.47	1.36	36.4
prolongation	0.17		
interpolation	3.47		
Total	533.52	20.21	26.4



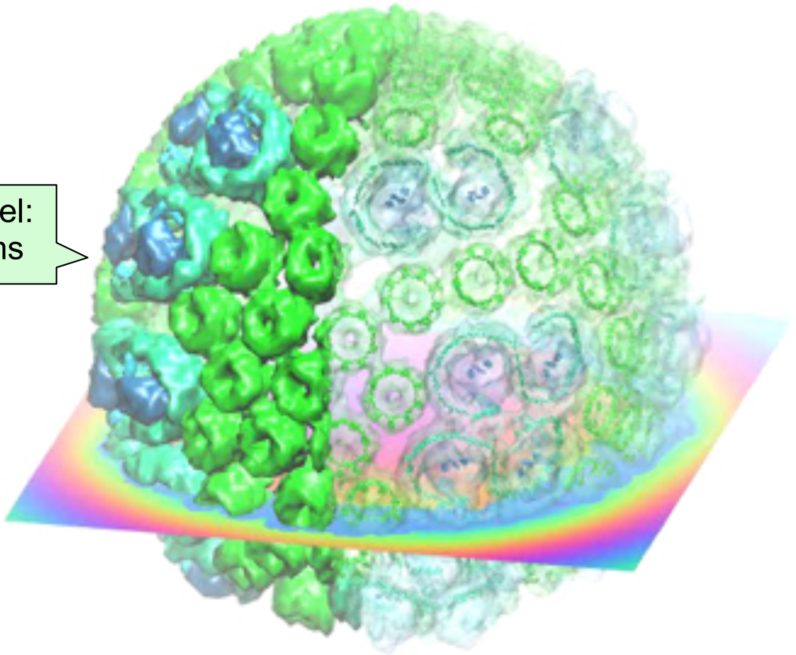
Multilevel summation of electrostatic potentials using graphics processing units.
D. Hardy, J. Stone, K. Schulten. *J. Parallel Computing*, 35:164-177, 2009.

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

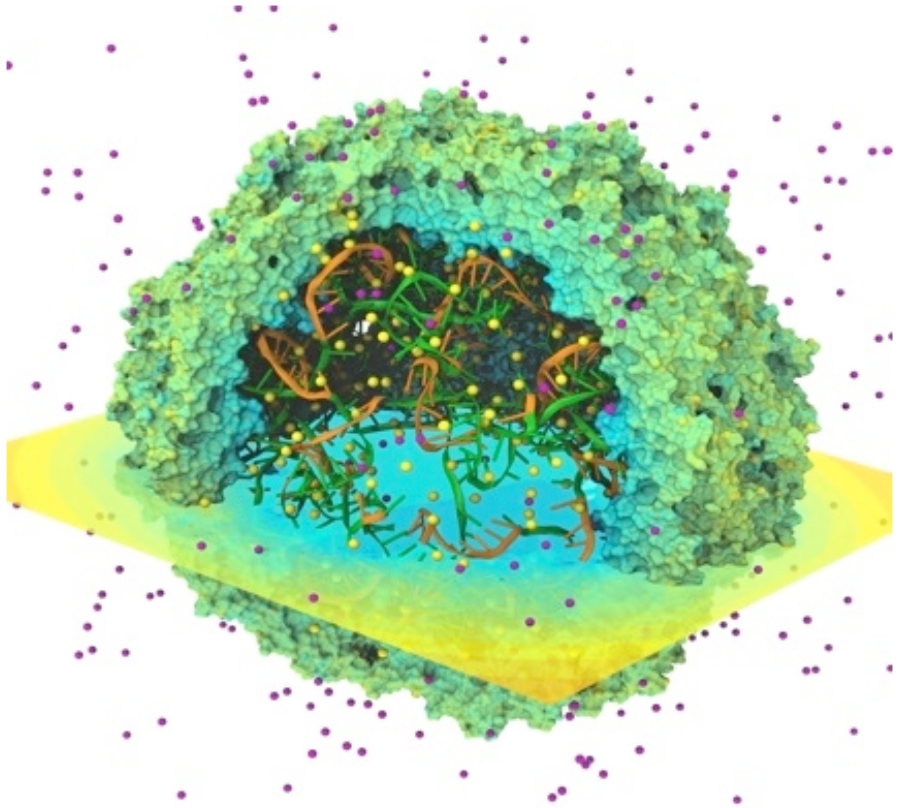


Electrostatic field of chromatophore model from **multilevel summation method**: computed with 3 GPUs (G80) in ~90 seconds, 46x faster than single CPU core in 1 hr, 10 min

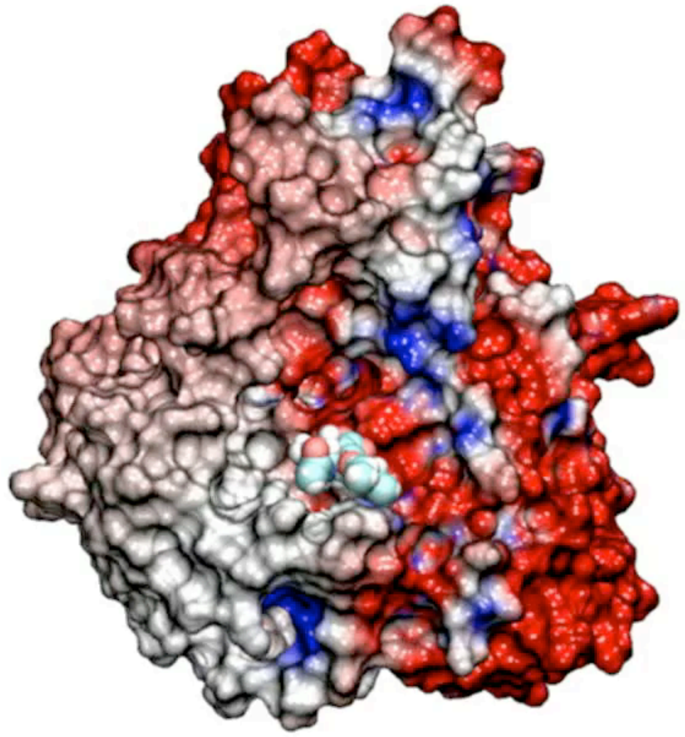
Full chromatophore model will permit structural, chemical and kinetic investigations at a structural systems biology level

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)

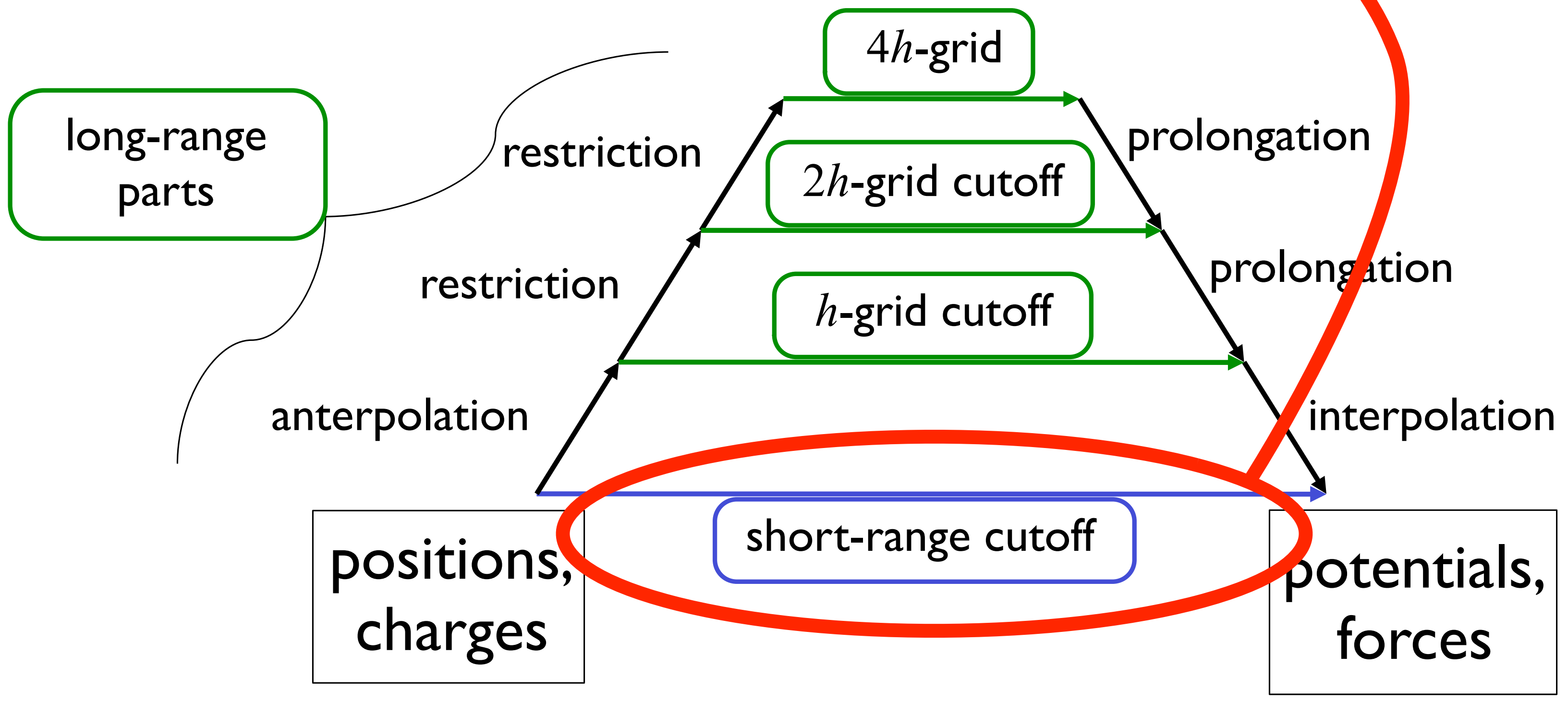


Investigation of drug (Tamiflu) resistance of the “swine” flu virus demanded **fast response!** Calculating electrostatics for 20,000 trajectory frames, 27.8 hour job reduced to 1.1 hours (Linux workstation with Quadro 5800)

MSM Calculation

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

Computational Steps

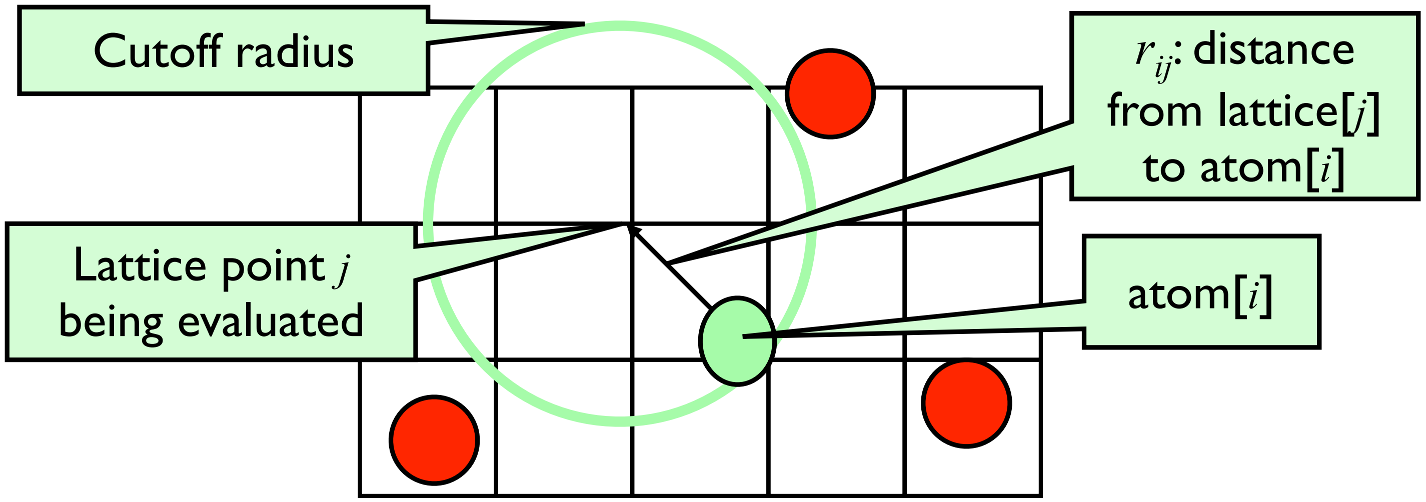


Short-range Cutoff Summation

- Potential at each lattice point is summed from atoms within cutoff distance:

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

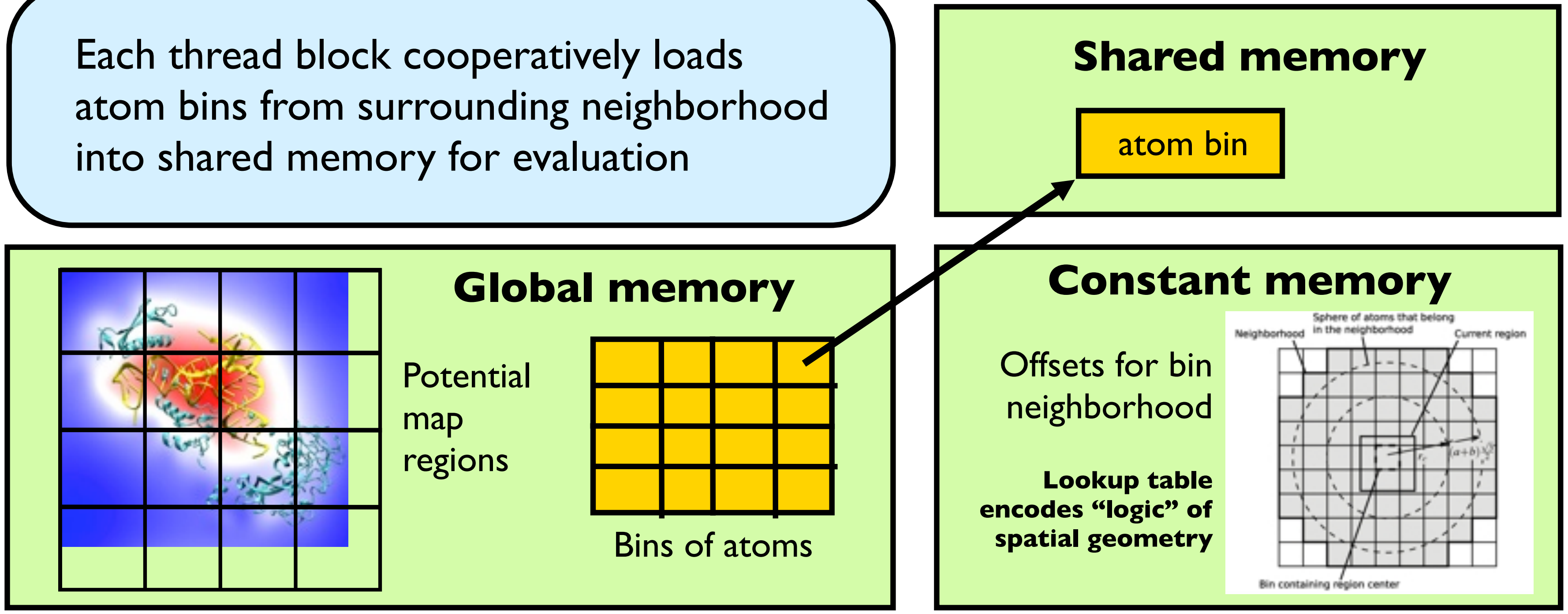
- Smoothing function $s(r)$ is algorithm dependent


Cutoff radius
r_{ij}: distance from lattice[j] to atom[i]
Lattice point j being evaluated
atom[i]

Short-range Cutoff Summation on GPUs

- Thread blocks are assigned to regions of lattice points
- Each thread sums potential to its assigned lattice point(s) from the nearby atoms
- Atoms [x/y/z/q] are are spatially hashed into fixed-size bins for **memory coalesced reads**
- Neighborhood of bins determined by a **lookup table of offsets in constant memory**
- Threads loop over bin neighborhood **caching each bin to shared memory** and looping over its atoms

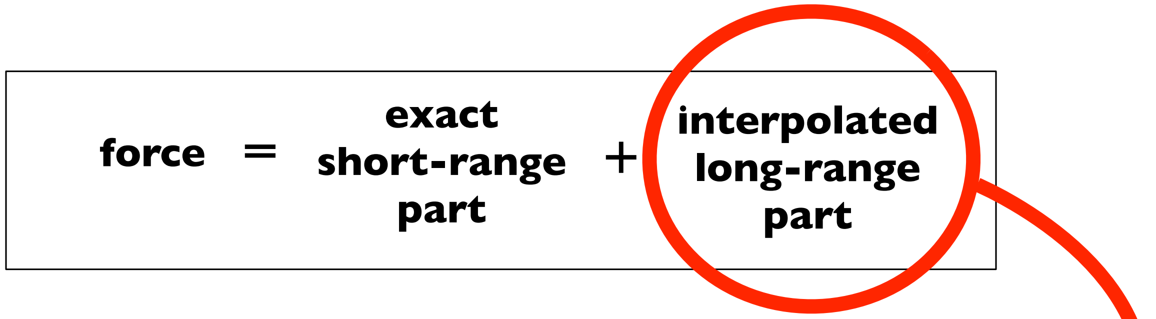
Each thread block cooperatively loads atom bins from surrounding neighborhood into shared memory for evaluation


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
Neighborhood
Sphere of atoms that belong in the neighborhood
Current region
r_c
$(a+b)\sqrt{3}$
Bin containing region center

Using CPU to Improve GPU Performance

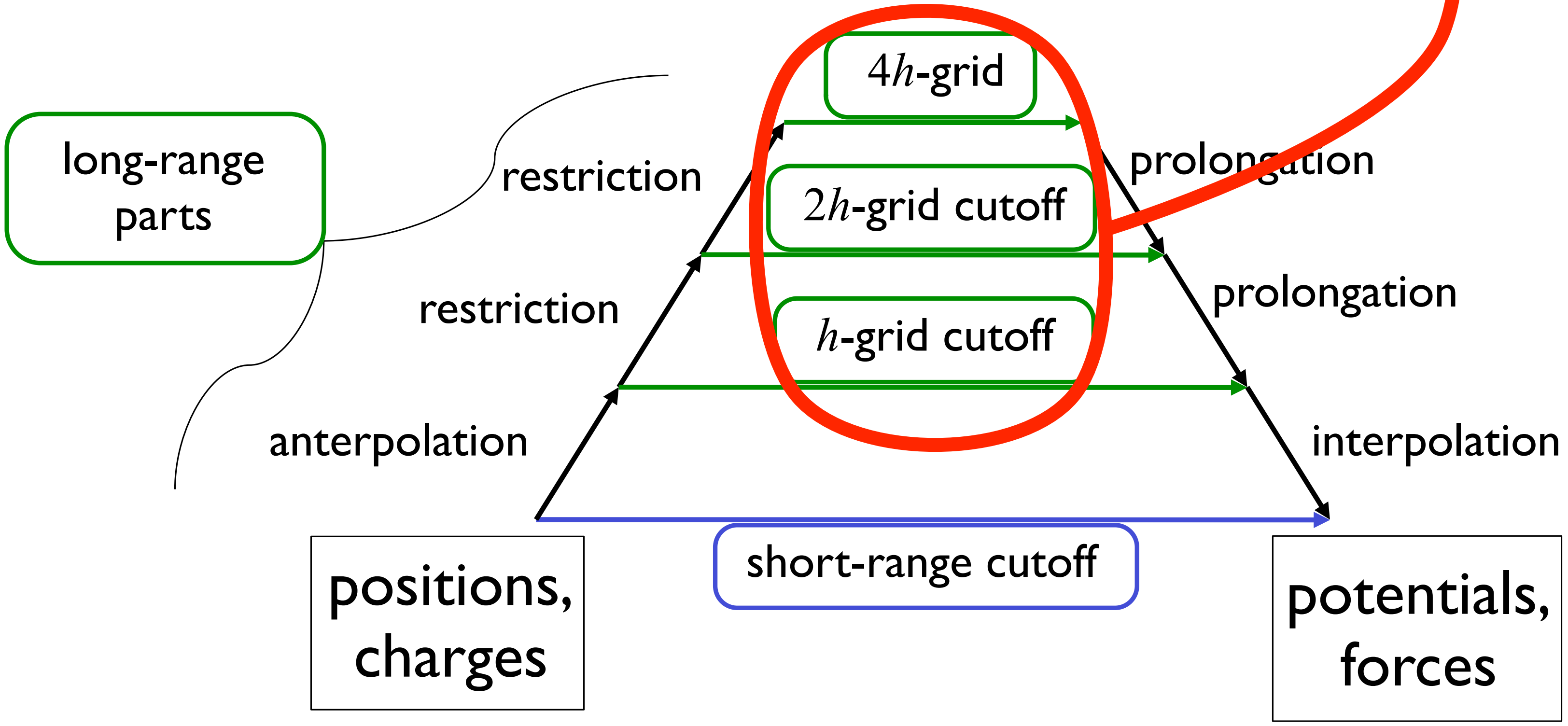
- GPU performs best when the work evenly divides into the number of threads / processing units
- Optimization strategy:
  - Use the CPU to regularize the GPU workload
  - Use fixed-size bin data structures, with "empty" slots skipped or producing zeroed out results
  - Handle exceptional or irregular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

MSM Calculation



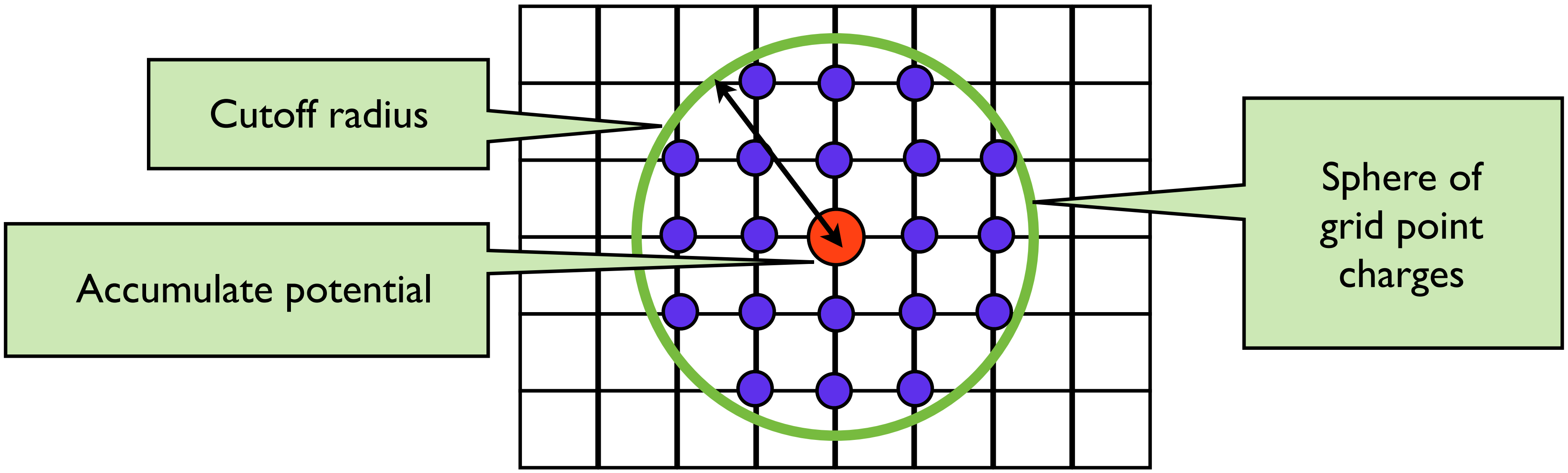
force = exact short-range part + interpolated long-range part

Computational Steps



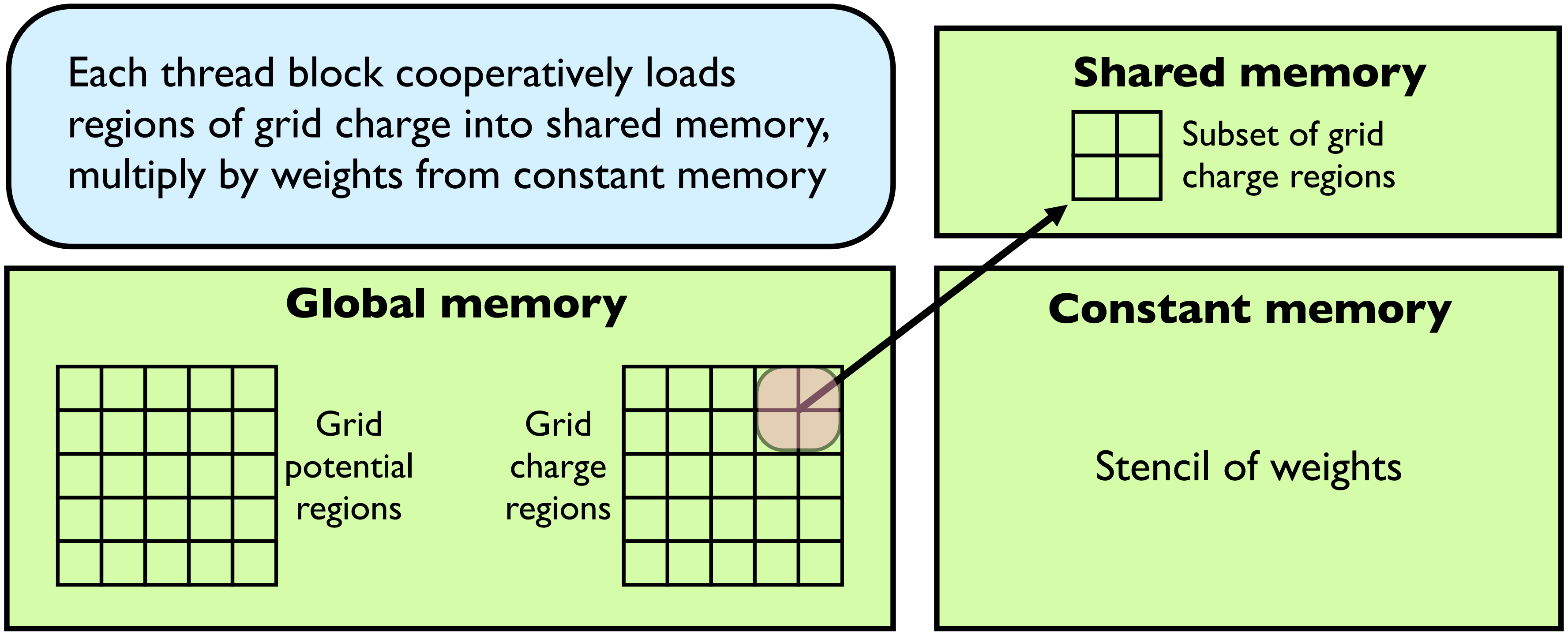
Lattice Cutoff Summation

- 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



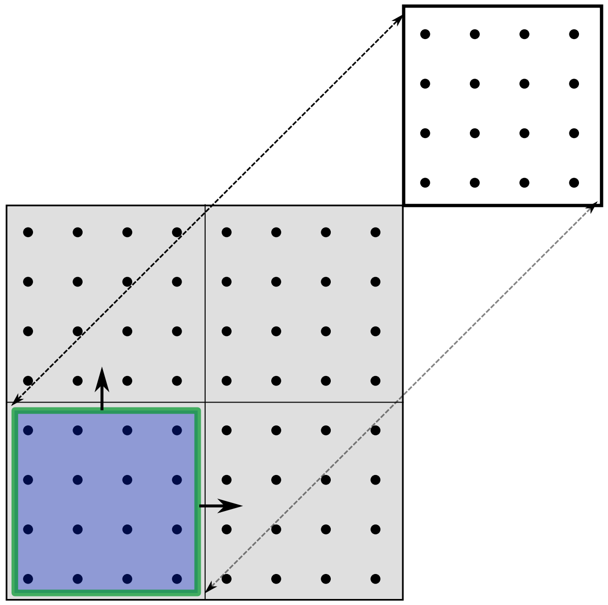
Lattice Cutoff Summation on GPU

- Thread blocks assigned to 4x4x4 region of grid potentials
- Each thread sums potential to its assigned grid point from the nearby grid charges
- Grid point regions are stored contiguously for **memory coalesced reads**
- Stencil of **weights stored in constant memory** (padded up to next multiple of 4)
- Threads loop over surrounding regions of charge, **caching cube of 8 regions into shared memory**


Each thread block cooperatively loads regions of grid charge into shared memory, multiply by weights from constant memory
Shared memory
Subset of grid charge regions
Global memory
Grid potential regions
Grid charge regions
Constant memory
Stencil of weights

Access 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



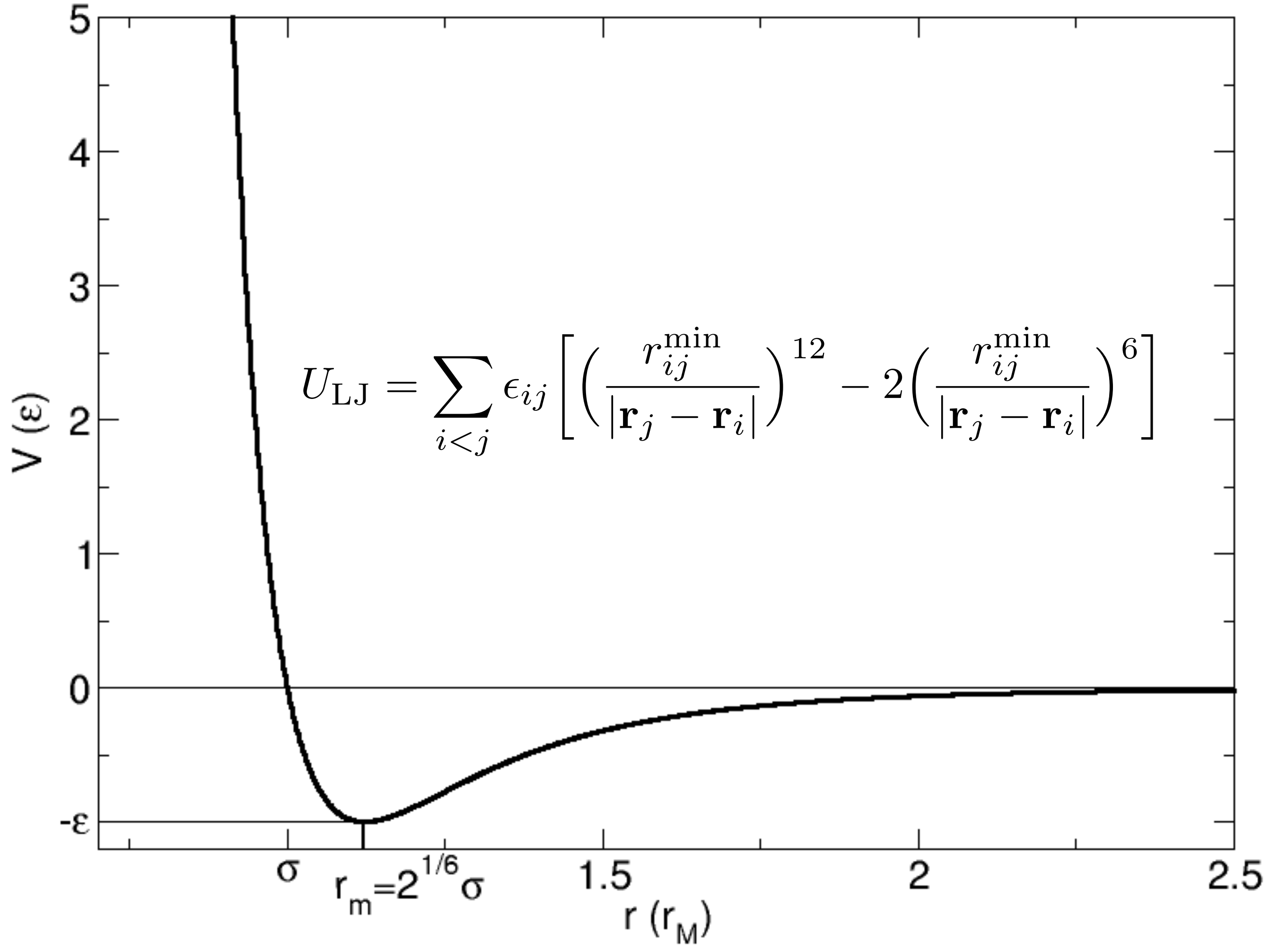
Calculating Grid Levels Concurrently

- The amount of work drops by a factor of 1/8 at each higher level
- Keep the GPU occupied by calculating all grid levels concurrently
    - Pack all regions over all levels into 1D array (with padding at edges of grid)
    - Store map of level array offsets in constant memory
    - Each thread scales its resulting potential by its scaling factor corresponding to its grid level

Calculating Electrostatic Forces with MSM on GPUs

- Grid cutoff calculation remains the same
- Short-range interactions between atoms
  - Choose bin volume based on thread block size
  - Atom [x/y/z/q] data are good size for GPU memory access
  - Neighborhood of bins determined by lookup table of offsets in constant memory
  - Threads loop over bin neighborhood caching each bin to shared memory and looping over its atoms
  - CPU handles atoms that overflow the fixed-size bins
- Produces great performance... *however...*

Need to Calculate Lennard-Jones with Electrostatics



$$U_{\mathrm{LJ}} = \sum_{i<j} \epsilon_{ij} \left[\left(\frac{r_{ij}^{\min}}{|\mathbf{r}_j - \mathbf{r}_i|} \right)^{12} - 2 \left(\frac{r_{ij}^{\min}}{|\mathbf{r}_j - \mathbf{r}_i|} \right)^{6} \right]$$

Model requires excluding pairs of atoms that are covalently bonded to each other or to a common atom

How to Handle Excluded Interactions

- Ignore them; add all interactions and subtract later
  - Works only if you can exactly reproduce the value and are not affected by floating point cancellation (e.g. D.E. Shaw's Anton computer)
- Detect and do not add
  - Use pairlists
  - Search an exclusion table (e.g. NAMD's GPU kernel)
- Other approaches?

Three Approaches That Don’t Quite Work

- Carrying an exclusion bit mask with the atom data
  - [x/y/z/q/emin/rmin/id/mask]
  - Excluded atoms generally have IDs further apart than 32
- Impose an upper bound on the magnitude of forces (“force clamping”)
  - Gives low accuracy results due to steep LJ function
- Impose a lower bound on the pairwise distance
  - Produces correct results but shifts too much work onto the CPU cleanup (GPU is faster by about a factor of 7)

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