Multilevel Summation Method in NAMD and VMD

David Hardy
http://www.ks.uiuc.edu/Research/gpu/
NAIS: State-of-the-Art Algorithms for Molecular Dynamics

Multilevel Summation Method

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

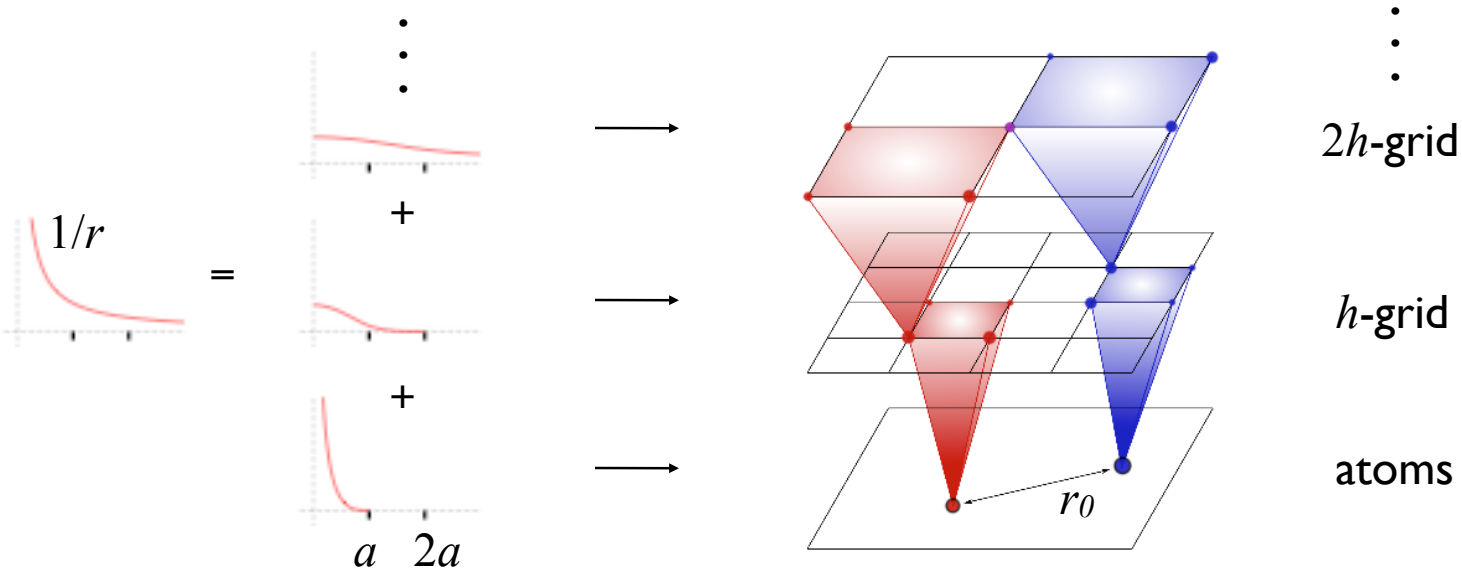
- Fast algorithm for N-body electrostatics
- Calculates sum of smoothed pairwise potentials interpolated from a hierarchal nesting of grids
- Advantages over PME (particle-mesh Ewald) and/or FMM (fast multipole method):
  - Algorithm has linear time complexity
  - Allows non-periodic or periodic boundaries
  - 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
  - Extends to other types of pairwise interactions (e.g., dispersion forces)

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

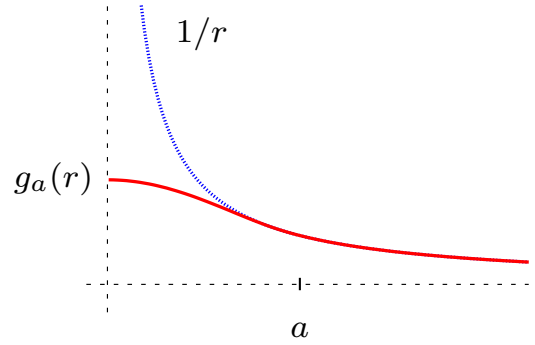
Interpolate the smoothed potentials



Separation

Separation of length scales:

$$\frac{1}{r} = \underbrace{\left(\frac{1}{r} - g_a(r)\right)}_{\textit{short-range}} + \underbrace{g_a(r)}_{\textit{smooth}}$$



Requirements for smoothing function:

- $g_a(r) = 1/r$ for $r \geq a$, short-range part vanishes beyond cutoff
- $g_a(\sqrt{x^2 + y^2 + z^2})$ and derivatives are slowly varying everywhere
- g_a has sufficient continuity

Approximation

Approximate smooth part on 3D grid with spacing h:

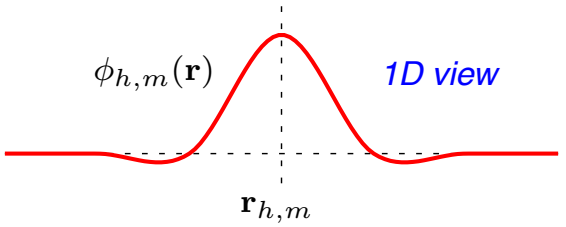
$$g_a(\mathbf{r},\mathbf{r}') \approx \sum_m \phi_{h,m}(\mathbf{r})\, g_a(\mathbf{r}_{h,m},\mathbf{r}') \qquad \textit{interpolate source}$$

$$\approx \sum_m \phi_{h,m}(\mathbf{r}) \left(\sum_n \phi_{h,n}(\mathbf{r}')\, g_a(\mathbf{r}_{h,m},\mathbf{r}_{h,n}) \right) \qquad \textit{interpolate destination}$$

$$= \sum_m \sum_n \phi_{h,m}(\mathbf{r})\, g_a(\mathbf{r}_{h,m},\mathbf{r}_{h,n})\, \phi_{h,n}(\mathbf{r}')$$

Nodal basis function $\phi_{h,m}$:

- continuously differentiable
- local support

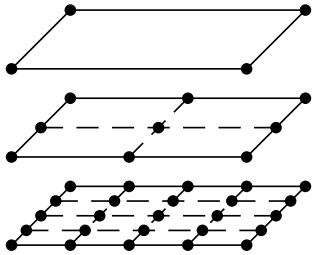


Hierarchy

Recursively apply *separation* and *approximation* using hierarchy of grids.

Separate smooth part: $$g_a(r) = \Big(g_a(r) - g_{2a}(r)\Big) + g_{2a}(r)$$

- $g_a(r) - g_{2a}(r)$ vanishes for $r \geq 2a$
- g_{2a} is more slowly varying than g_a



Approximate g_{2a} on 3D grid of spacing $2h$:

$$g_{2a}(\mathbf{r}_{h,m}, \mathbf{r}_{h,n}) \approx \sum_i \sum_j \phi_{2h,i}(\mathbf{r}_{h,m})\, g_{2a}(\mathbf{r}_{2h,i}, \mathbf{r}_{2h,j})\, \phi_{2h,j}(\mathbf{r}_{h,n})$$

Double *cutoff* and *grid spacing* at each new grid level.

Matrix Formulation

$$U = \frac{1}{2}\sum_i \sum_{j \neq i} \frac{q_i q_j}{\|\mathbf{r}_j - \mathbf{r}_i\|} = \frac{1}{2}\mathbf{q}^{\mathsf{T}} G \mathbf{q}$$

Separation: $G = \hat{G} + \tilde{G}$

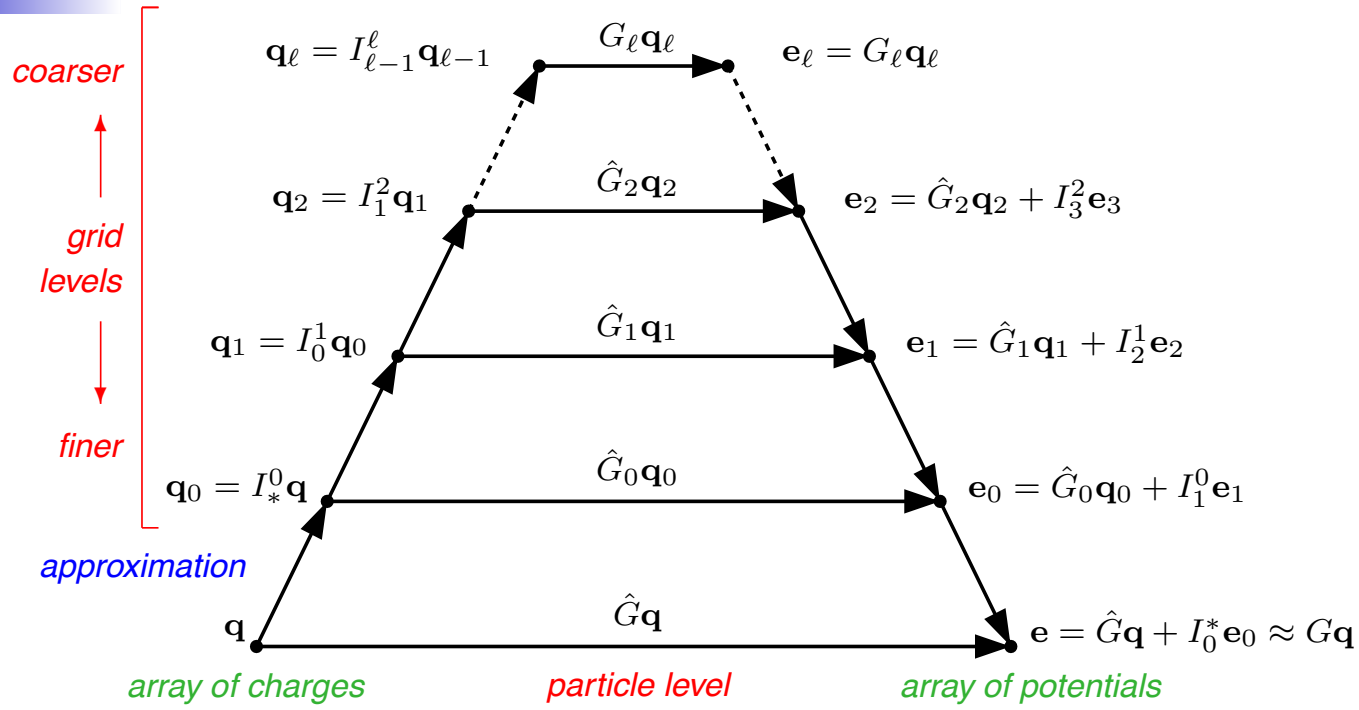
$$\tilde{G}_{ij} = g_a(\|\mathbf{r}_j - \mathbf{r}_i\|), \qquad \hat{G}_{ij} = \begin{cases} \|\mathbf{r}_j - \mathbf{r}_i\|^{-1} - g_a(\|\mathbf{r}_j - \mathbf{r}_i\|), & \text{for } i \neq j, \\ -g_a(\|\mathbf{r}_j - \mathbf{r}_i\|), & \text{otherwise,} \end{cases}$$

Approximation: $\tilde{G} \approx I_h^* G_h I_*^h$

$$(G_h)_{mn} = g_a(\|\mathbf{r}_{h,n} - \mathbf{r}_{h,m}\|), \qquad (I_h^*)_{im} = \phi_{h,m}(\mathbf{r}_i), \qquad I_*^h = (I_h^*)^{\mathsf{T}}$$

Hierarchy: $G \approx \hat{G} + I_h^* \left(\hat{G}_h I_*^h + I_{2h}^h (G_{2h} I_h^{2h} I_*^h) \right)$

Multilevel Algorithm

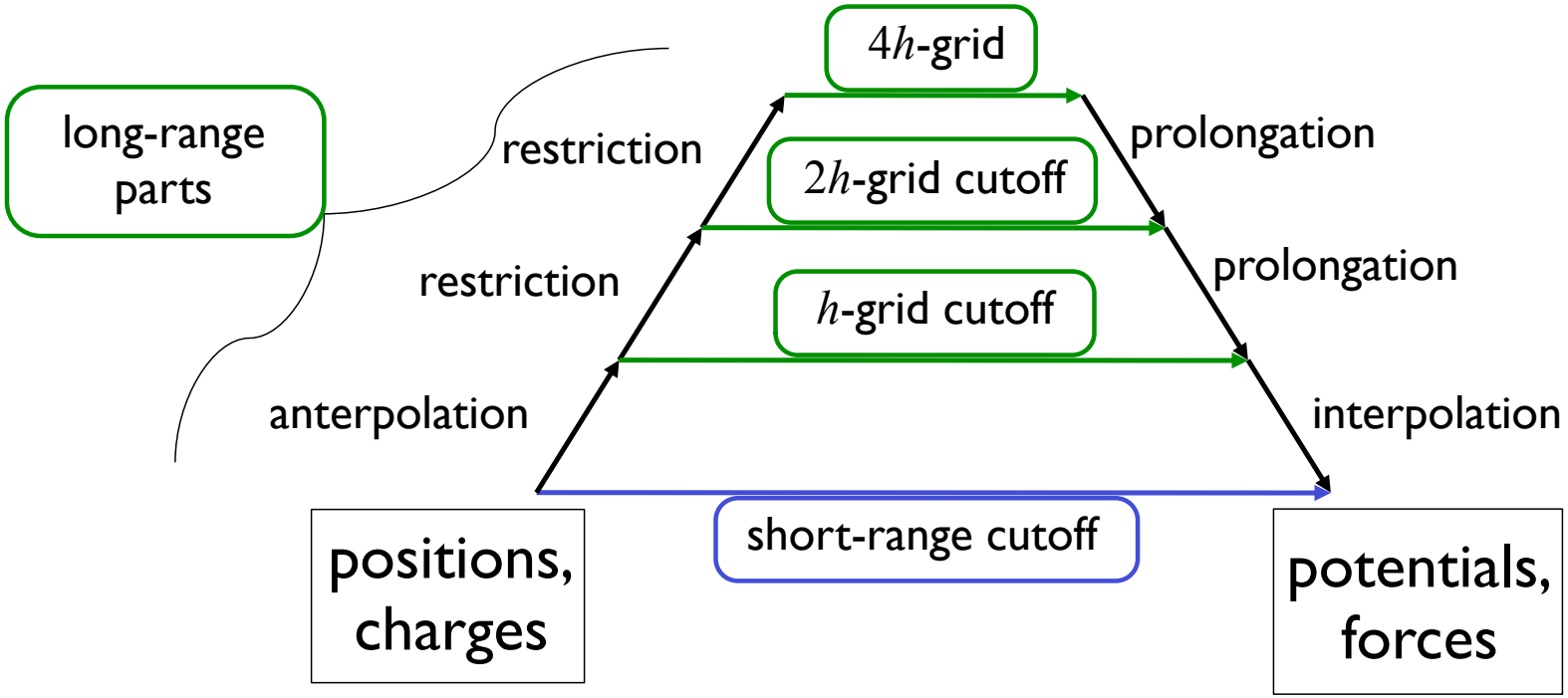


Computational work requires $O(N)$ operations.

MSM Calculation

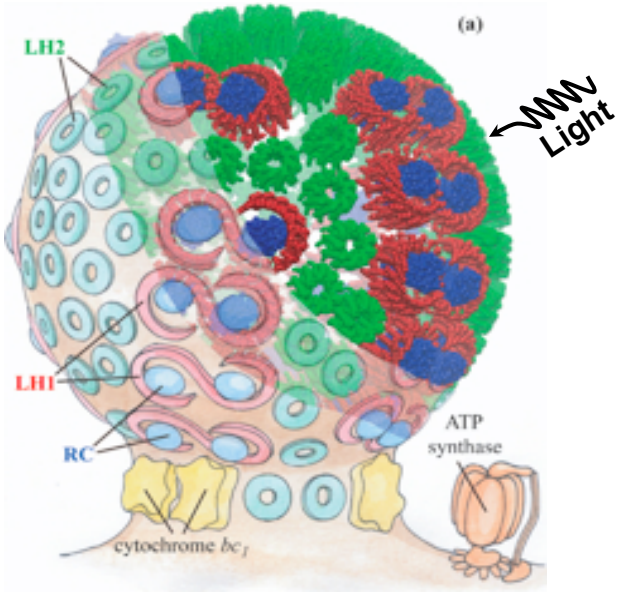
force = **exact short-range part** + **interpolated long-range part**

Computational Steps

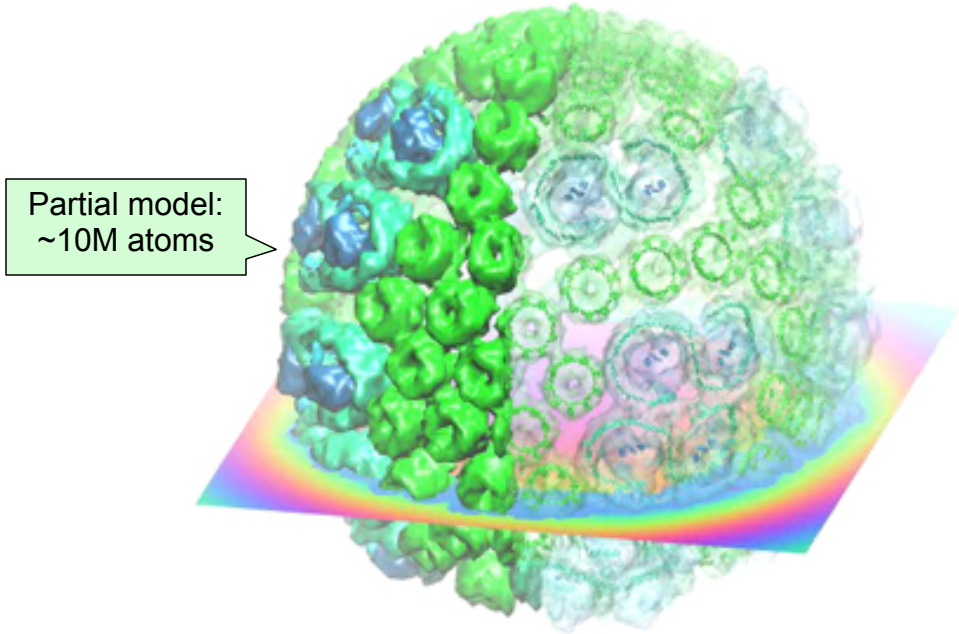


Application of MSM in VMD to Photosynthesis

Investigations of the chromatophore, a photosynthetic organelle



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

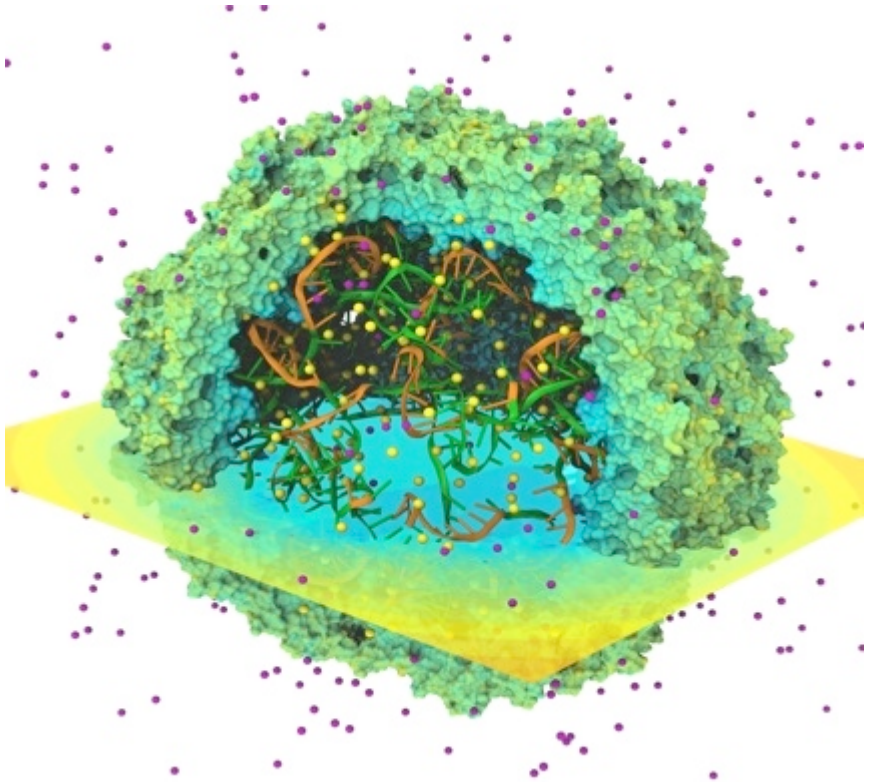


Electrostatic field of chromatophore model from **multilevel summation method**: computed with 3 GPUs (G80) in ~90 seconds, 46x faster than single CPU core in 1 hr, 10 min

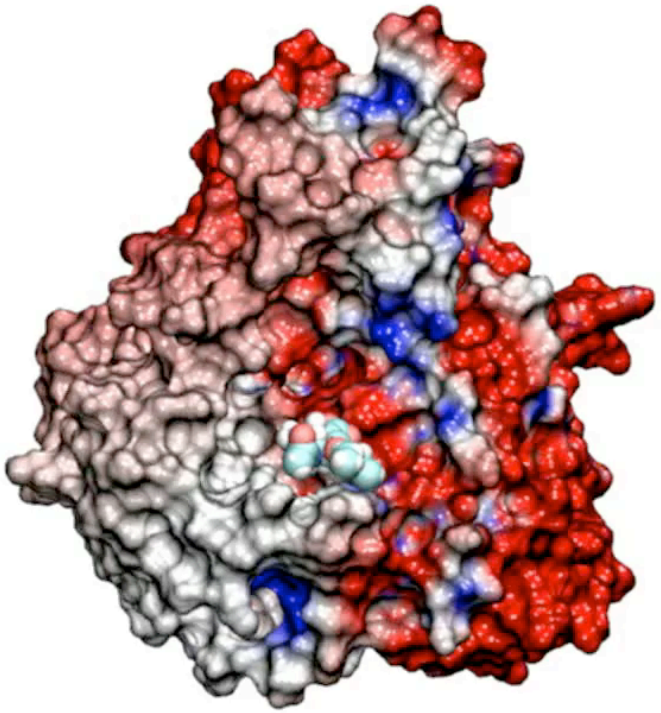
Full chromatophore model will permit structural, chemical and kinetic investigations at a structural systems biology level

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)



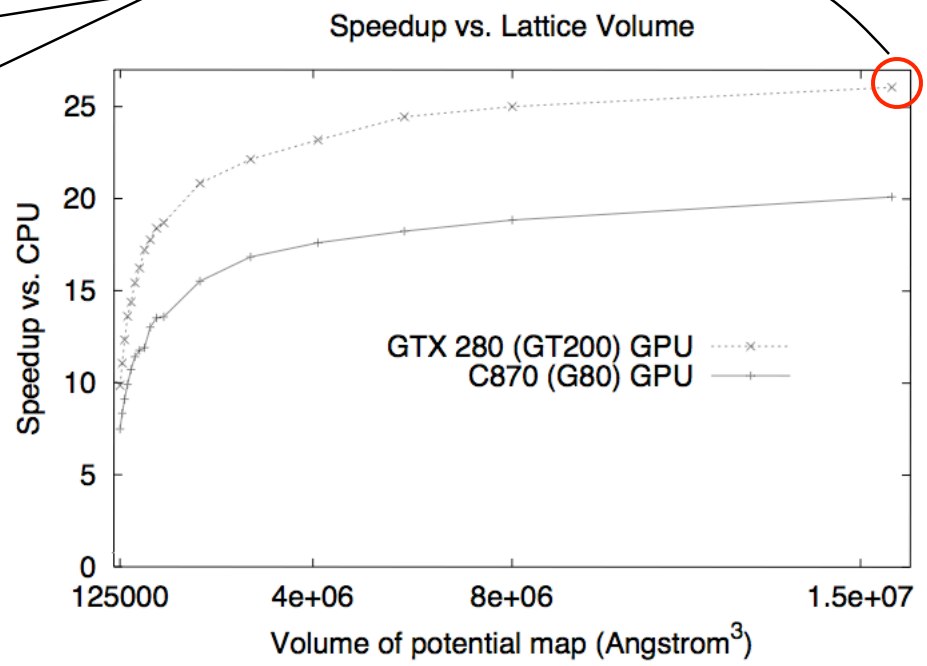
Investigation of drug (Tamiflu) resistance of the "swine" flu virus demanded **fast response!**
Calculating electrostatics for 20,000 trajectory frames, 27.8 hour job reduced to 1.1 hours
(Linux workstation with Quadro 5800)

MSM Potentials on the GPU

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

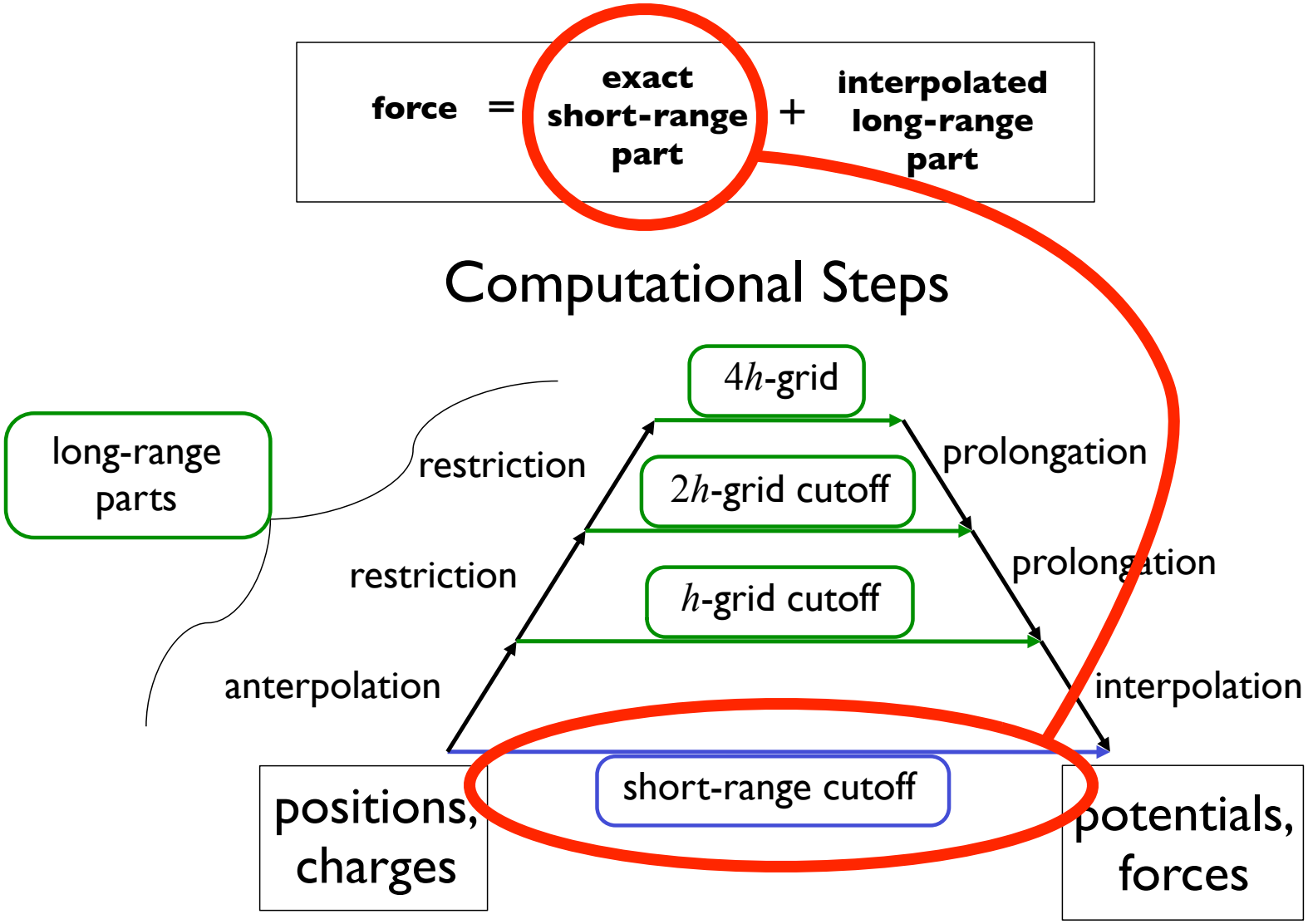
Performance profile for 0.5 Å map of potential for 1.5 M atoms.
Hardware platform is Intel QX6700 CPU and NVIDIA GTX 280.

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

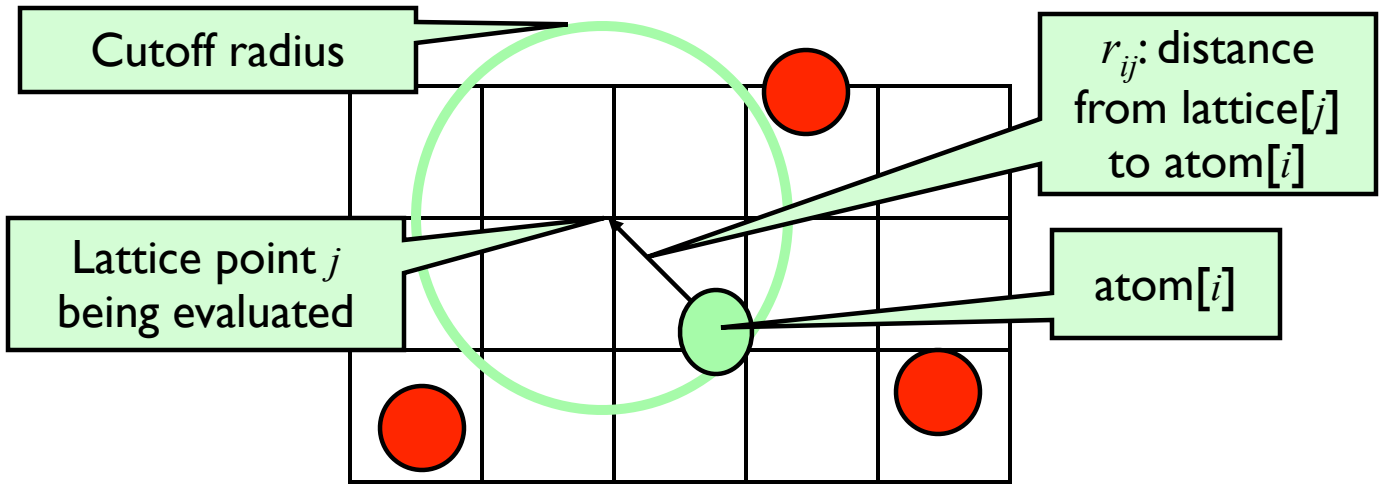


Short-range Cutoff Summation

- 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



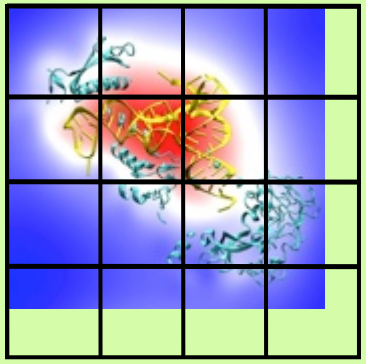
Short-range Cutoff Summation on GPU

- Atoms are spatially hashed into fixed-size bins
- CPU handles overflowed bins (GPU kernel can be aggressive)
- GPU thread block calculates corresponding region of potential map
- Bin/region neighbor checks costly; solved with universal lookup table

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

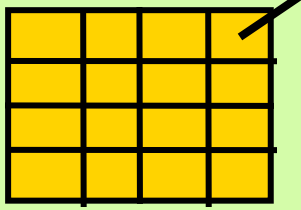
Shared memory

atom bin



Global memory

Potential map regions



Bins of atoms

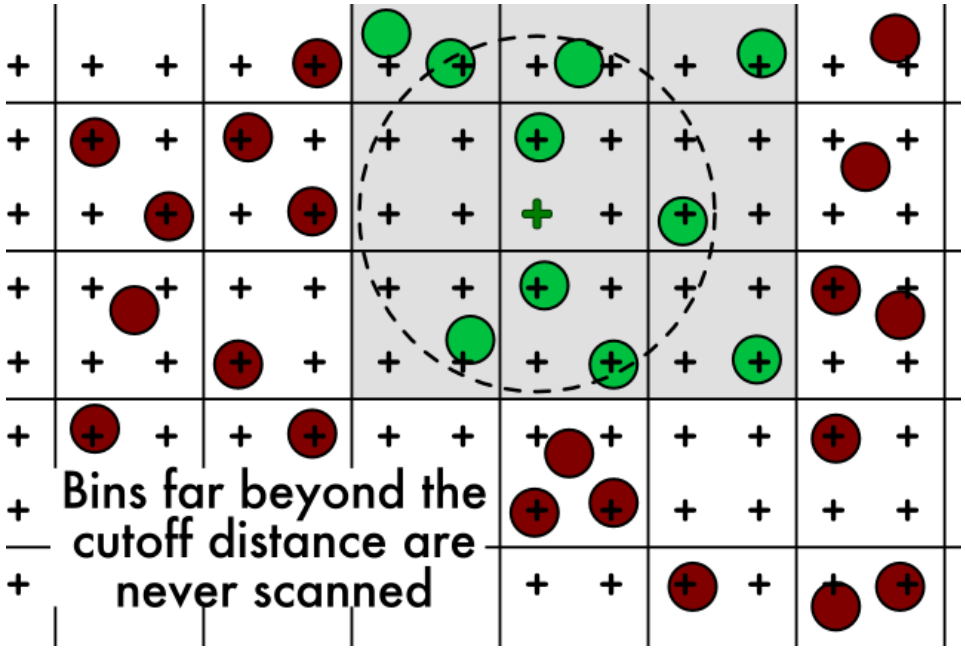
Constant memory

Offsets for bin neighborhood

Lookup table encodes "logic" of spatial geometry

Spatial Sorting of Atoms Into Bins

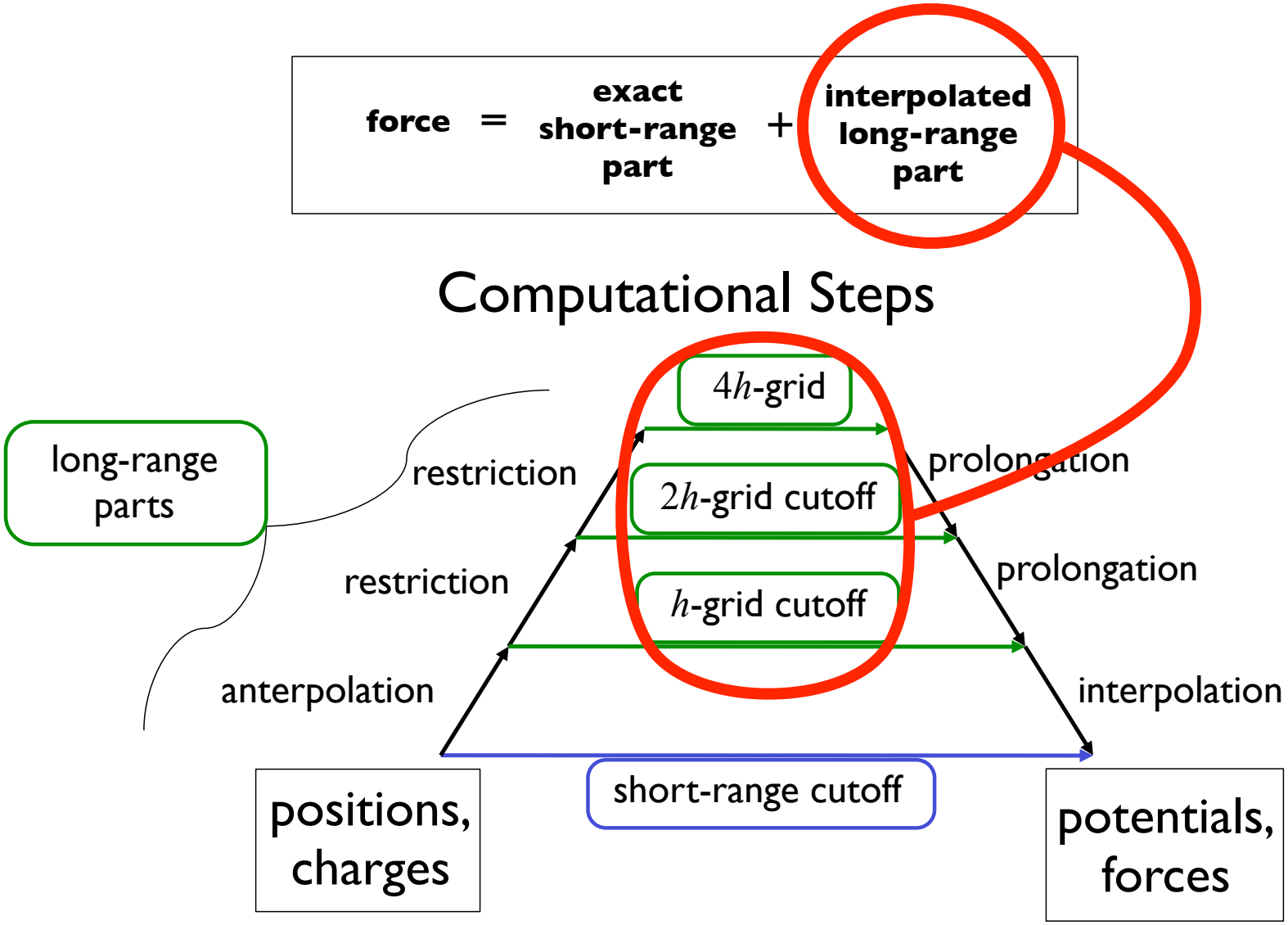
- Sort atoms into bins by their coordinates
- Each bin is sized to guarantee GPU **memory coalescing**
- Each bin holds up to 8 atoms, containing 4 FP values (coords, charge)
- Each lattice point **gathers** potentials from atom bins within cutoff



Using CPU to Improve GPU Performance

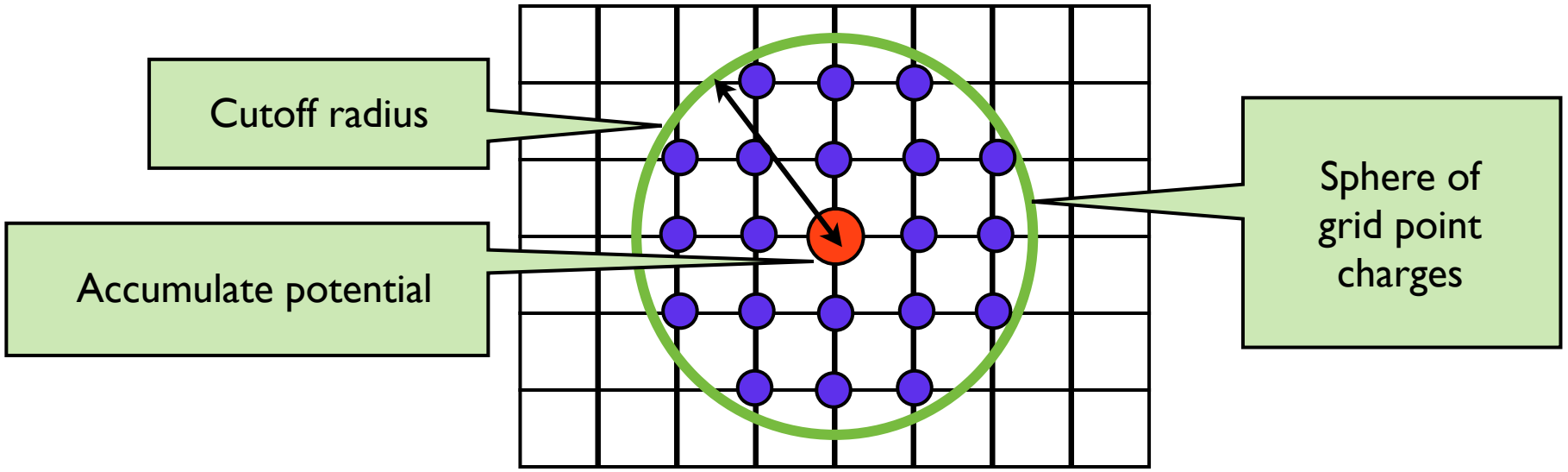
- GPU performs best when the work evenly divides into the number of threads / processing units
- Optimization strategy:
  - Use the CPU to "regularize" the GPU workload
  - Use fixed size bin data structures, with "empty" slots skipped or producing zeroed out results
  - Handle exceptional or irregular 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



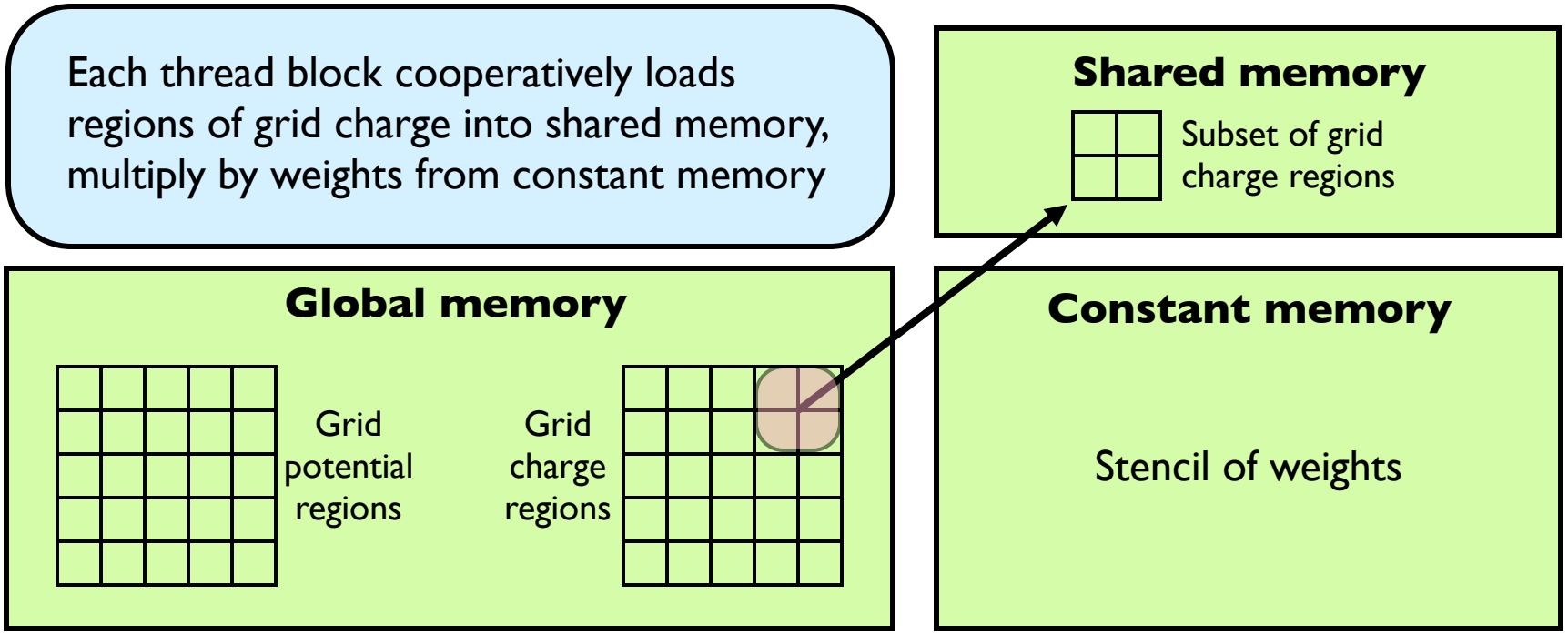
Lattice Cutoff Summation

- 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