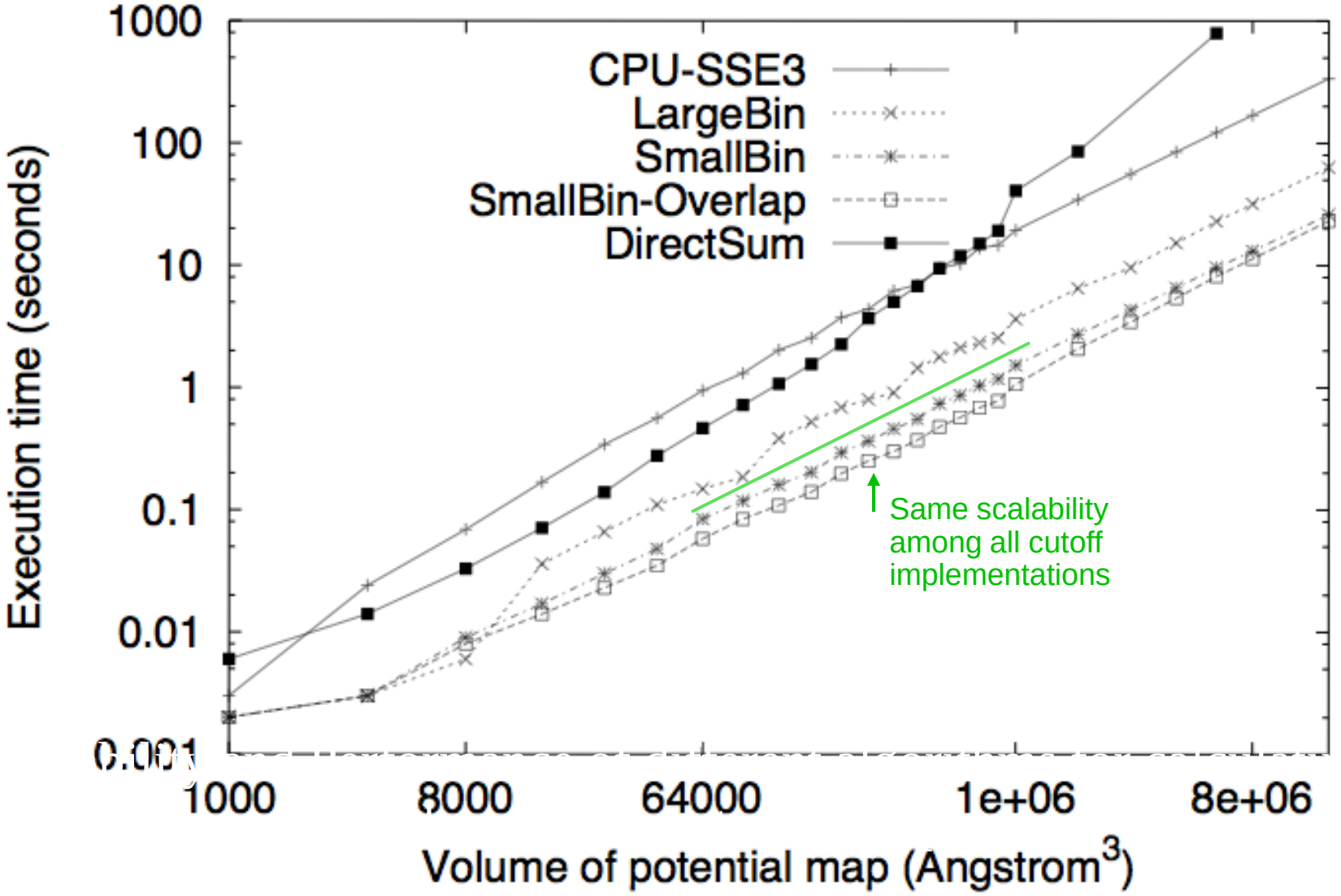
VSCSE Summer School

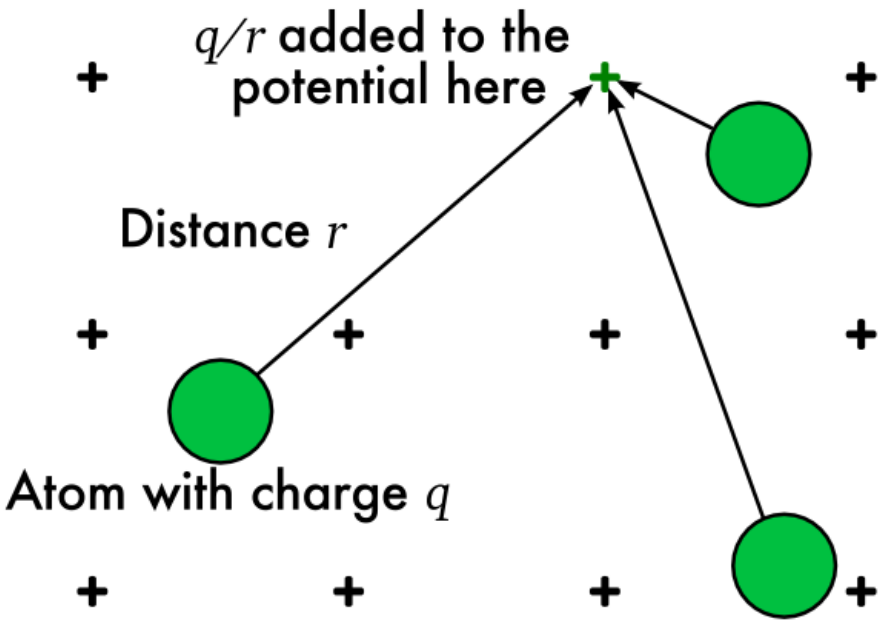
Proven Algorithmic Techniques for Many-core Processors

Lecture 4: Cut-off and Binning for Regular Data Sets

Direct Summation is accurate but has poor data scalability

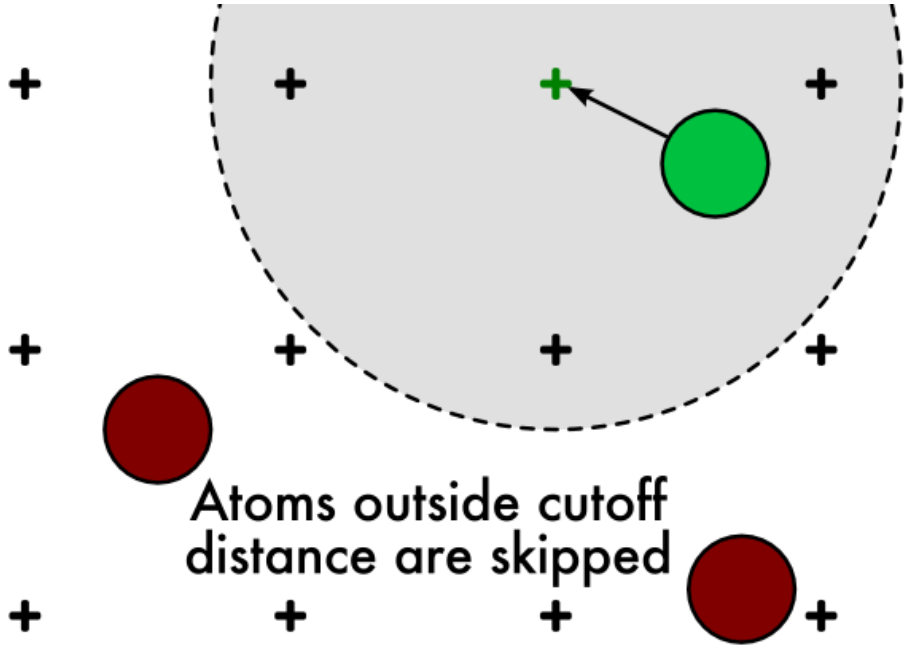


DCS Algorithm for Electrostatic Potentials



- At each grid point, sum the electrostatic potential from all atoms
- Highly data-parallel
- But has quadratic complexity
  - Number of grid points × number of atoms
  - Both proportional to volume

Algorithm for Electrostatic Potentials With a Cutoff



- Ignore atoms beyond a *cutoff distance*, r_c
  - Typically 8Å–12Å
  - Long-range potential may be computed separately
- Number of atoms within cutoff distance is roughly constant (uniform atom density)
  - 200 to 700 atoms within 8Å–12Å cutoff sphere for typical biomolecular structures

Cut-off Summation

- With fixed partial charge q_i, electrostatic potential V at position r over all N atoms:

$$V(\vec{r};\vec{r}_1,\vec{r}_2,\ldots,\vec{r}_N) = \sum_{i=1}^{N} \frac{q_i}{4\pi\varepsilon_0|\vec{r}-\vec{r}_i|} s(|\vec{r}-\vec{r}_i|)$$

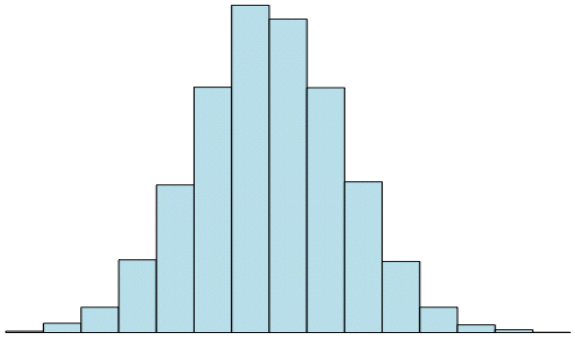
$$s(r) = \begin{cases} (1-r^2/r_c^2)^2, & \text{if } r < r_c, \\ 0, & \text{otherwise} \end{cases}$$

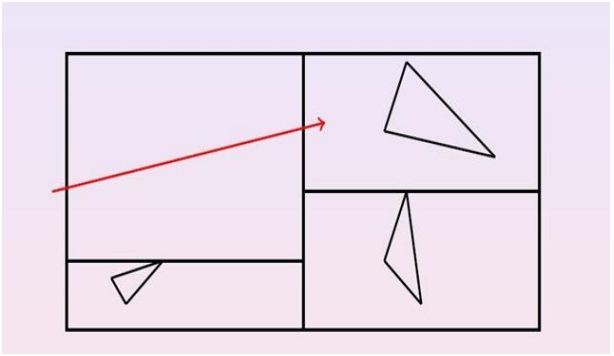
Implementation Challenge

- For each tile of grid points, we need to identify the set of atoms that need to be examined
  - One could naively examine all atoms and only use the ones whose distance is within the given range (but this examination still takes time)
  - We need to avoid examining the atoms outside the range

Binning

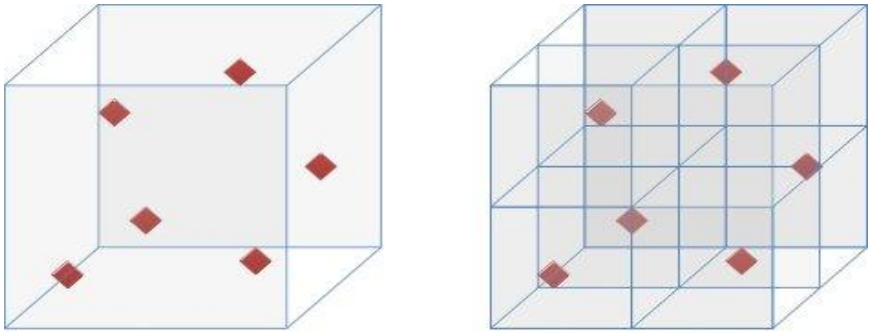
- A process that groups data to form a chunk called *bin*
- Each bin collectively represents a property for data in the bin
- Helps problem solving due to data coarsening
- Histogram, KD Tree, …





Binning with uniform cube

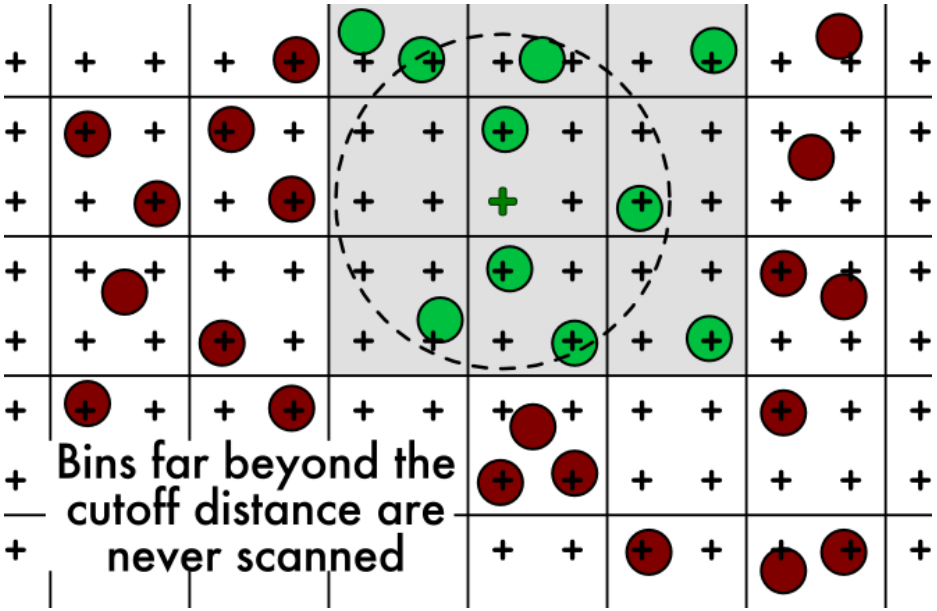
- Divide the simulation volume with non-overlapping cubes
- Every atom in the simulation volume falls into a cube based on its spatial location
- After binning, each cube has unique index in the simulation space



(a) Simulation volume

(b) Simulation volume with eight bins

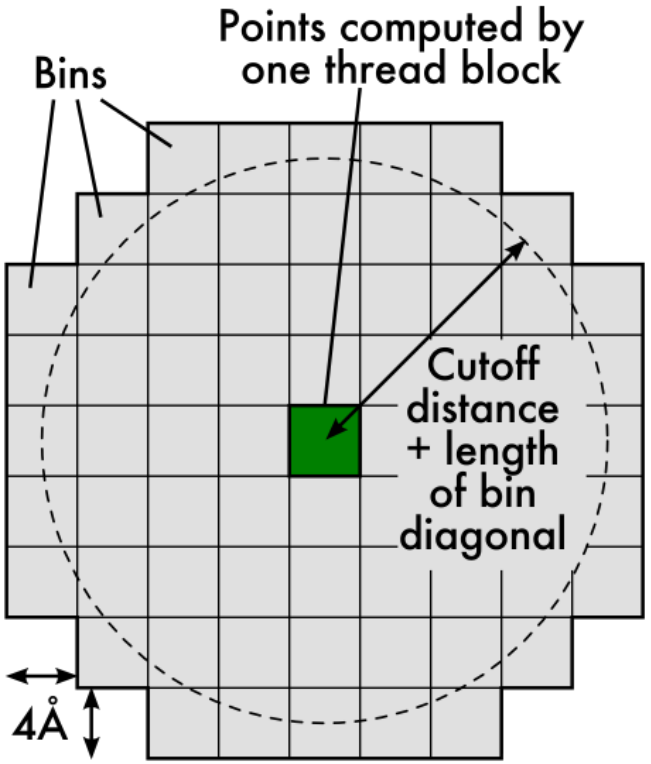
Spatial Sorting Using Binning



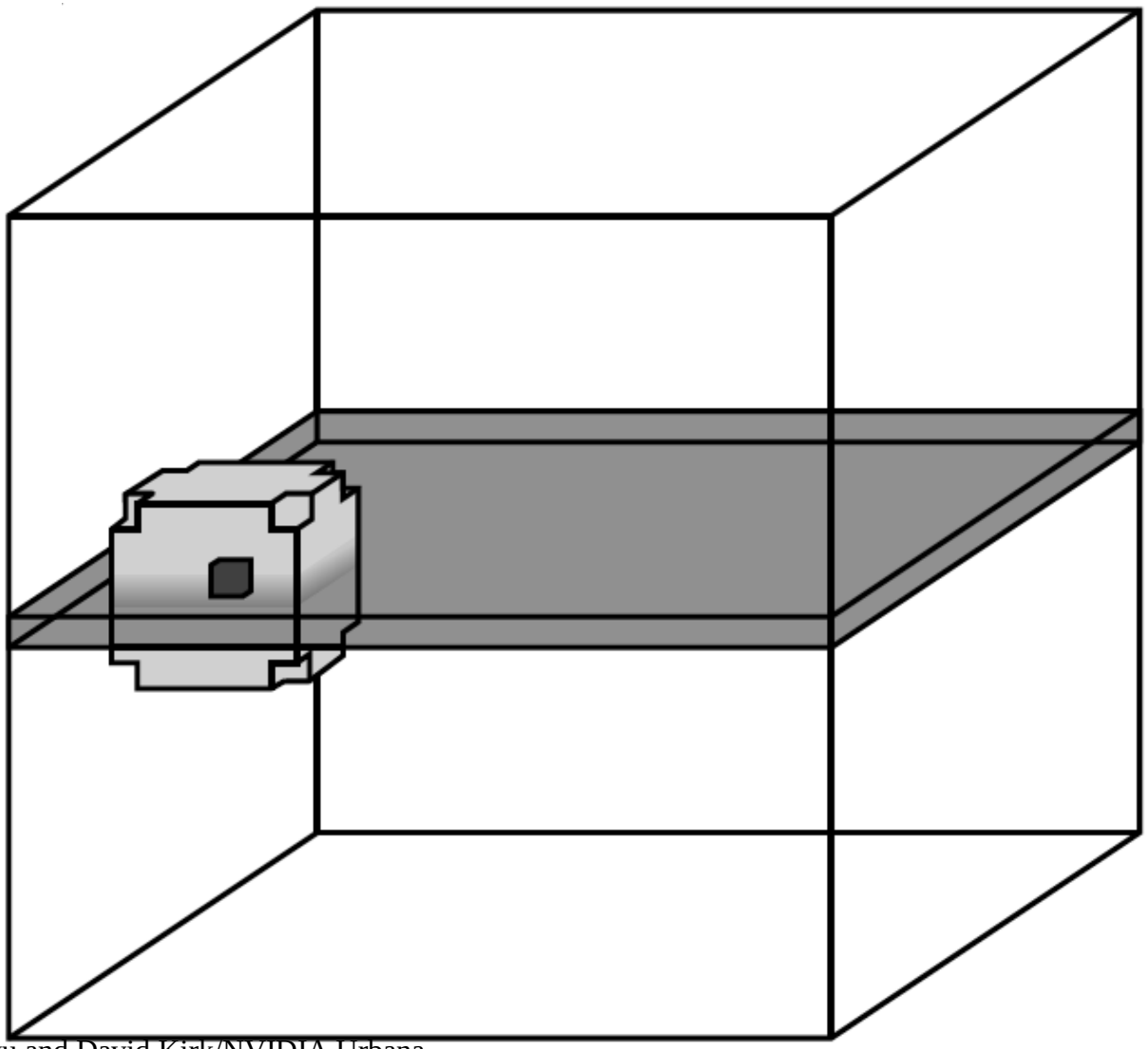
- Presort atoms into *bins* by location in space
- Each bin holds several atoms
- Cutoff potential only uses bins within r_c
  - Yields a linear complexity cutoff potential algorithm

Improving Work Efficiency

- Thread block examines atom bins up to the cutoff distance
  - Use a sphere of bins
  - All threads in a block scan the same atoms
    - No hardware penalty for multiple simultaneous reads of the same address
    - Simplifies fetching of data

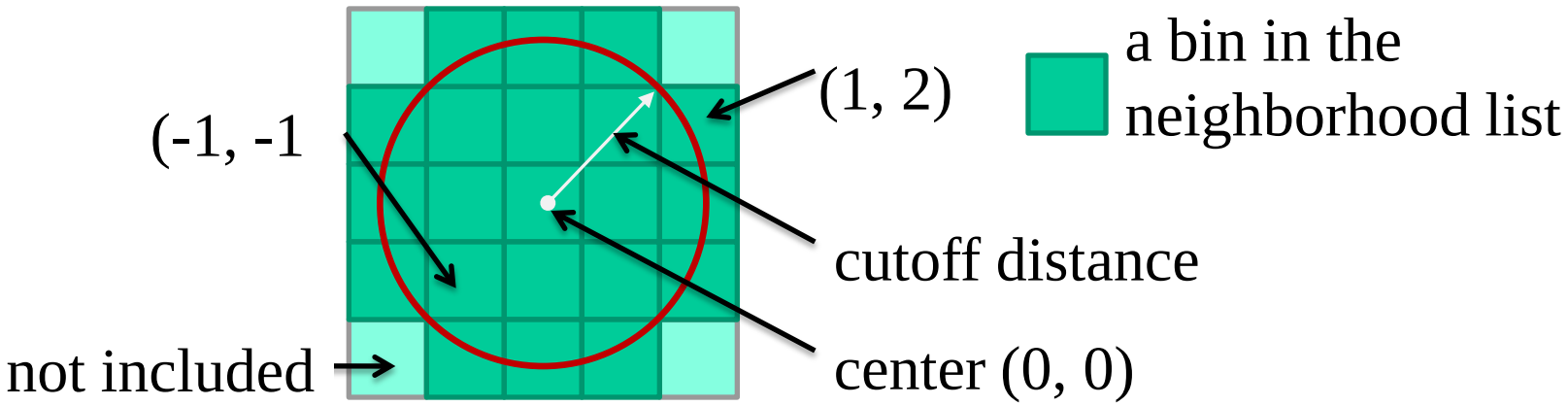


The Neighborhood is a volume



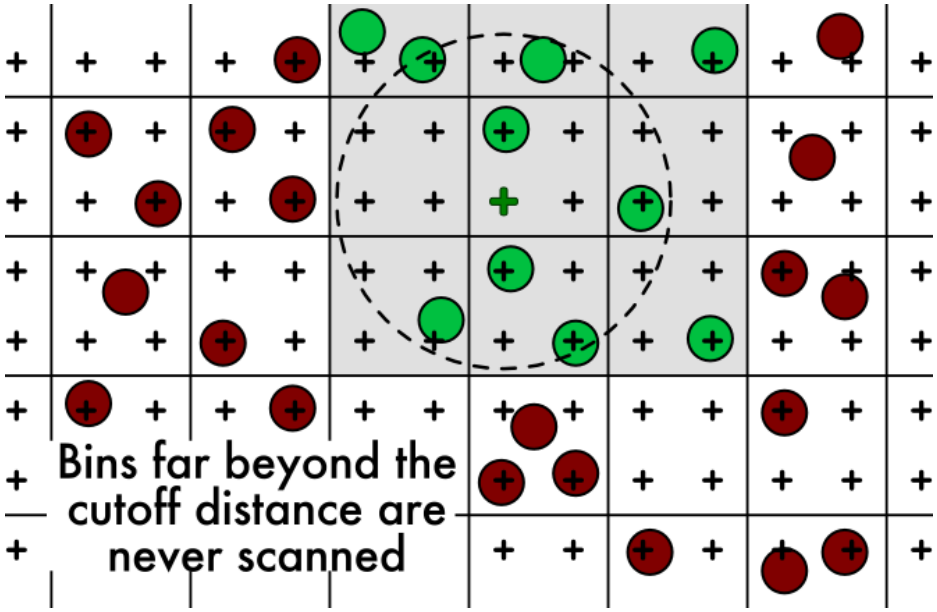
Neighborhood Offset List

- A list of offsets enumerating the bins that are located within the cutoff distance for a given location in the simulation volume
- Detection of surrounding atoms becomes realistic for output grid points
  - By visiting bins in the neighborhood offset list and iterating atoms they contain



Bin Design

- Uniform sized bins allows array implementation
- Bin size (capacity) should be big enough to contain all the atoms that fall into a bin
  - Cut-off will screen away atoms that weren't processed
  - Performance penalty if too many are screened away



Pseudo Code

```
// 1. binning
for each atom in the simulation volume,
    index_of_bin := atom.addr / BIN_SIZE
    bin[index_of_bin] += atom

// 2. generate the neighborhood offset list
for each c from -cutoff to cutoff,
    if distance(0, c) < cutoff,
        nlist += c
```

CPU

```
// 3. do the computation
for each point in the output grid,
    index_of_bin := point.addr / BIN_SIZE
    for each offset in nlist,
        for each atom in bin[index_of_bin + offset],
            point.potential += atom.charge / (distance from point to atom)
```

GPU

Performance

- O(MN') where M and N' are the number of output grid points and atoms in the neighborhood offset list, respectively
  - In general, N' is small compared to the number of all atoms
- Works well if the distribution of atoms is uniform

Bin Size Considerations

- Capacity of atom bins needs to be balanced
  - Too large – many dummy atoms in bins
  - Too small – some atoms will not fit into bins
  - Target bin capacity to cover more than 95% or atoms

- CPU places all atoms that do not fit into bins into an overflow bin
  - Use a CPU sequential algorithm to calculate their contributions to the energy grid lattice points.
  - CPU and GPU can do potential calculations in parallel

Going from DCS Kernel to Large Bin Cut-off Kernel

- Adaptation of techniques from the direct Coulomb summation kernel for a cutoff kernel
- Atoms are stored in constant memory as with DCS kernel
- CPU loops over potential map regions that are $(24Å)^3$ in volume (cube containing cutoff sphere)
- Large bins of atoms are appended to the constant memory atom buffer until full, then GPU kernel is launched
- Host loops over map regions reloading constant memory and launching GPU kernels until complete

Large Bin Design Concept

- Map regions are $(24Å)^3$ in volume
- Regions are sized large enough to provide the GPU enough work in a single kernel launch
  - $(48$ lattice points$)^3$ for lattice with 0.5Å spacing
  - Small bins don't provide the GPU enough work to utilize all SMs, to amortize constant memory update time, or kernel launch overhead

Large Bin Cut-off Kernel Code

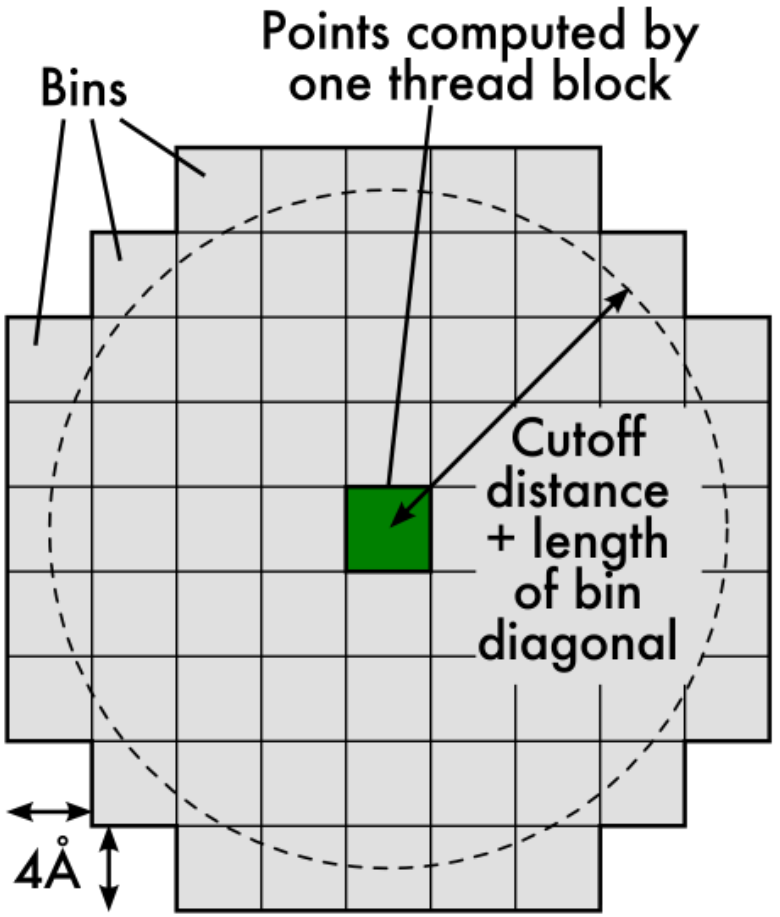
```
static __constant__ float4 atominfo[MAXATOMS];
__global__ static void mgpot_shortrng_energy(…) {
  […]
  for (n = 0;  n < natoms;  n++) {
    float dx = coorx - atominfo[n].x;
    float dy = coory - atominfo[n].y;
    float dz = coorz - atominfo[n].z;
    float q = atominfo[n].w;
    float dxdy2 = dx*dx + dy*dy;
    float r2 = dxdy2 + dz*dz;
    if (r2 < CUTOFF2) {
      float gr2 = GC0 + r2*(GC1 + r2*GC2);
      float r_1 = 1.f/sqrtf(r2);
      accum_energy_z0 += q * (r_1 - gr2);
    }
…
```

Large-bin Cutoff Kernel Evaluation

- 6× speedup relative to fast CPU version
- Work-inefficient
  - Coarse spatial hashing into $(24Å)^3$ bins
  - Only 6.5% of the atoms a thread tests are within the cutoff distance
- Better adaptation of the algorithm to the GPU will gain another 2.5×

Small Bin Design

- For 0.5Å lattice spacing, a $(4Å)^3$ cube of the potential map is computed by each thread block
  - 8×8×8 potential map points
  - 128 threads per block (4 points/thread)
  - 34% of examined atoms are within cutoff distance

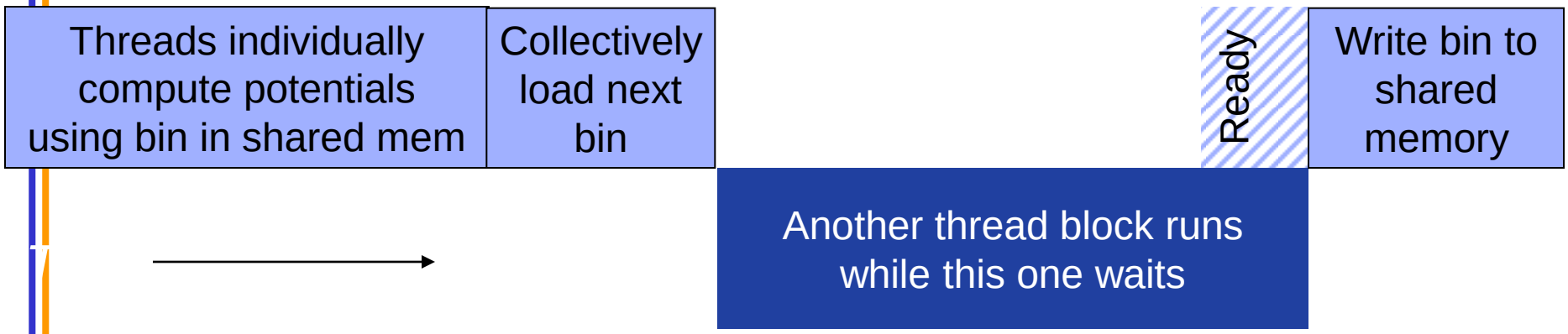


More Design Considerations for the Cutoff Kernel

- High memory throughput to atom data essential
  - Group threads together for locality
  - Fetch bins of data into shared memory
  - Structure atom data to allow fetching
- After taking care of memory demand, optimize to reduce instruction count
  - Loop and instruction-level optimization

Tiling Atom Data

- Shared memory used to reduce Global Memory bandwidth consumption
  - Threads in a thread block collectively load one bin at a time into shared memory
  - Once loaded, threads scan atoms in shared memory
  - Reuse: Loaded bins used 128 times



Coalesced Global Memory Access to Atom Data

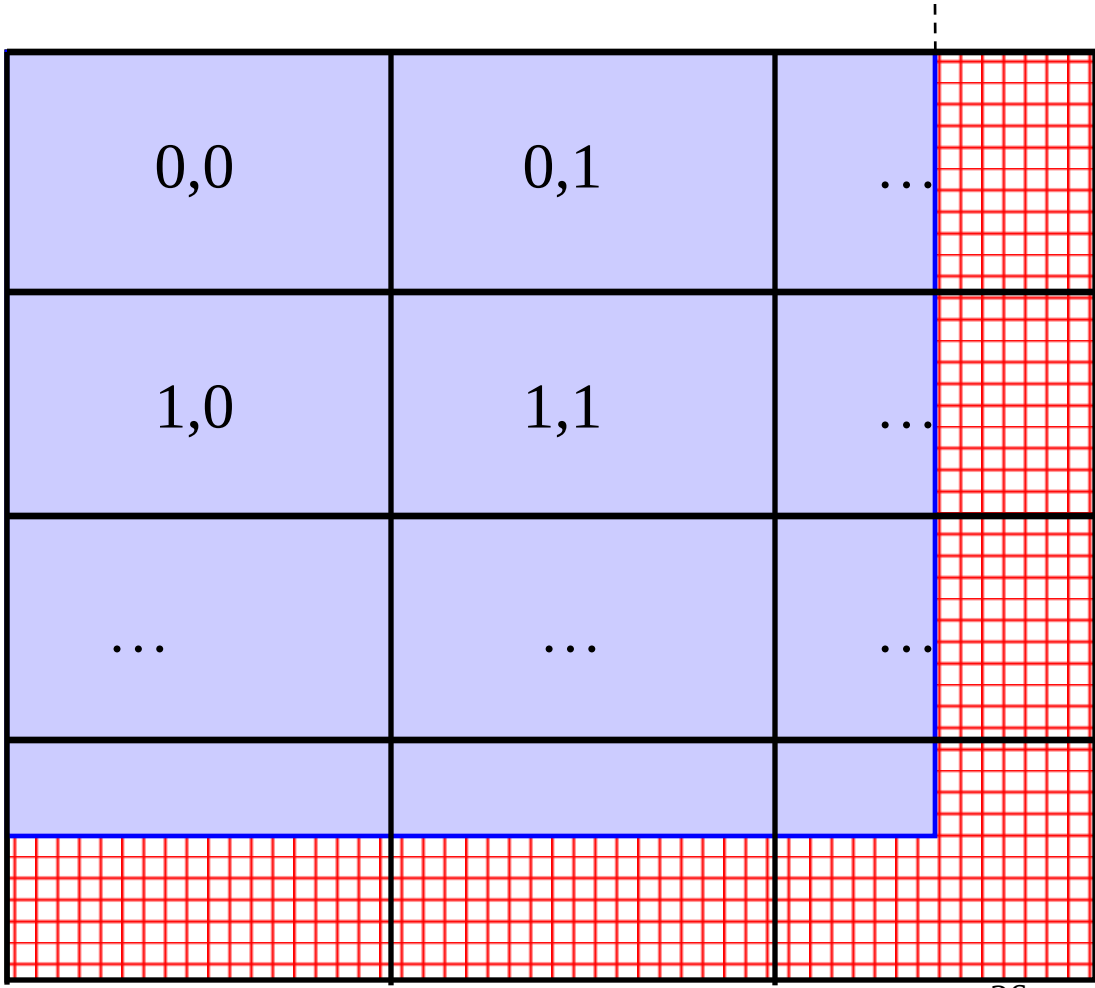
- Full global memory bandwidth only with 64-byte, 64-byte-aligned memory accesses
  - Each bin is exactly 128 bytes
  - Bins stored in a 3D array
  - 32 threads in each block load one bin into shared memory, then processed by all threads in the block
- 128 bytes = 8 atoms (x,y,z,q)
  - Nearly uniform density of atoms in typical systems
    - 1 atom per 10 $Å^3$
  - Bins hold atoms from $(4Å)^3$ of space (example)
  - Number of atoms in a bin varies
    - For water test systems, 5.35 atoms in a bin on average
    - Some bins overfull

Handling Overfull Bins

- In typical use, 2.6% of atoms exceed bin capacity
- Spatial sorting puts these into a list of extra atoms
- Extra atoms processed by the CPU
  - Computed with CPU-optimized algorithm
  - Takes about 66% as long as GPU computation
  - Overlapping GPU and CPU computation yields in additional speedup
  - CPU performs final integration of grid data

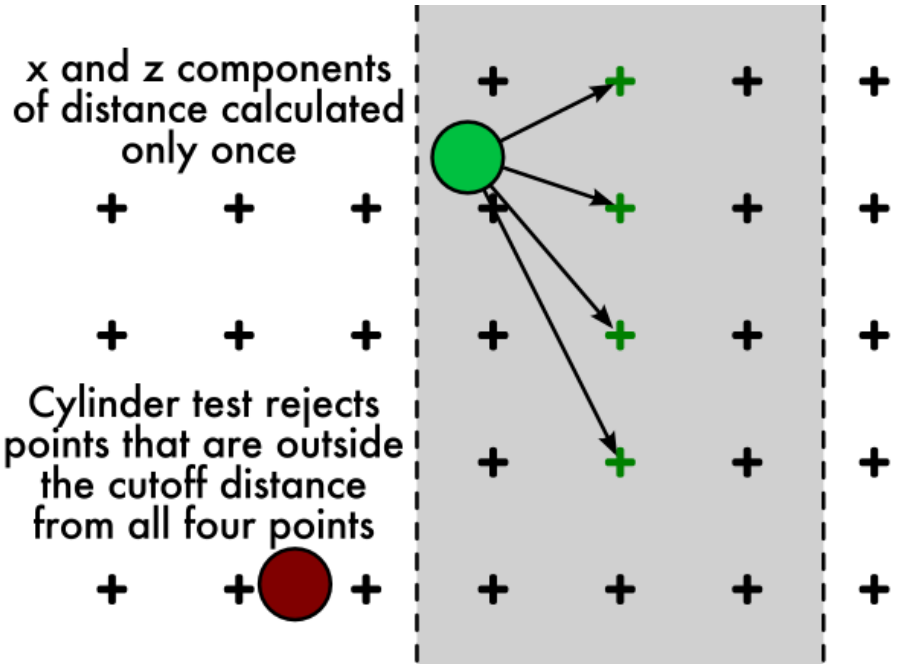
CPU Grid Data Integration

- Effect of overflow atoms are added to the CPU master energygrid array
- Slice of grid point values calculated by GPU are added into the master energygrid array while removing the padded elements



GPU Thread Coarsening

- Each thread computes potentials at four potential map points
  - Reuse x and z components of distance calculation
  - Check x and z components against cutoff distance (cylinder test)
- Exit inner loop early upon encountering the first empty slot in a bin



GPU Thread Inner Loop

Exit when an empty atom bin entry is encountered

```
if (aq == 0) break;
```

Cylinder test

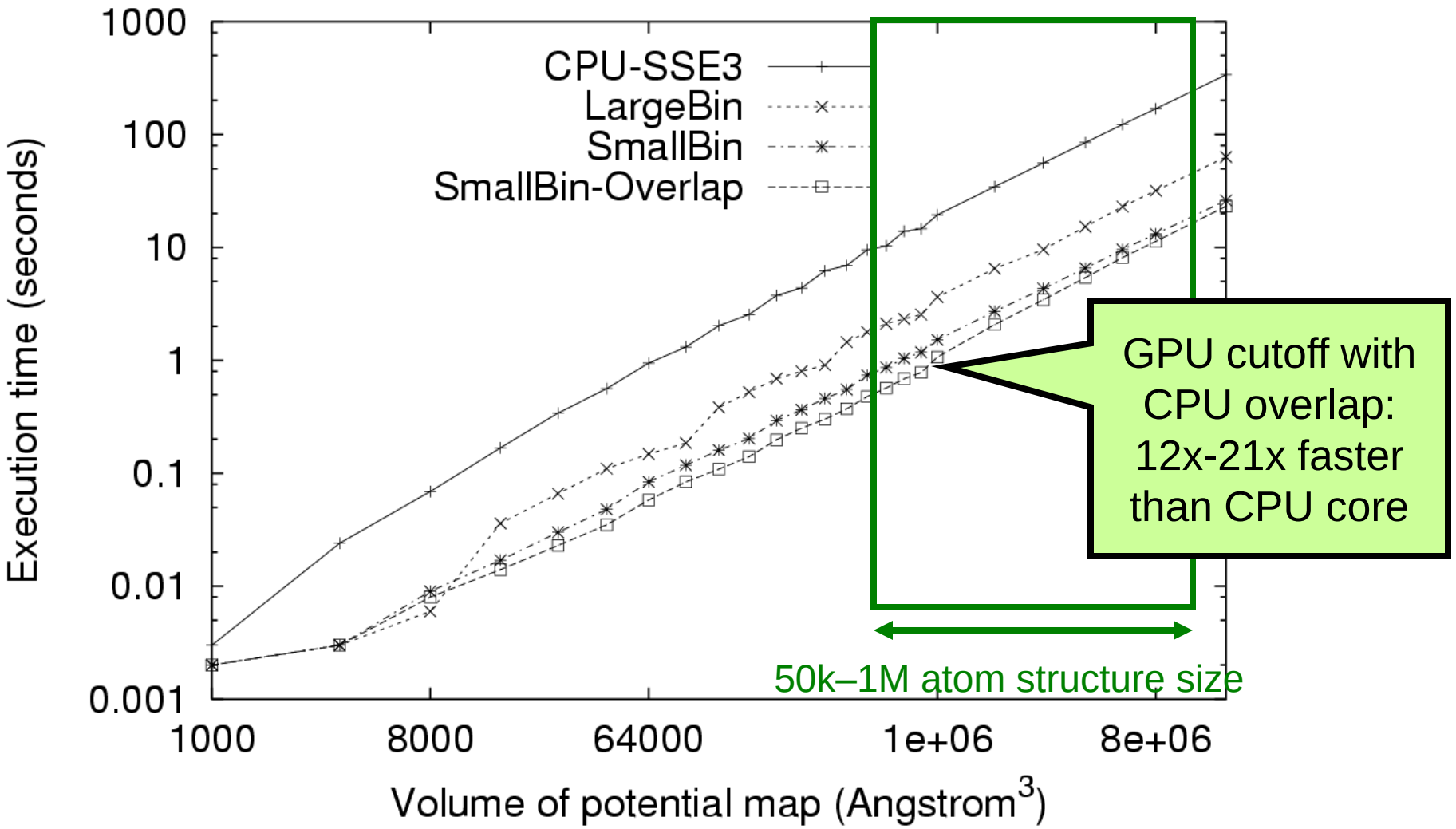
```
if (dxdz2 < cutoff2) continue;
```

Cutoff test and potential value calculation

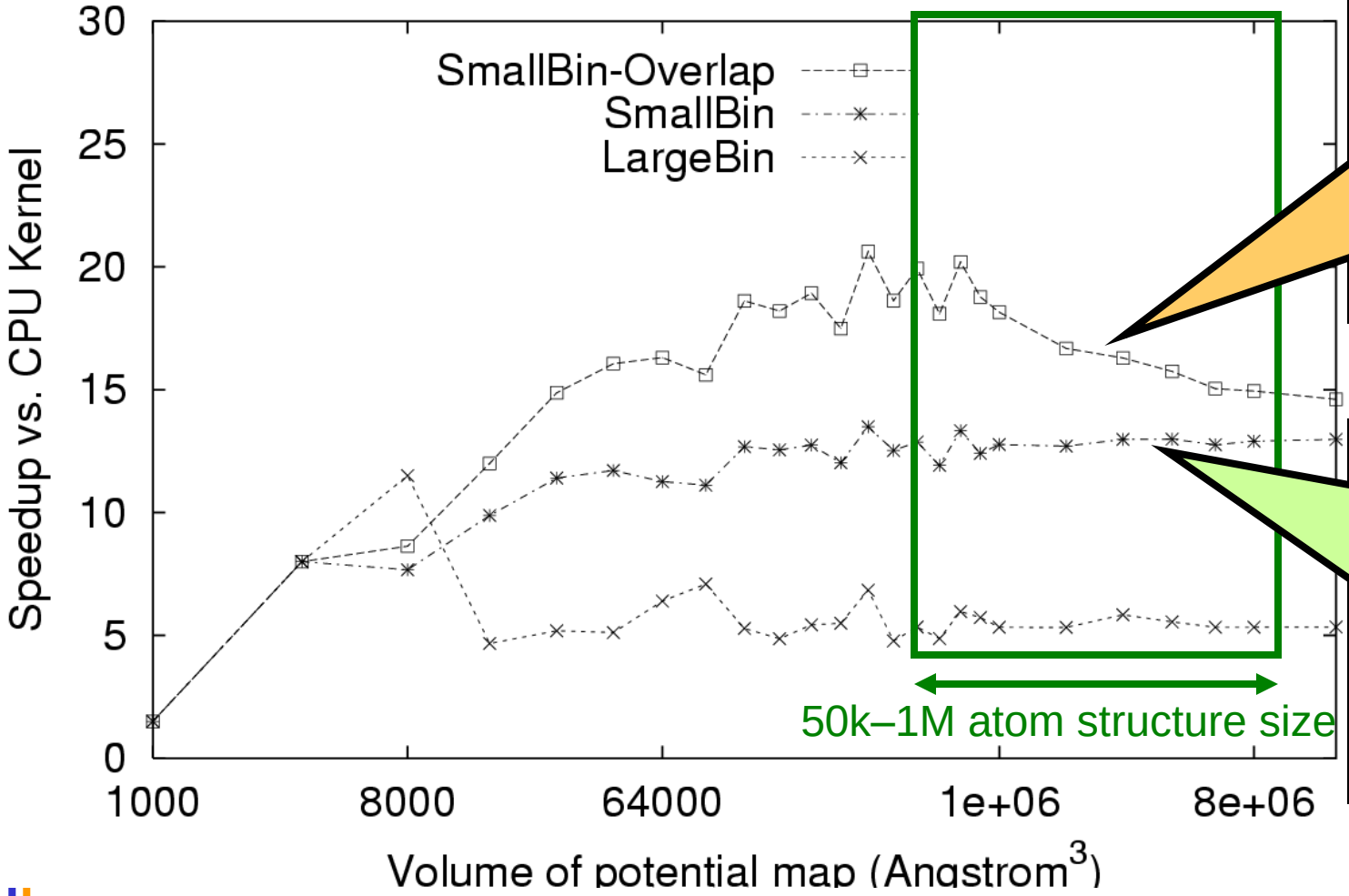
```
if (r2 < cutoff2)
  poten0 += aq * rsqrtf(r2);  // Simplified example

/* Repeat three more times */
```

Cutoff Summation Runtime



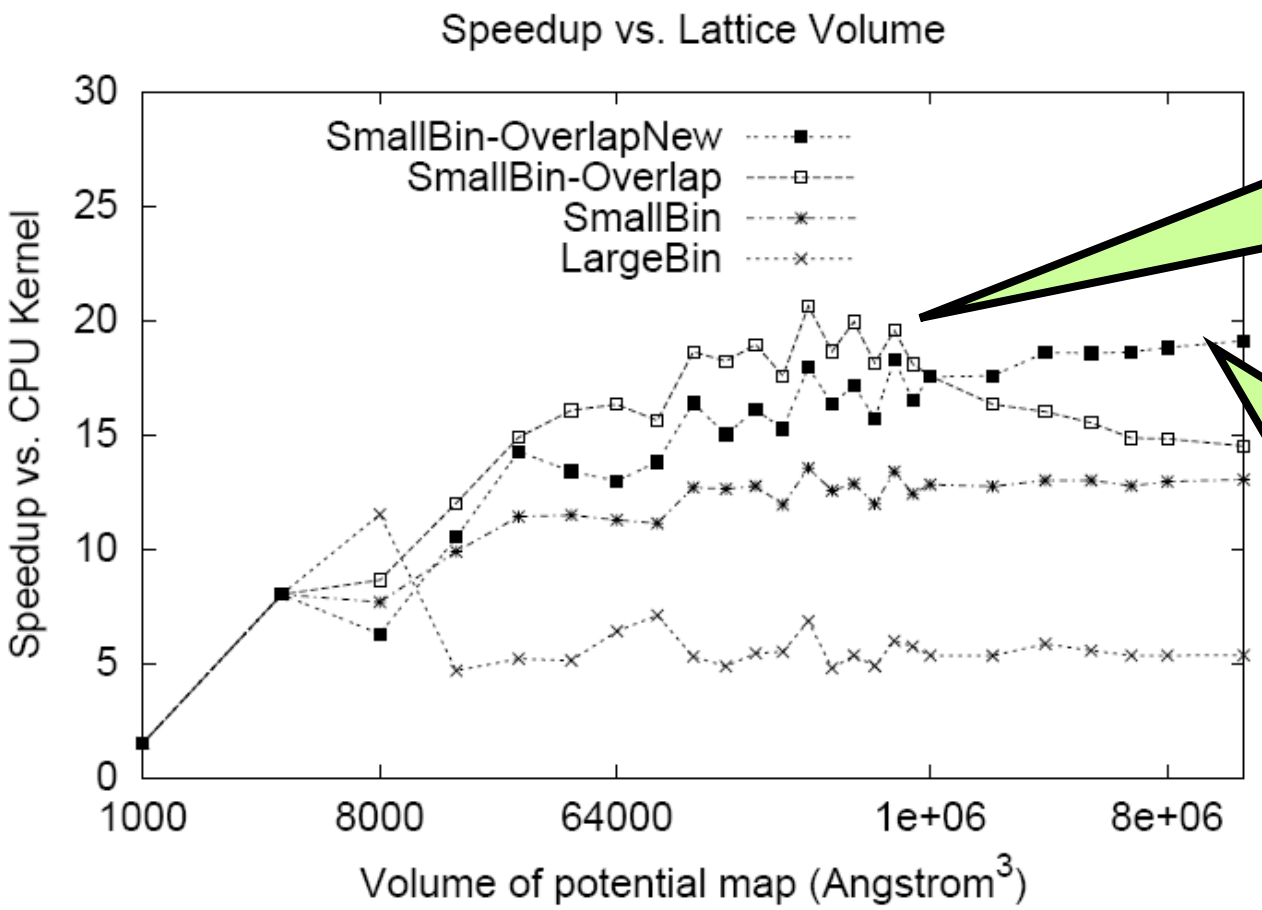
Cutoff Summation Speedup



Diminished overlap benefit due to limited queue size (16 entries)

Cutoff summation alone 9-13× faster than CPU

Cutoff Summation Runtime



GPU cutoff with CPU overlap: 17x-21x faster than CPU core

Avoid overfilling asynchronous stream queues to maintain performance for larger problems

GPU acceleration of cutoff pair potentials for molecular modeling applications. C. Rodrigues, D. Hardy, J. Stone, K. Schulten, W. Hwu. *Proceedings of the 2008 Conference On Computing Frontiers*, pp. 273-282, 2008.

Summary

- Cutoff pair potentials heavily used in molecular modeling applications
- Use CPU to **regularize** the work given to the GPU to optimize its performance
  - GPU performs very well on 64-byte-aligned array data
- Run CPU and GPU concurrently to improve performance
- Use shared memory as a program-managed cache