ICCS School

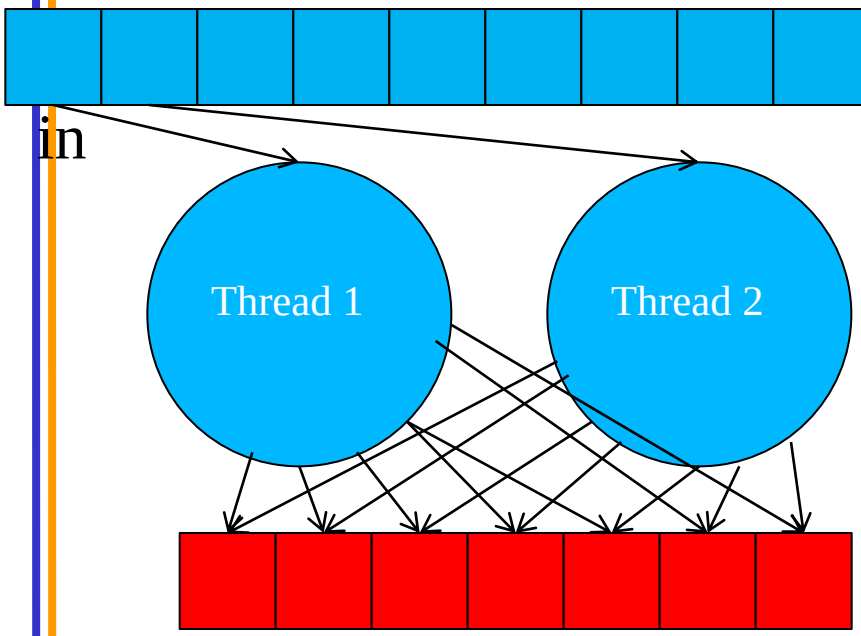
Advanced GPU Programming for Science

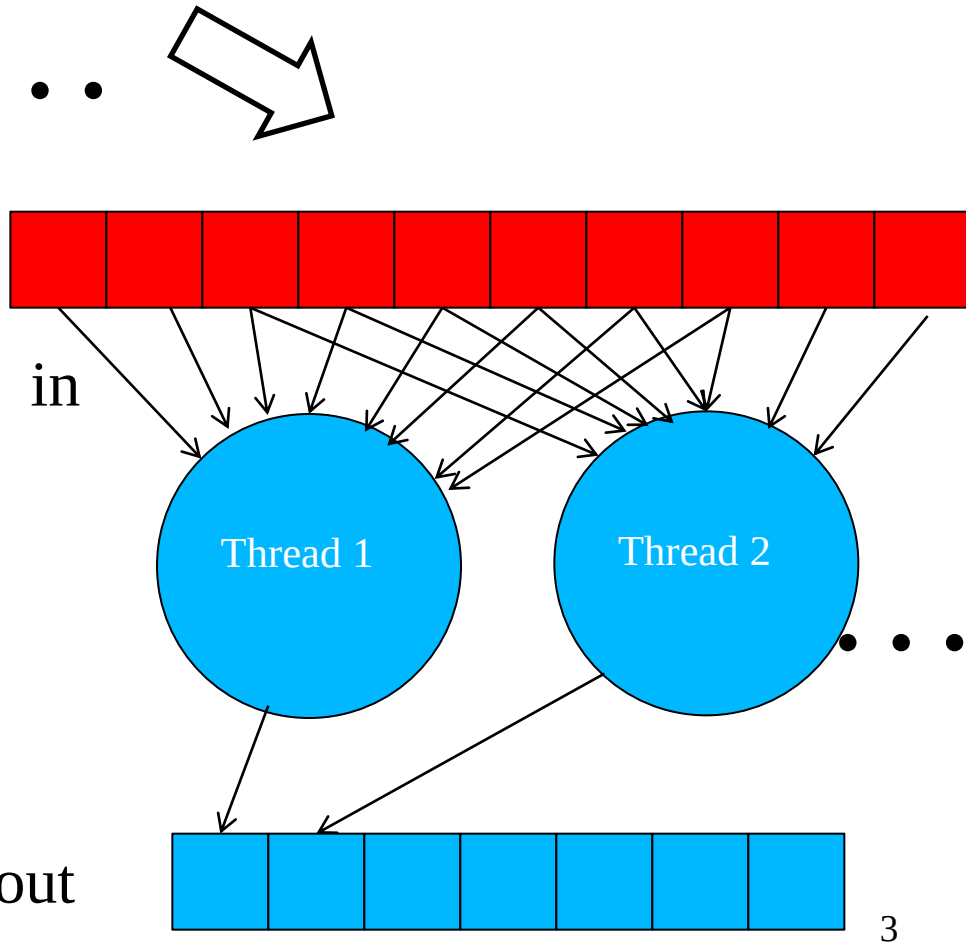
Lecture 5: Input Binning for Regular Data Sets

Objective

- To understand how data scalability problems in gather parallel execution motivate input binning
- To learn basic input binning techniques
- To understand common tradeoffs in input binning

Scatter to Gather Transformation

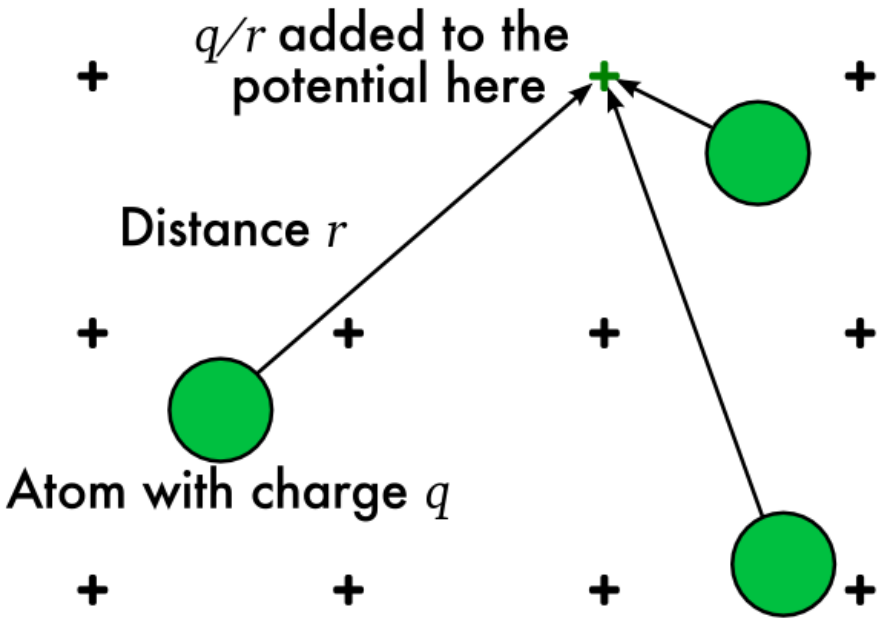




However

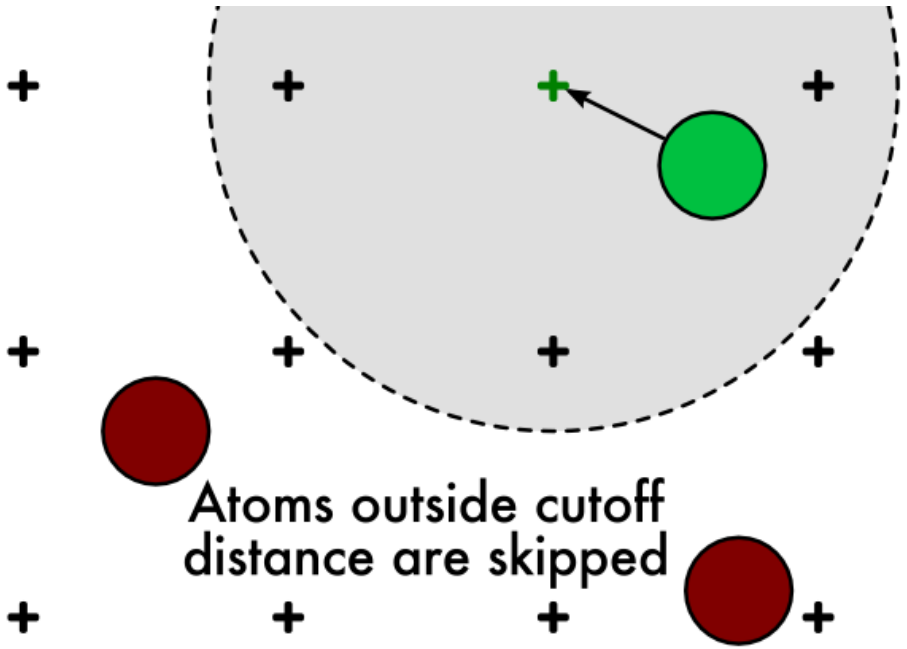
- Input tends to be much less regular than output
  - It may be difficult for each thread to efficiently locate all inputs relevant to its output
  - Or, to efficiently exclude all inputs irrelevant to its output
- In a naïve arrangement, all threads may have to process all inputs to decide if each input is relevant to its output
  - This makes execution time scale poorly with data set size – data scalability problem
  - Especially a problem for many-cores designed to process large data sets

DCS Algorithm for Electrostatic Potentials Revisited



- At each grid point, sum the electrostatic potential from all atoms
  - All threads read all inputs
- Highly data-parallel
- But has quadratic complexity
  - Number of grid points × number of atoms
  - Both proportional to volume
  - Poor data scalability in terms of 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

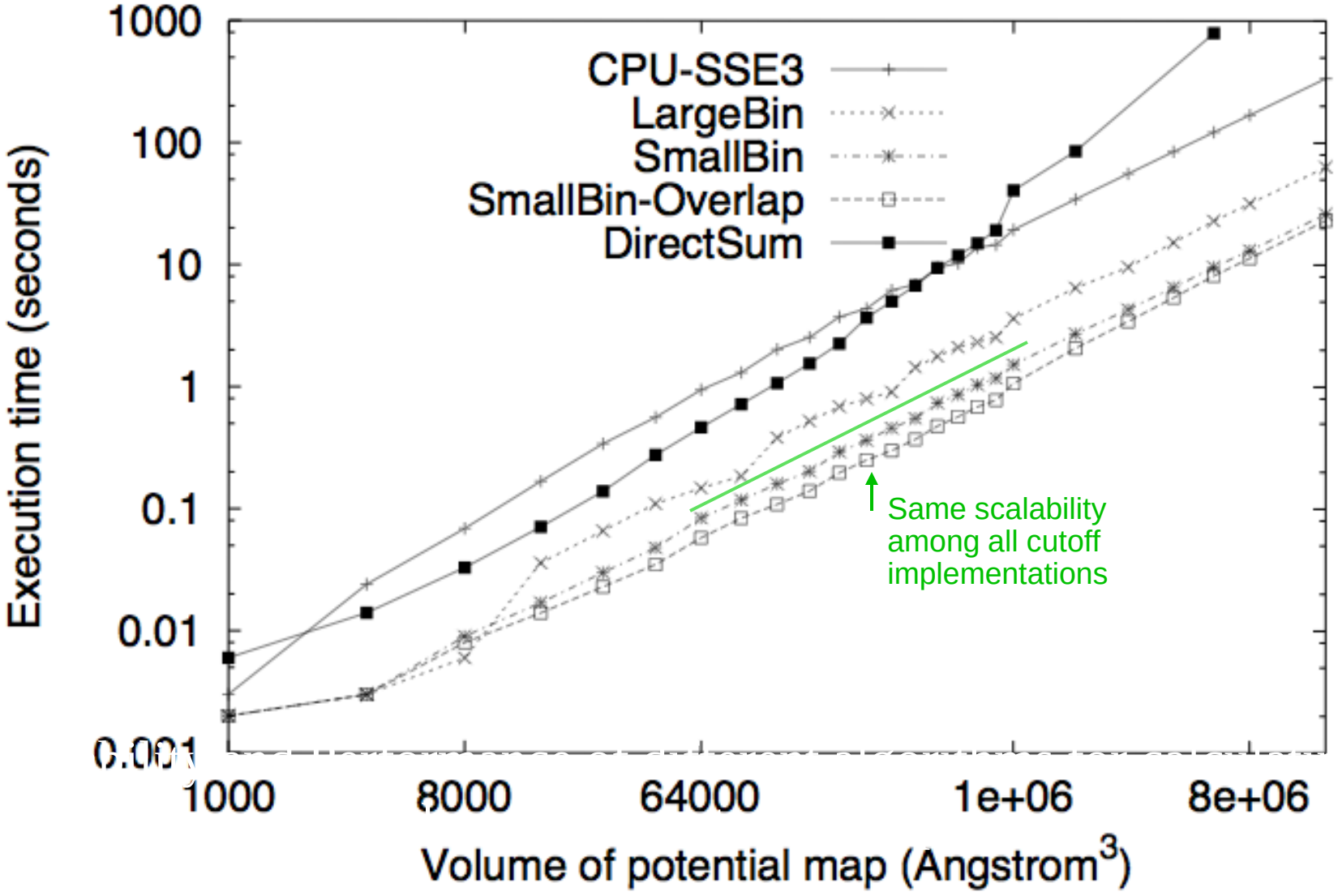
Cut-off Summation

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

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

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

Direct Summation is accurate but has poor data scalability

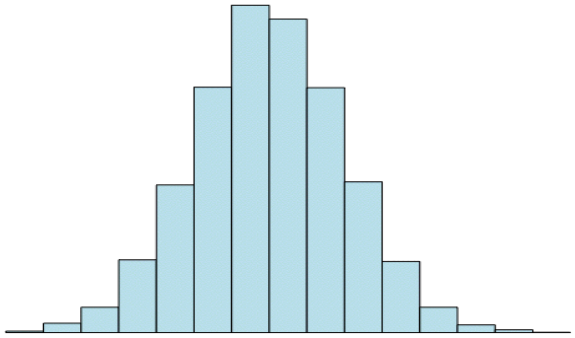


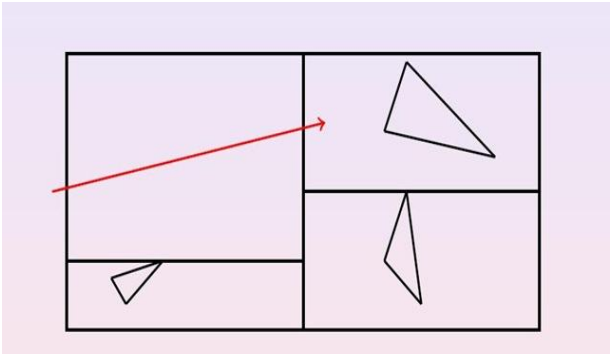
Implementation Challenge

- 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, and brings the time complexity right back to
    - number of atoms * number of grid points
  - Each thread needs to avoid examining the atoms outside the range of its grid point(s)

Binning

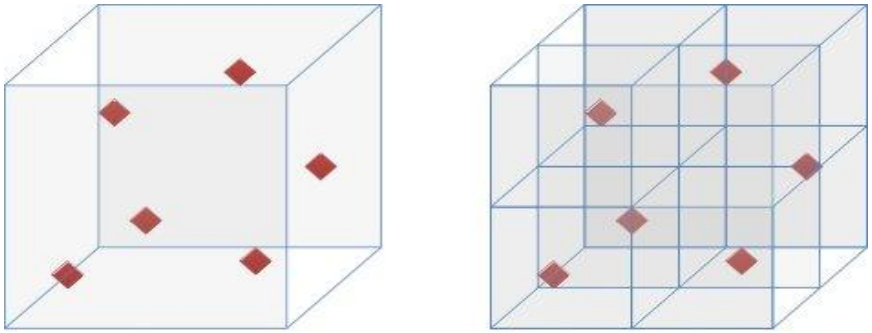
- A process that groups data to form a chunk called *bin*
- Each bin collectively represents a property for data in the bin
- Helps problem solving due to data coarsening
- Uniform bin arrays, Variable bins, KD Trees, …





Binning for Cut-Off Potential

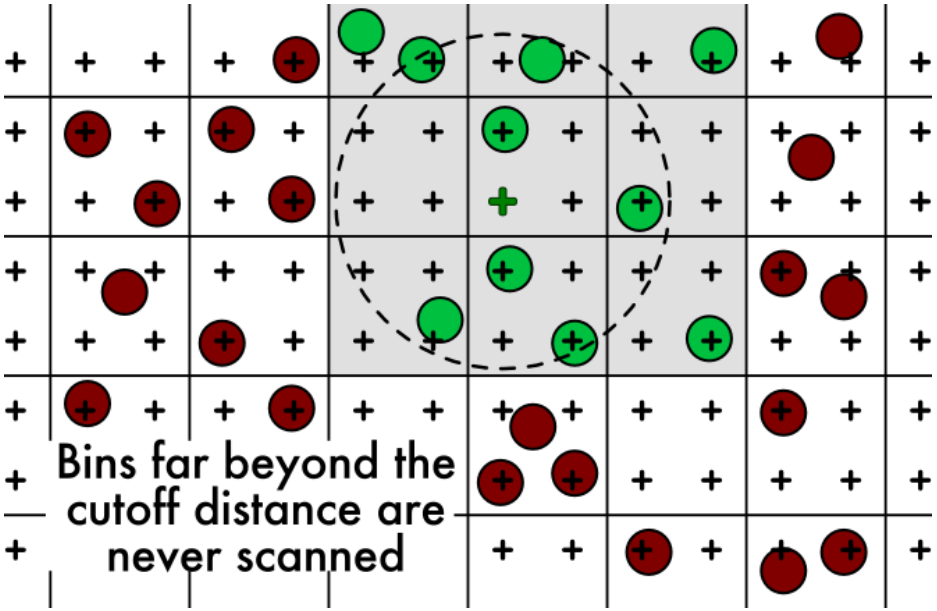
- Divide the simulation volume with non-overlapping uniform cubes
- Every atom in the simulation volume falls into a cube based on its spatial location
  - Bins represent location property of atoms
- After binning, each cube has a unique index in the simulation space for easy parallel access



(a) Simulation volume

(b) Simulation volume with eight bins

Spatial Sorting Using Binning



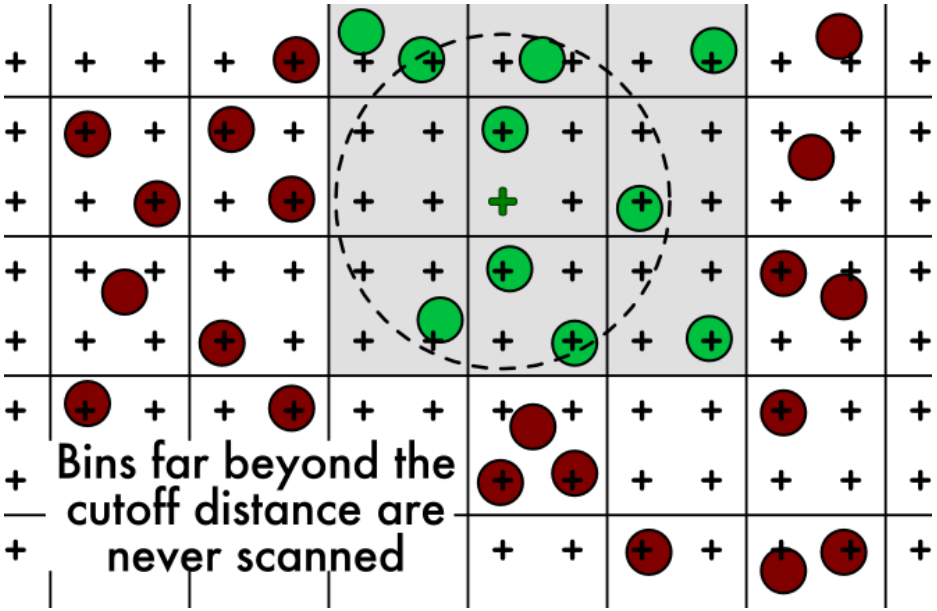
- Presort atoms into *bins* by location in space
- Each bin holds several atoms
- Cutoff potential only uses bins within r_c
  - Yields a linear complexity cutoff potential algorithm
  - Some atoms will be examined by a thread but not used

Terminology

- Bin Size
  - The size of the bin cubes that partition the simulation volume
  - The bigger the bin size, the more the atoms that will fall into each bin
- Bin Capacity
  - The number of atoms that can be accommodated by each bin in the array implementation

Bin Design

- Uniform sized/capacity bins allow array implementation
  - And the relative offset list approach
- Bin capacity should be big enough to contain most of the atoms that fall into a bin (> 95%)
  - Cut-off will screen away atoms that weren't processed
  - Performance penalty if too many are screened away



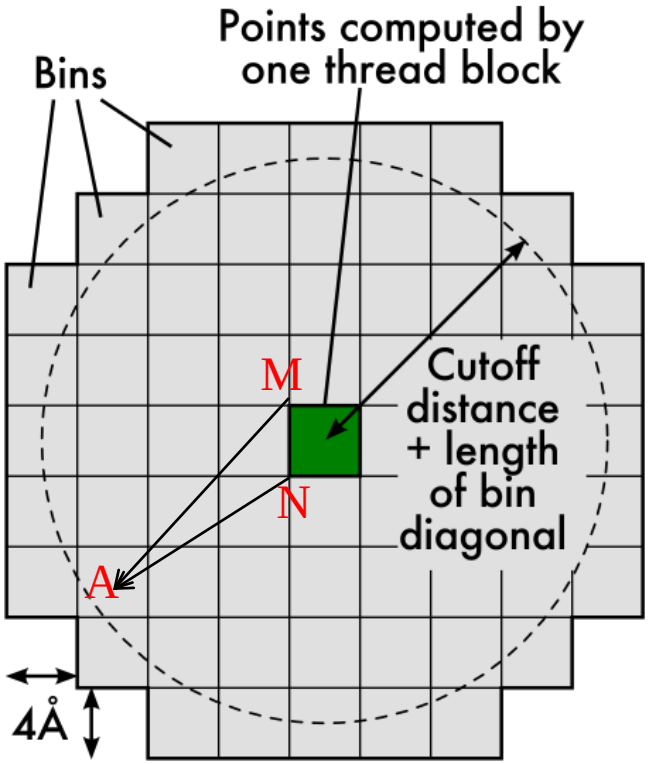
Bin Capacity Considerations

- 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

- Place 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

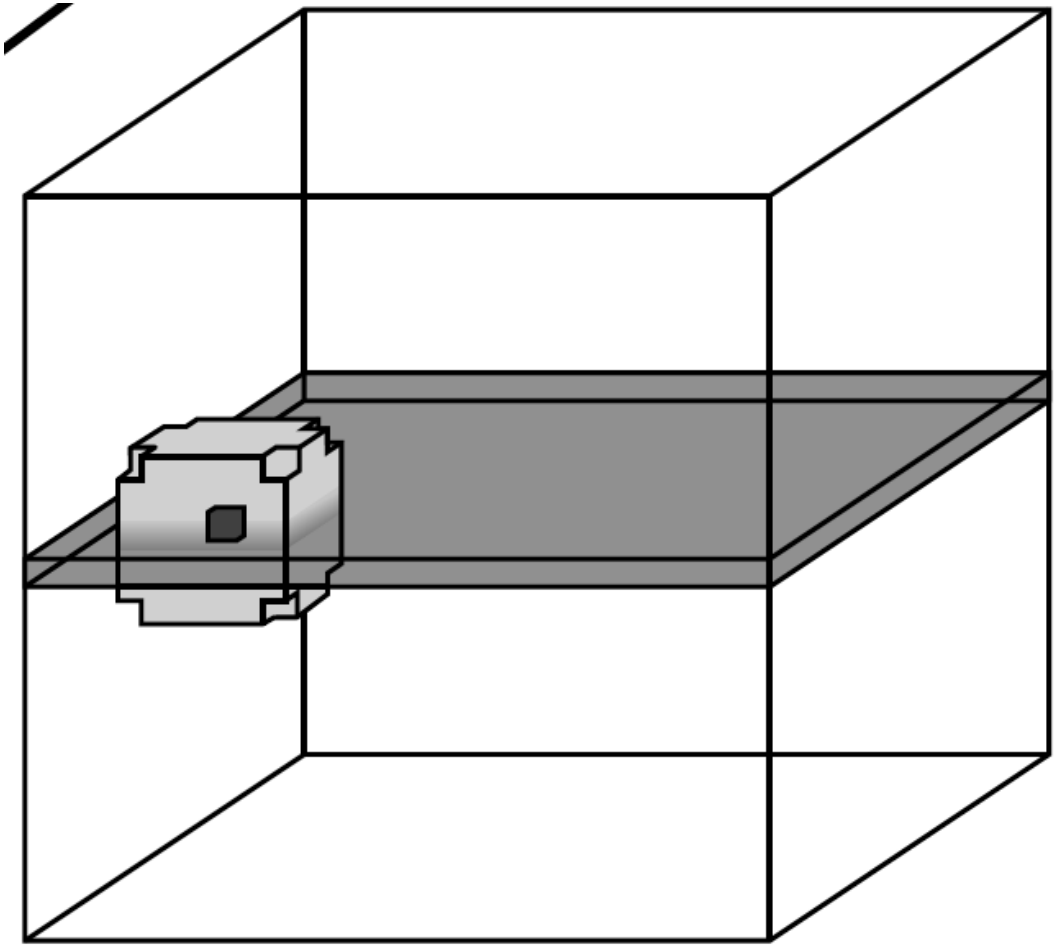
Small-bin Kernels – Work Efficiency

- Thread block examines atom bins up to the cutoff distance
  - Use a sphere of bins
  - All threads in a block scan the same bins and atoms
    - No hardware penalty for multiple simultaneous reads of the same address
    - Simplifies fetching of data
  - The sphere has to be big enough to cover all grid points at corners
  - There will be a small level of divergence
    - Not all grid points processed by a thread block relate to all atoms in a bin the same way
    - (A within cut-off distance of N but outside cut-off of M)



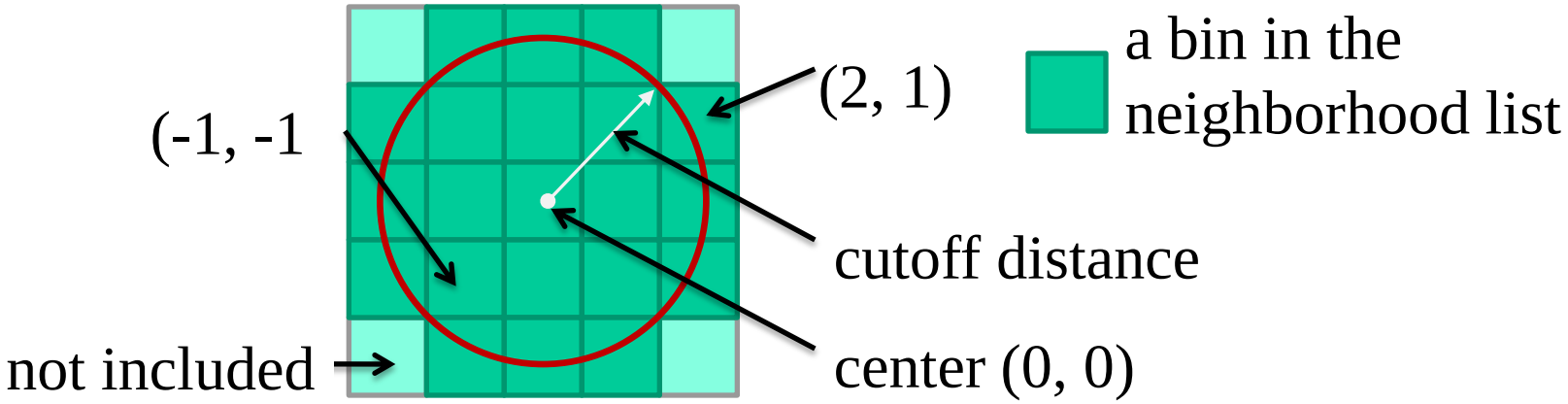
The Neighborhood is a volume

- Calculating and specifying all bin indexes of the sphere can be quite complex
  - Rough approximations reduce efficiency



Neighborhood Offset List (Pre-calculated)

- A list of relative 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



Pseudo Code of an Implementation

```
// 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)
```

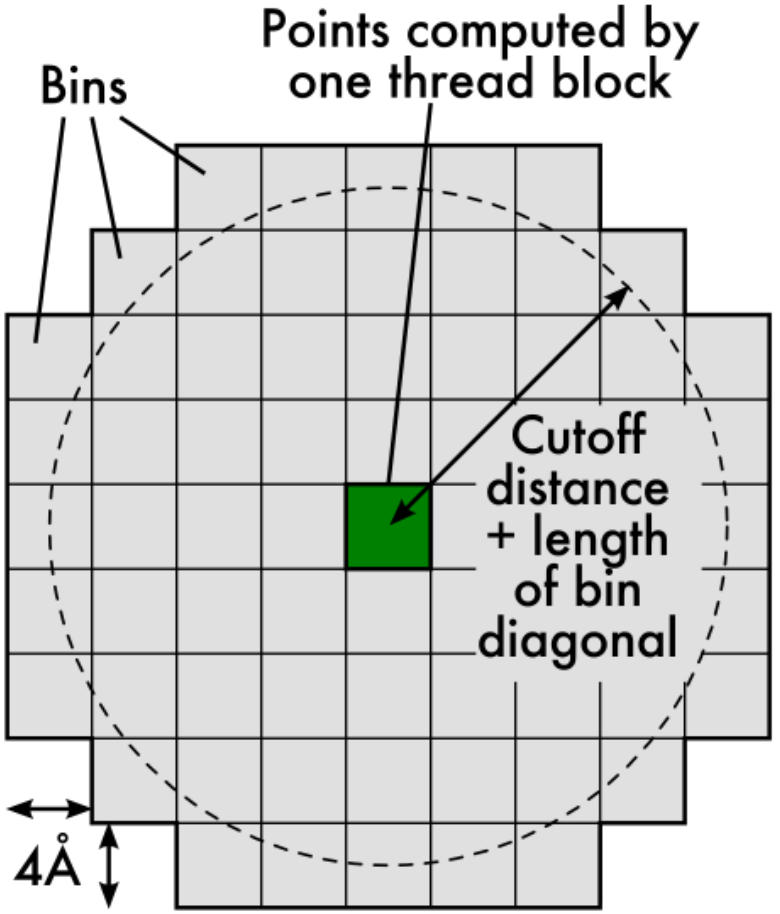
GPU

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

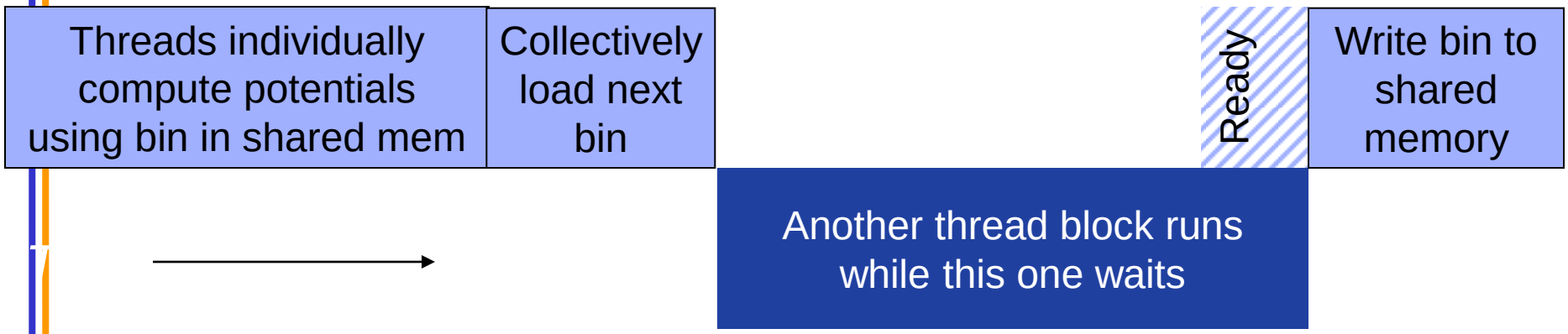
Details on Small Bin Design

- 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)
  - Using the same cube bin size 34% of examined atoms are within cutoff distance



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



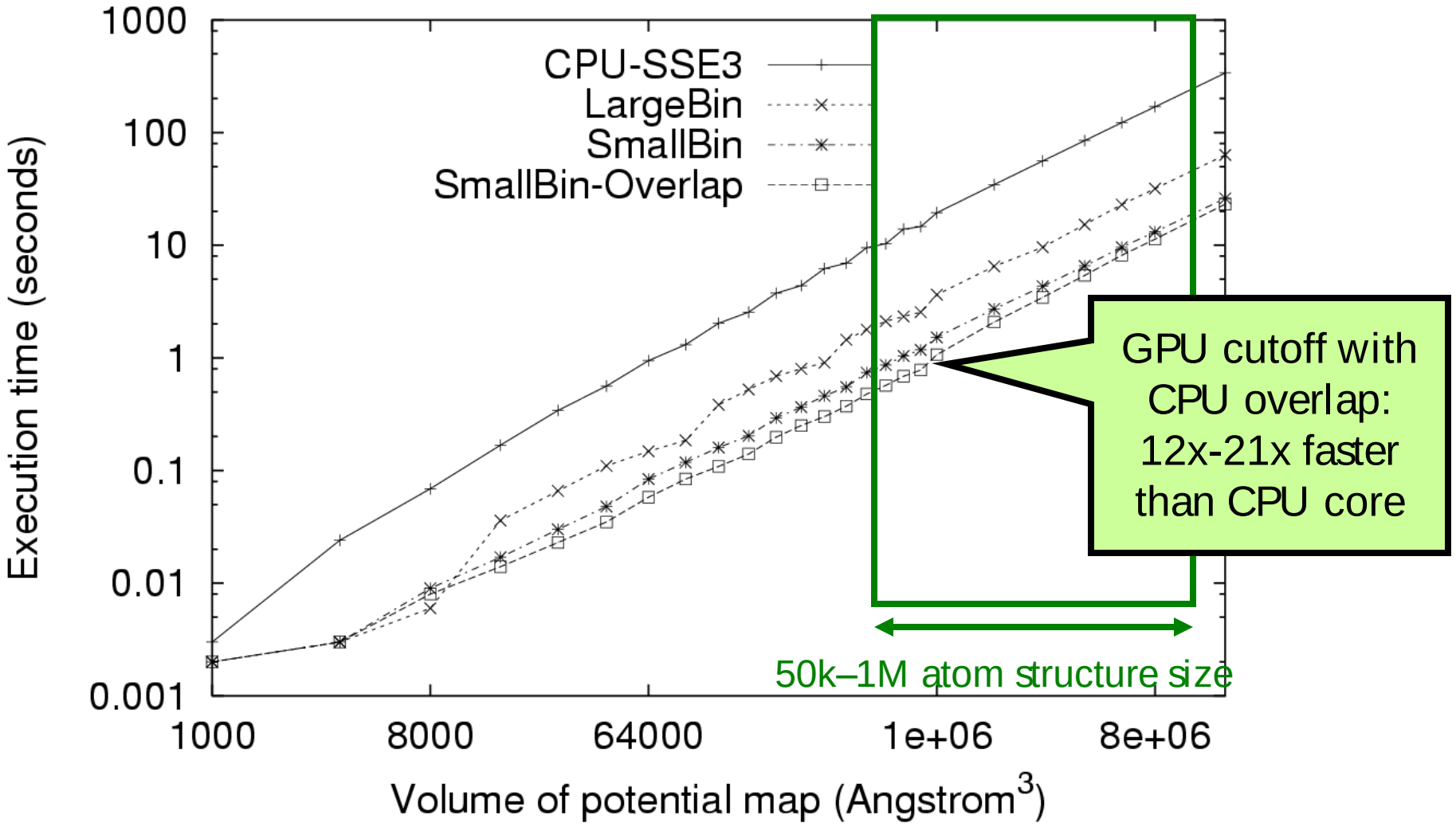
Coalesced Global Memory Access to Atom Data

- 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

Cutoff Summation Runtime



QUESTIONS?