We’ll start at Berkeley time

12:10PM

CS 267 Lab 2.3

CUDA/GPU Particle Simulation

Guanhua Wang

Mar 9, 2022

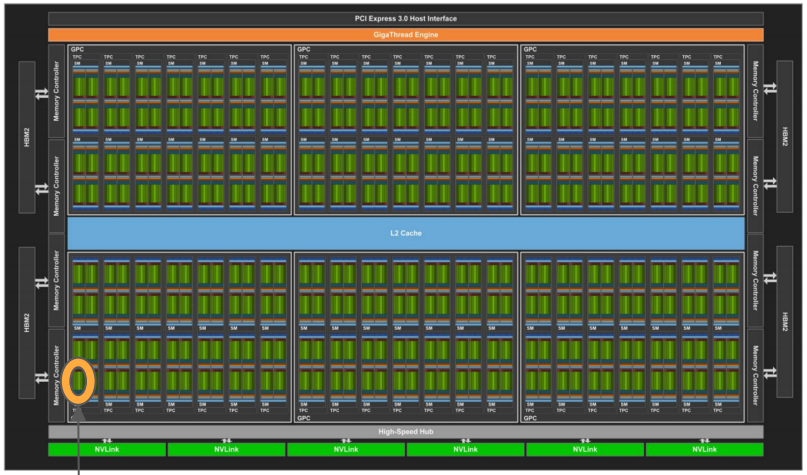
Berkeley
UNIVERSITY OF CALIFORNIA

Refresher on GPUs from Lecture 7

What is a GPU?

- Separate device (accelerator)
- Multicore on steroids (“manycore”)
  - KNL processors had 68 cores / 272 threads
  - NVIDIA V100s have **5120 cores**

- Drawback: each core is weak
- CUDA is the language for GPUs
- More details in Lecture 7



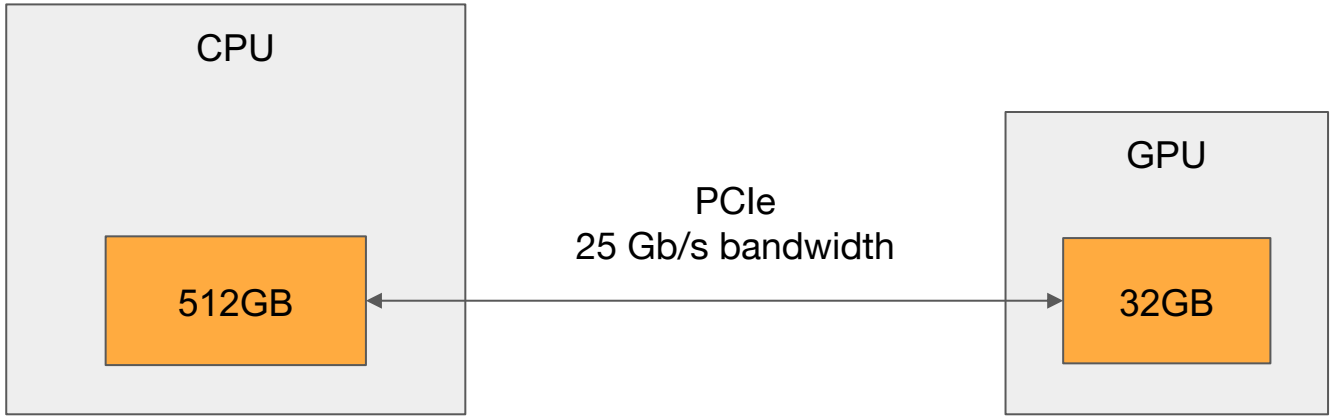
1 SM (stream multi-processor) = 64 cores

NVIDIA V100

Can we just implement our OpenMP solution in CUDA?

Depends on your OpenMP solution

- Storing particle bins as std::vector's won't work well
  - Would need to copy vectors to GPU memory per step
    - Can't add/remove to vector on GPU
  - thrust::device_vector's seem appealing, but just copy data under the hood



Need a way to store particles per bin without std::vectors

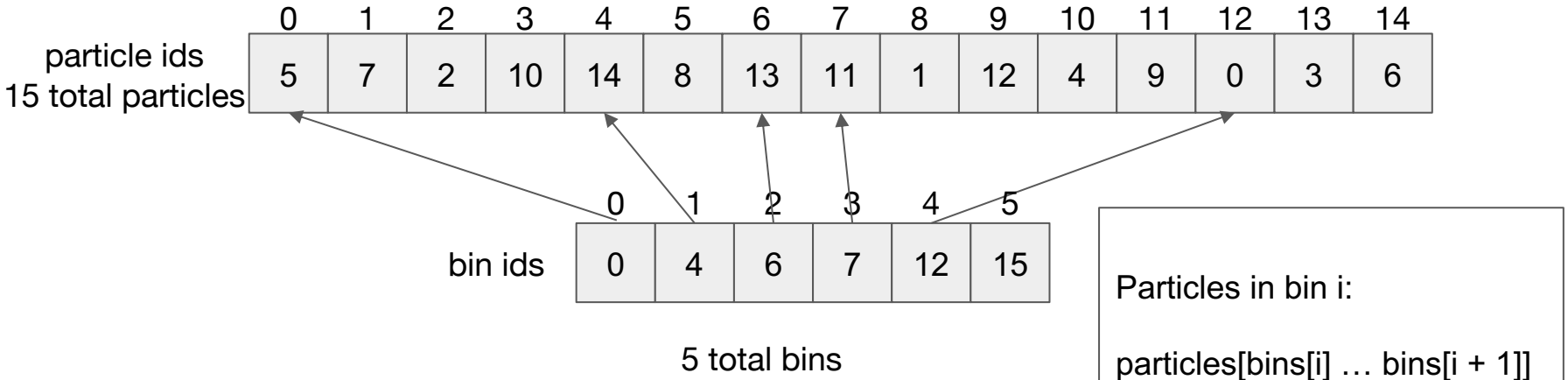
We can do this with arrays instead

Particle bins with arrays

- Naive way: One array of length #particles per bin
  - #particles * #bins is too much memory (GPUs don’t have a lot of memory)
    - e.g., 16/32GB device memory, which is orders-of-magnitude less than CPU memory

Particle bins with arrays

- Naive way: One array of length #particles per bin
  - #particles * #bins is too much memory (GPUs don't have a lot of memory)
- Better way: Array of particles sorted by bin id

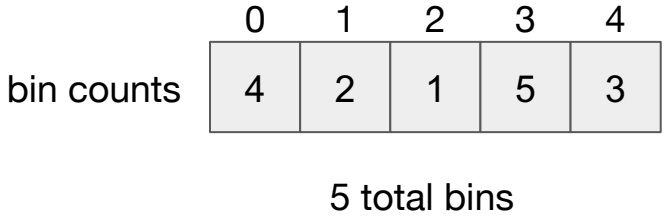


Particles in bin i:

particles[bins[i] … bins[i + 1]]

How to compute this structure?

1. Count number of particles per bin
    a. Can parallelize across particles on GPU
    b. Likely need atomic instructions (e.g. atomicAdd/atomicInc == fetch-and-add in c) instead of locks



How to compute this structure?

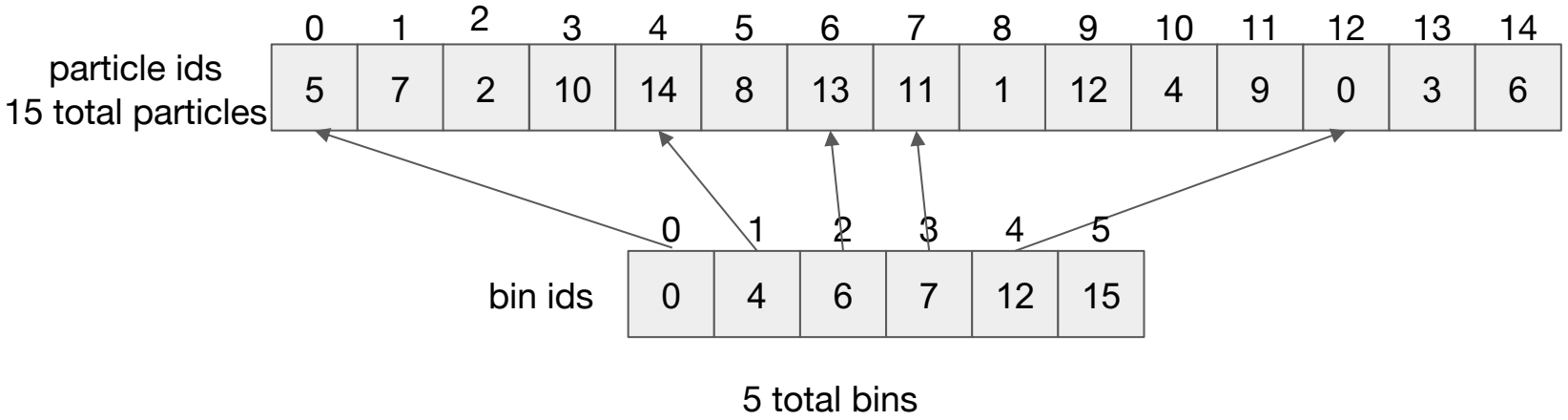
2. Prefix sum the bin counts
    a. Can run prefix sums efficiently in parallel -- see Lecture 8
    b. Either implement yourself or use thrust library functions

	0	1	2	3	4	5
bin counts	0	4	6	7	12	15

5 total bins

How to compute this structure?

3. Add particles to separate array starting from bin idx
    a. Can parallelize across particles on GPU
    b. Likely need atomic instructions (e.g. atomicAdd/atomicInc)



How to compute this structure

1. Count number of particles per bin
2. Prefix sum the bin counts
3. Add particles to separate array starting from bin idx

- Every step on GPU
- Rebin every particles at each step (not only particles that changes bin)
- No need for memory copies

General Tips for GPU programming

- Don’t need to implement everything in GPU kernels all at once
- Useful approach
  - Start off implementing everything on CPU
  - Move some part to a GPU kernel, and copy memory to and from GPU memory as necessary
  - Eventually, everything is in a GPU kernel w/o any CPU to GPU memcpy
  - For atomic operation, if atomicAdd/Sub/Inc/Dec/Min/Max are not sufficient, you can implement any atomic operations using atomicCAS (Compare And Swap).

- cuda-gdb is a useful tool.

- nvprof is for performance debugging (memCpy time, kernel compute time).

Questions?