Description of Seed finding Kernels

Beomki Yeo

Cuda Seed Finding Flow

```
traccc::cuda::doublet_counting(m_seedfinder_config, isp_container,
                               doublet_counter_container, m_mr);

traccc::cuda::doublet_finding(
    m_seedfinder_config, isp_container, doublet_counter_container,
    mid_bot_container, mid_top_container, m_mr);

traccc::cuda::triplet_counting(m_seedfinder_config,
                               isp_container, doublet_counter_container,
                               mid_bot_container, mid_top_container,
                               triplet_counter_container, m_mr);

traccc::cuda::triplet_finding(
    m_seedfinder_config, m_seedfilter_config, isp_container,
    doublet_counter_container, mid_bot_container, mid_top_container,
    triplet_counter_container, triplet_container, m_mr);

traccc::cuda::weight_updating(m_seedfilter_config, isp_container,
                              triplet_counter_container,
                              triplet_container, m_mr);

traccc::cuda::seed_selecting(
    m_seedfilter_config, isp_container, doublet_counter_container,
    triplet_counter_container, triplet_container, seed_container, m_mr);

return seed_container;
```

1. Doublet counting
   - Counts the number of doublets for every middle spacepoint
2. Doublet finding
   - Does the same thing with Doublet counting but also adds the found doublets into the container in a sorted order
3. Triplet counting
   - Counts the number of triplets for every middle-bottom doublet
4. Triplet finding
   - Does the same thing with Triplet counting but adds the found triplets into the container in a sorted order
5. Weight updating
   - For every triplet, iterates over other triplets with the same middle-bottom doublets to update its weight based on the number of compatible triplets (curvature and distance)
6. Seed selecting
   - Seed selecting based on experiment-dependent cuts

Doublet, Triplet and Seed Container

- The (bottom and top) spacepoints in neighbor bins are accessed with the index of neighbor bins and spacepoint in the internal spacepoint container

- Multiplet (doublet and triplet) struct is defined with the index member variables, while seed struct takes spacepoint as member variable

Doublet definition

```
// Middle - Bottom or Middle - Top
struct doublet {
    sp_location sp1;
    sp_location sp2;
};
```

Spacepoint location

```
struct sp_location {
    /// index of the bin of the spacepoint grid
    unsigned int bin_idx;
    /// index of the spacepoint in the bin
    unsigned int sp_idx;
};
```

Triplet definition

```
// Bottom - Middle - Top
struct triplet {
    sp_location sp1;  // bottom
    sp_location sp2;  // middle
    sp_location sp3;  // top
    scalar curvature;
    scalar weight;
    scalar z_vertex;
};
```

Seed definition

```
struct seed {
    spacepoint spB;
    spacepoint spM;
    spacepoint spT;
    float weight;
    float z_vertex;
```

- Container
  - Header: number of doublets or triplets per bin
  - Item: doublets or triplets per bin

```
using host_doublet_container = host_container<unsigned int, doublet>;
```

Doublet and Triplet Counter Container (only for gpu)

- Counter objects are used to count the number of doublet or triplet

- Doublet counter container: counts number of doublets per middle space point

```
using host_doublet_counter_container =
    host_container<unsigned int, doublet_counter>;
```

Header: number of compatible middle sp per bin
Item: doublet counter for compatible middle sp

```
struct doublet_counter {
    sp_location spM;
    size_t n_mid_bot = 0;
    size_t n_mid_top = 0;
};
```

- Triplet counter container: counts number of triplets per middle-bottom doublet

```
using host_triplet_counter_container =
    host_container<unsigned int, triplet_counter>;
```

Header: number of compatible middle-bot doublets per bin
Item: triplet counter for compatible middle-bot doublets

```
struct triplet_counter {
    doublet mid_bot_doublet;
    size_t n_triplets = 0;
};
```

Thread block configuration policy

- One thread for one item object (spacepoint or doublet or triplet)
- ~~One block for one grid bin~~ : The number of blocks is not enough to fill GPU resources
- Multiple blocks for one grid bin
  - If the number of items is larger than the number of threads, assign more blocks for the grid bin

Kernel 1: Doublet Counting

- Thread block setup
  - Num threads: 2×32
  - Num blocks (N_b):

$$N_b = \sum_{i=1}^{130} N_i \quad \text{where } N_i = \text{num of middle spacepoint of i-th bin/num of threads + 1}$$

- Every thread (for a middle spacepoint) iterates over bottom and top spacepoints in neighbor bins to count doublets
- If the number of middle-bottom and middle-top doublets is more than zero, doublet counter object is recorded

```
if (n_mid_bot > 0 && n_mid_top > 0) {
    auto pos = atomicAdd(&num_compat_spM_per_bin, 1);
    doublet_counter_per_bin[pos].spM = spM_loc;
    doublet_counter_per_bin[pos].n_mid_bot = n_mid_bot;
    doublet_counter_per_bin[pos].n_mid_top = n_mid_top;
}
```

Kernel 2: Doublet Finding

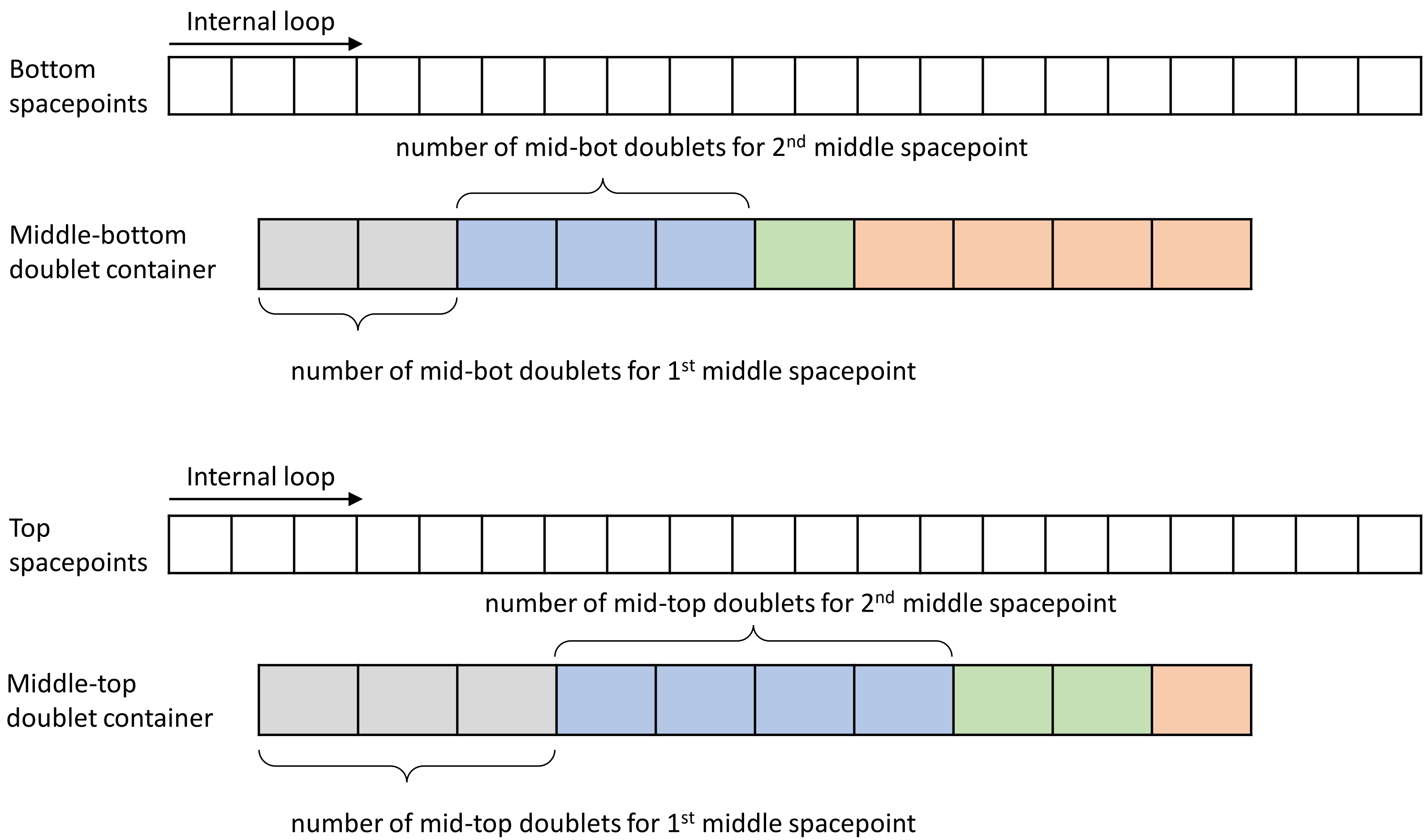
- Thread block setup
  - Num threads: 2×32
  - Num blocks (N_b):

$$N_b = \sum_{i=1}^{130} N_i$$ where $N_i =$ num of compatible middle sp of i-th bin/num of threads + 1

- Every thread (for a compatible middle sp) iterates over bottom and top spacepoint in neighbor bins to record the doublet objects

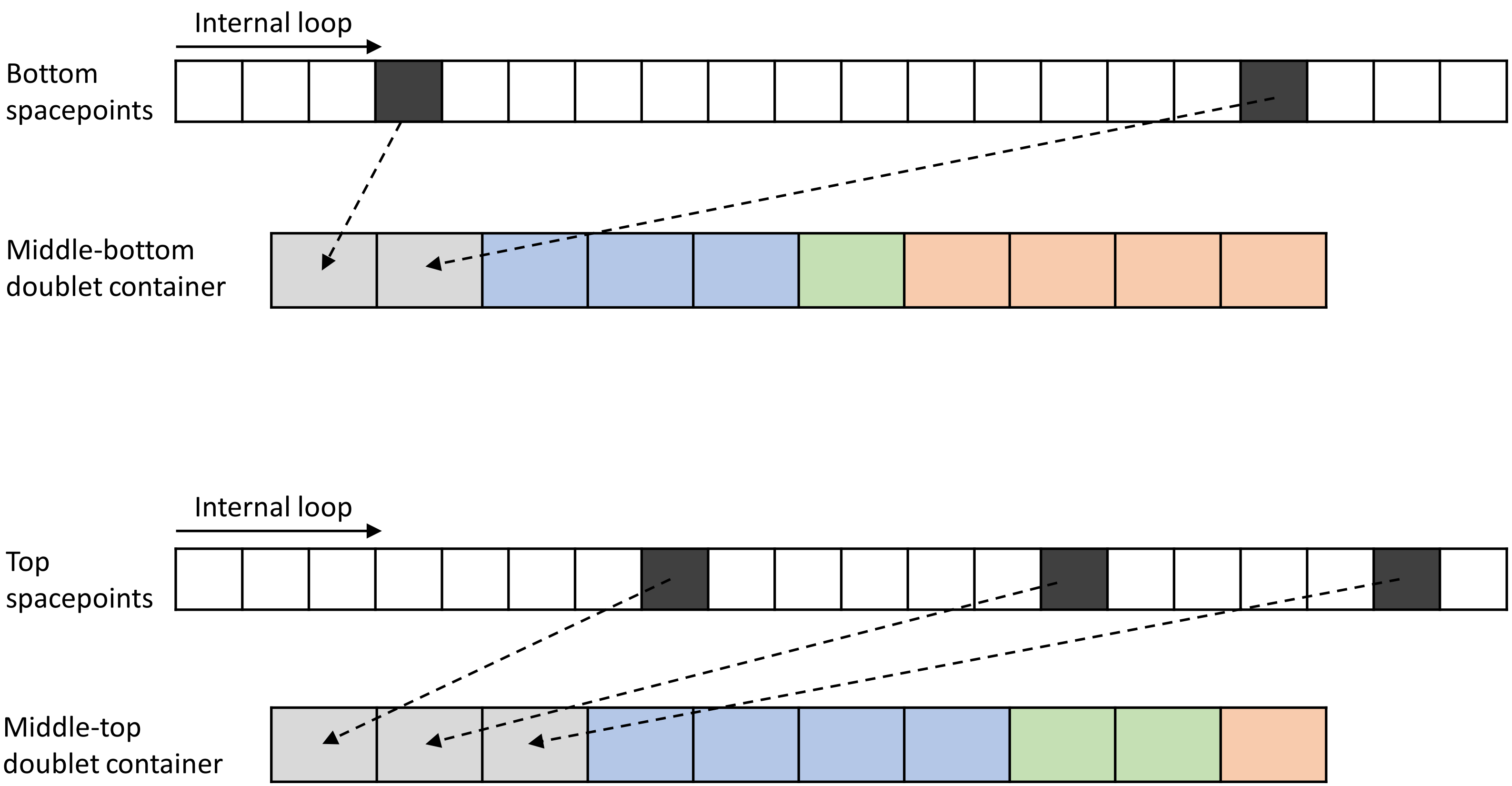
- The doublet counter is used to pre-assign the memory space for doublet objects

Kernel 2: Doublet Finding (cont.)



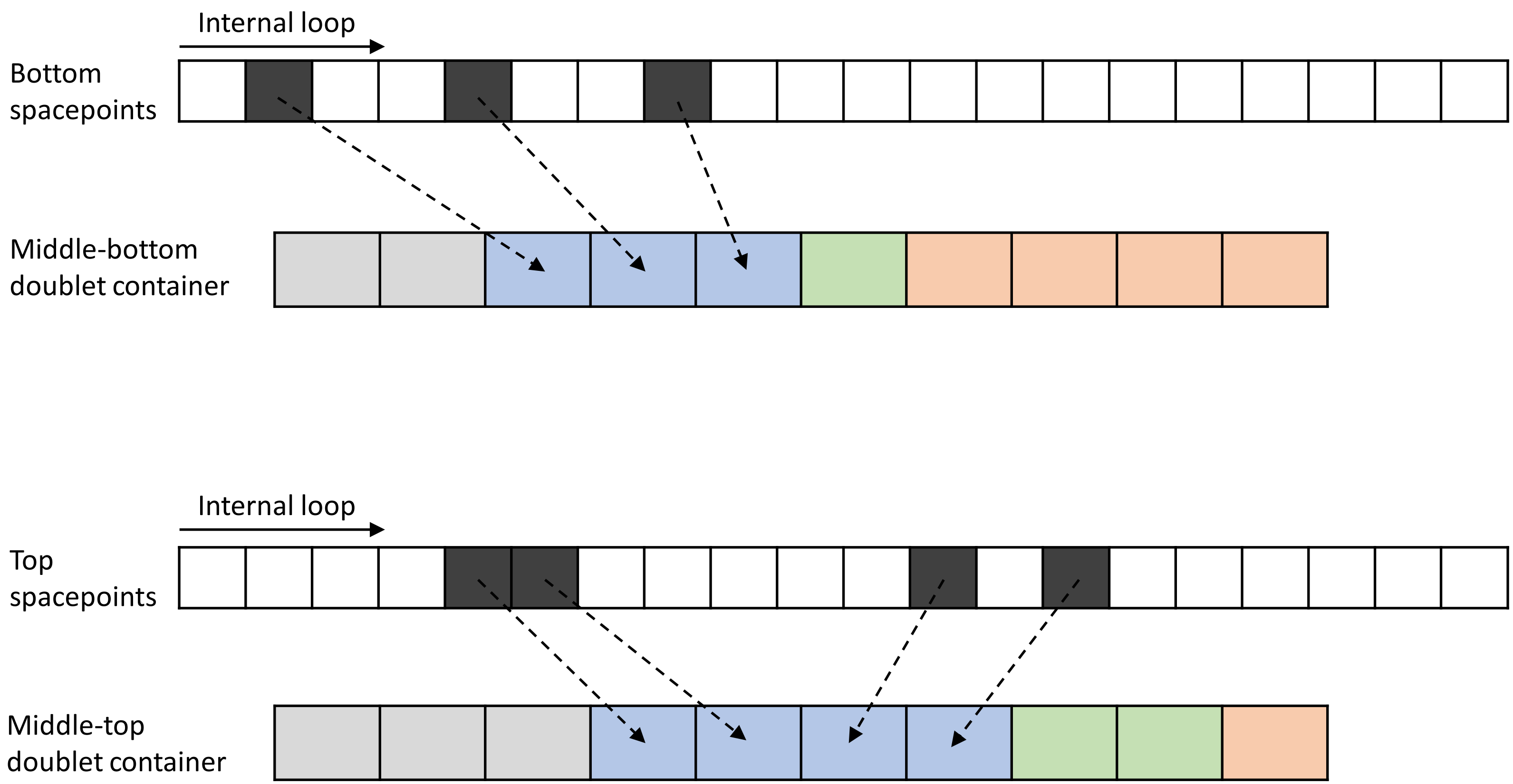
Kernel 2: Doublet Finding (cont.)

- Thread 1: 1st middle spacepoint



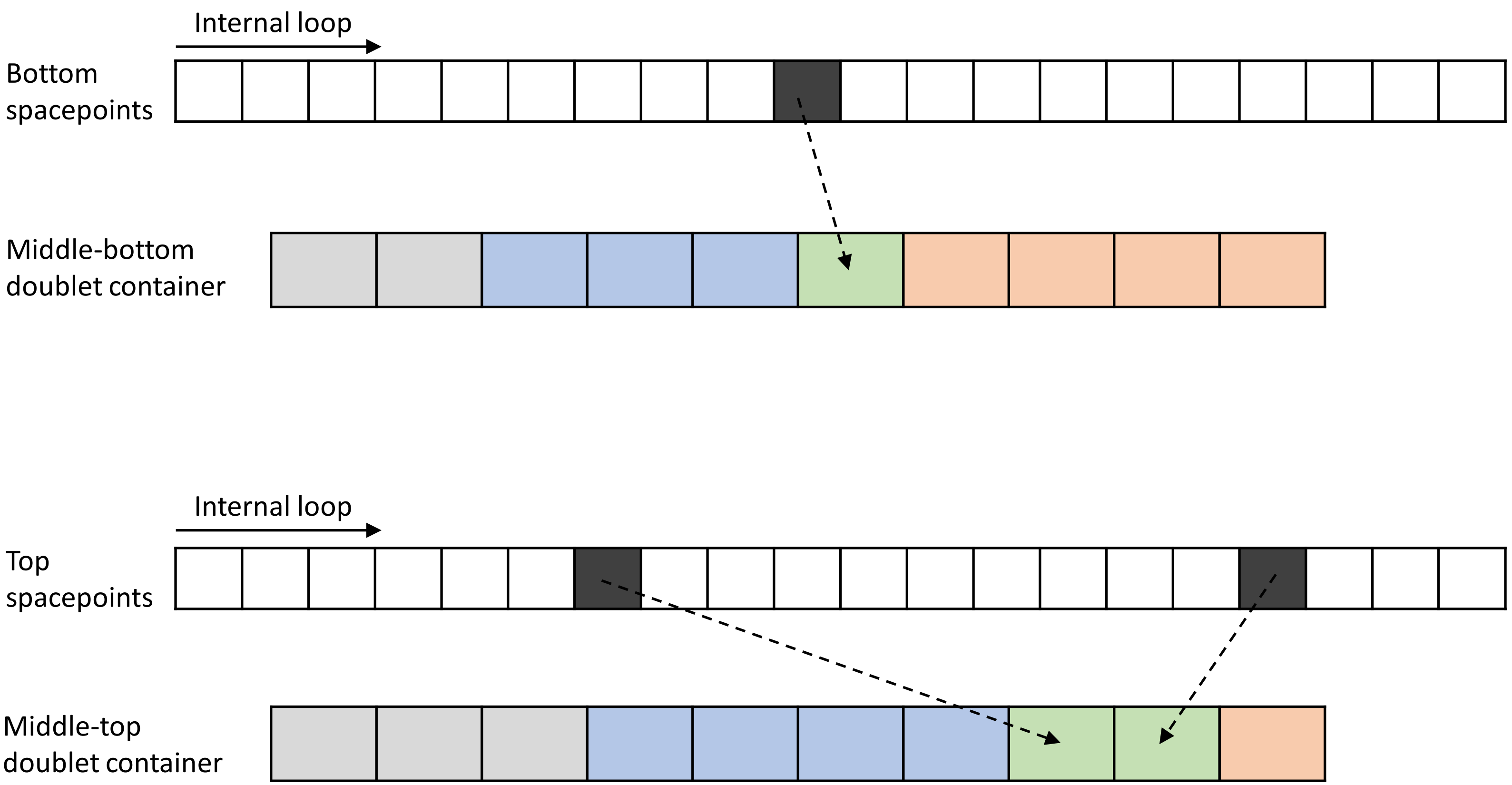
Kernel 2: Doublet Finding (cont.)

- Thread 2: 2nd middle spacepoint



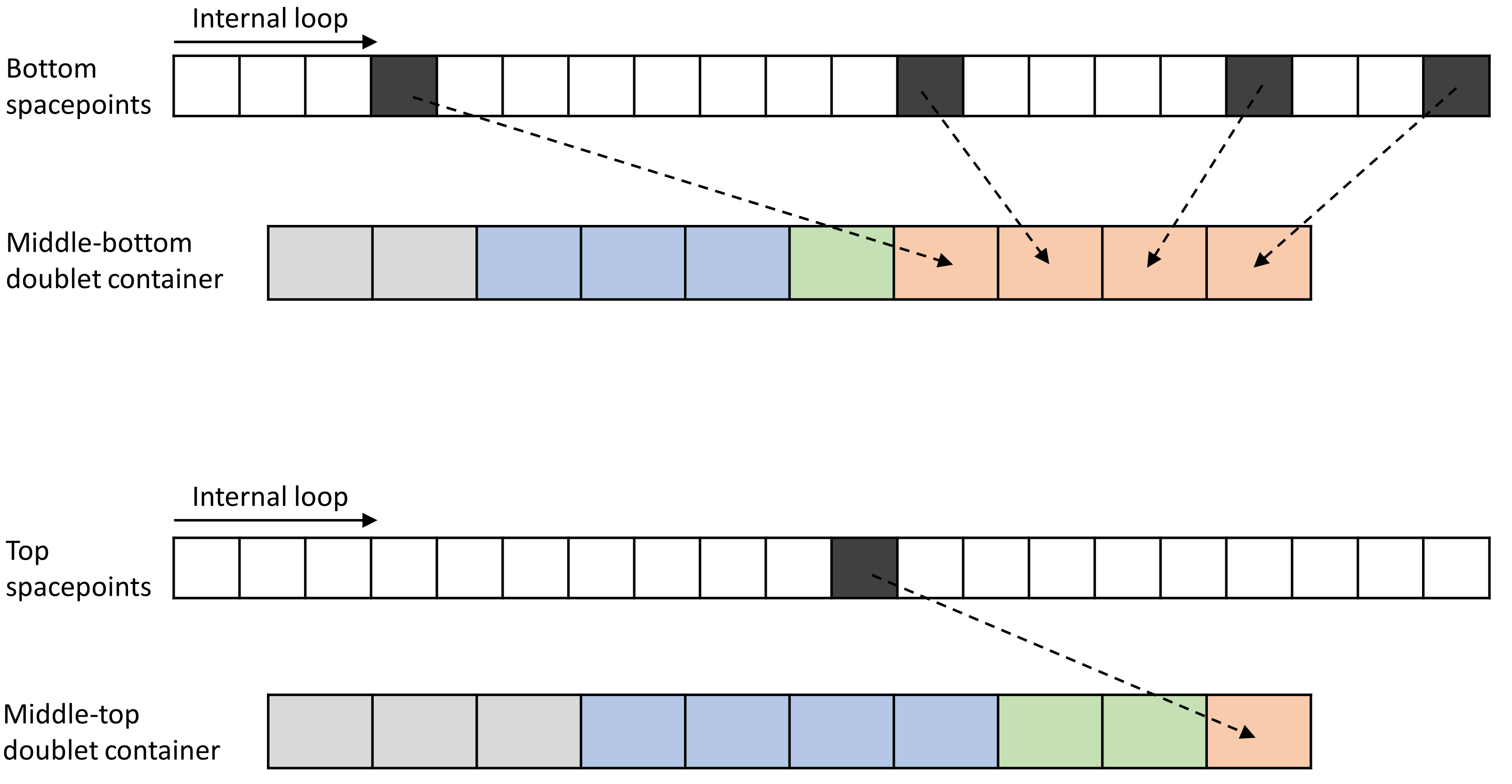
Kernel 2: Doublet Finding (cont.)

- Thread 3: 3rd middle spacepoint



Kernel 2: Doublet Finding (cont.)

- Thread 4: 4th middle spacepoint



Kernel 3: Triplet Counting

- Thread block setup
  - Num threads: 8×32
  - Num blocks (N_b):

$$N_b = \sum_{i=1}^{130} N_i$$ where N_i = num of mid-bot doublets of i-th bin/num of threads + 1

- Every thread (for a middle-bottom doublet) iterates over middle-top doublets, whose middle spacepoint is the same, to count triplets
- If the number of triplets for a middle-bottom doublet is more than zero, triplet counter object is recorded

```
if (n_triplets > 0) {
    auto pos = atomicAdd(&num_compat_mb_per_bin, 1);
    triplet_counter_per_bin[pos].n_triplets = n_triplets;
    triplet_counter_per_bin[pos].mid_bot_doublet = mid_bot_doublet;
}
```

Kernel 4: Triplet Finding

- Thread block setup
  - Num threads: 2×32
  - Num blocks (N_b):

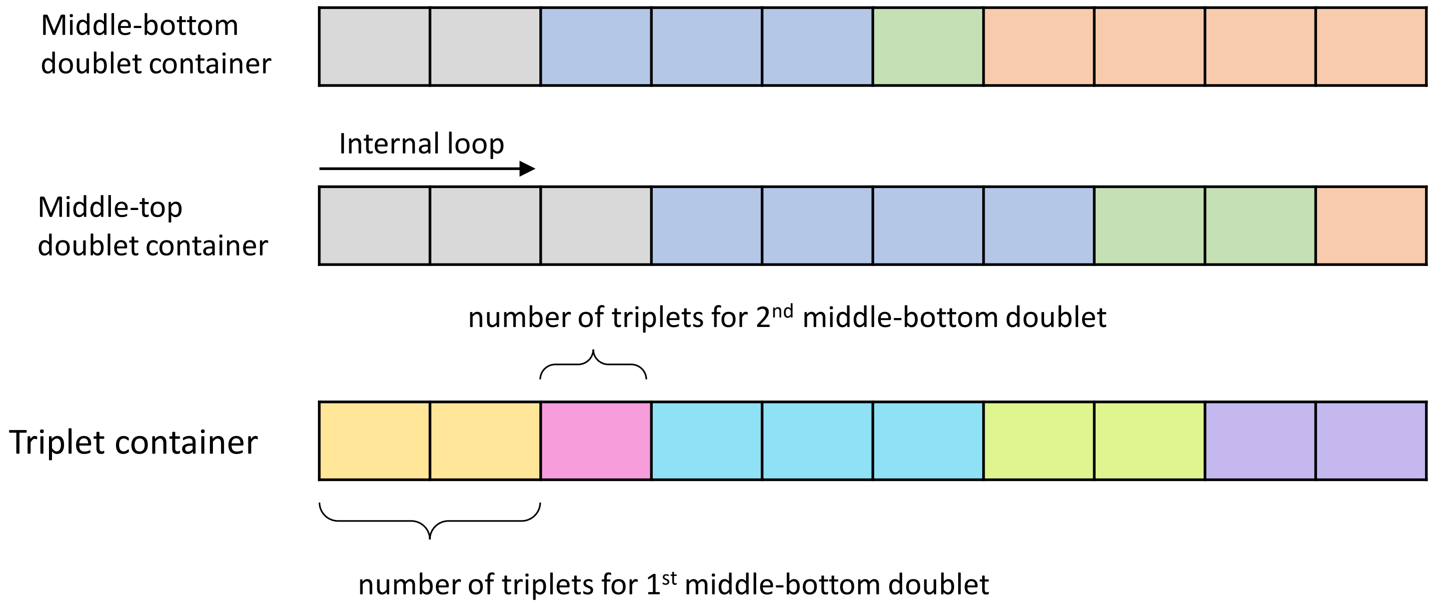
$$N_b = \sum_{i=1}^{130} N_i$$

where N_i = num of compatible mid-bot doublets of i-th bin/num of threads + 1

- Every thread (for a compatible middle-bottom doublet) iterates over middle-top doublets, whose middle spacepoint is the same, to record the triplet objects

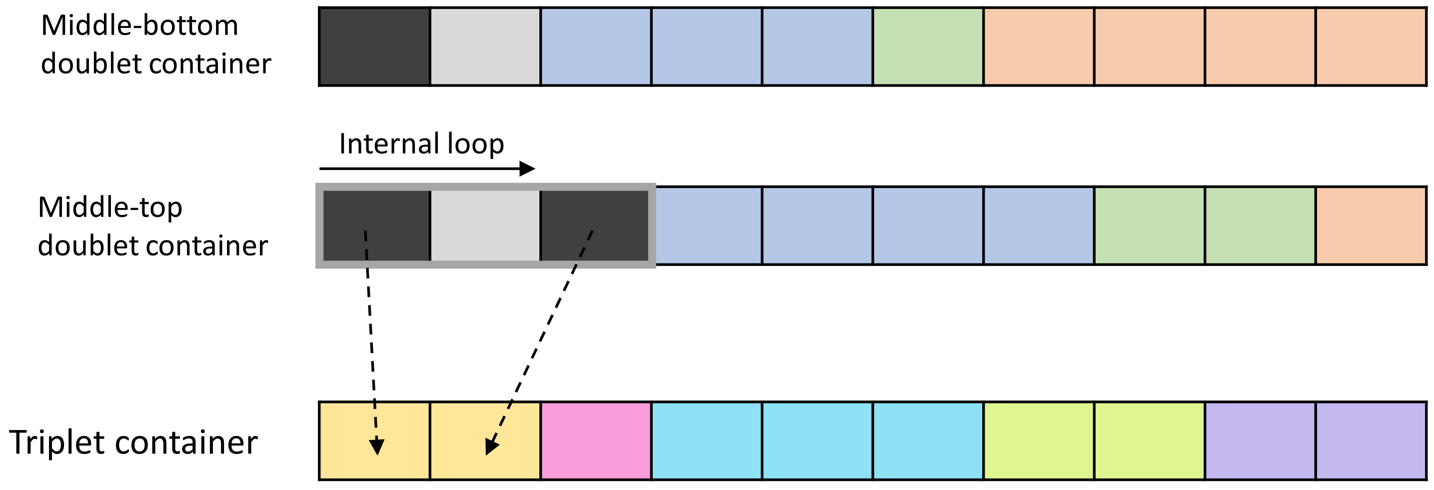
- The triplet counter is used to pre-assign the memory space for triplet objects

Kernel 4: Triplet Finding (cont.)



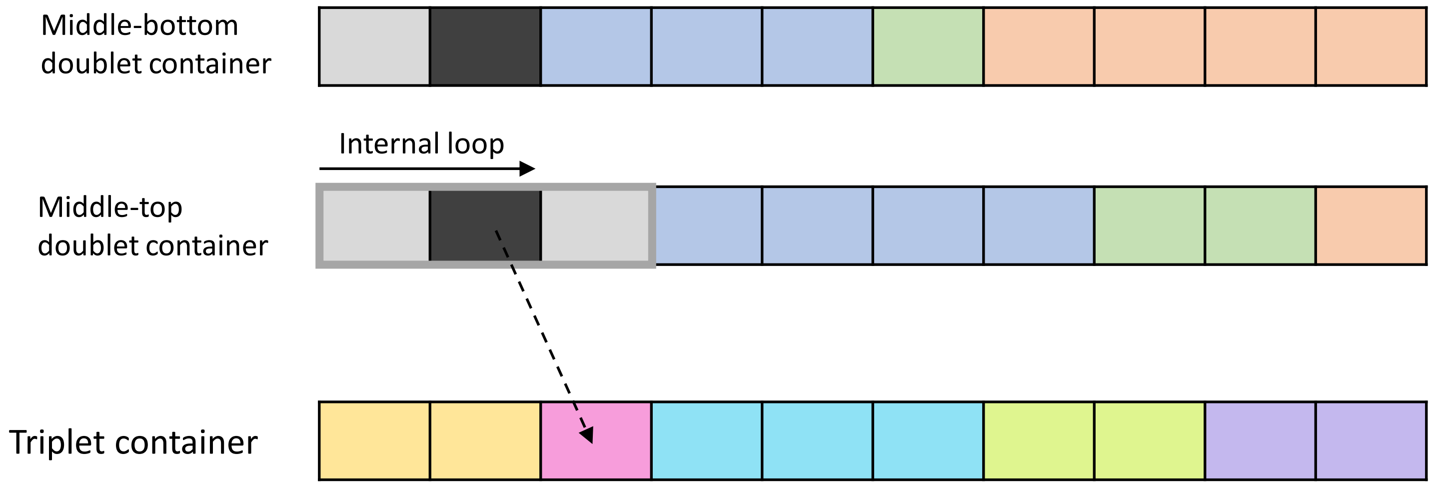
Kernel 4: Triplet Finding (cont.)

- Thread 1: 1st middle-bottom doublet



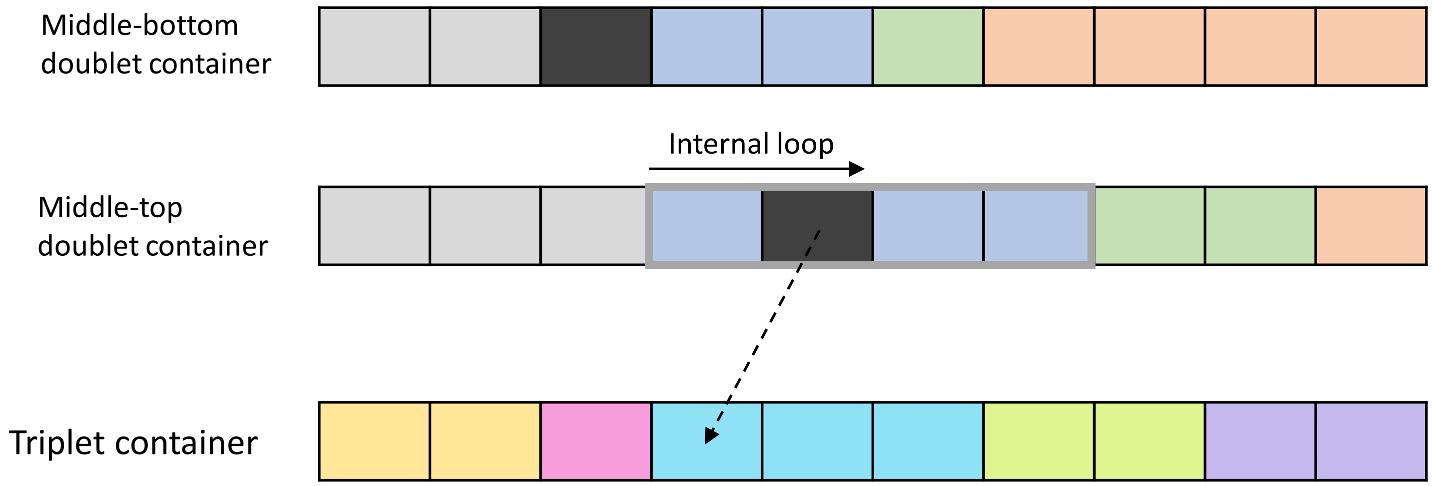
Kernel 4: Triplet Finding (cont.)

- Thread 2: 2nd middle-bottom doublet



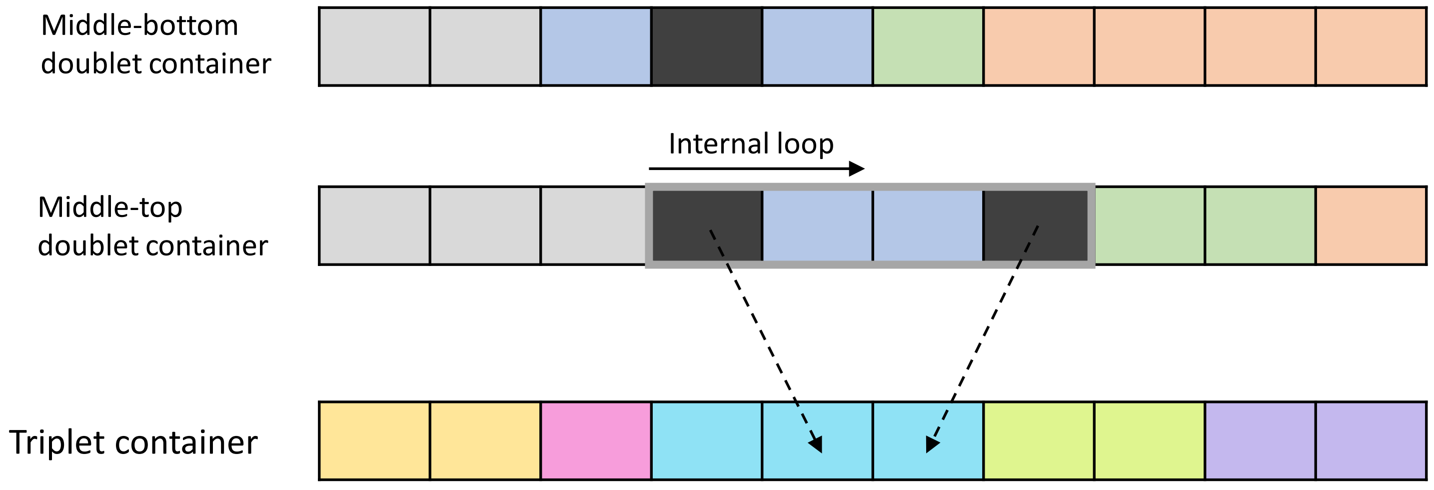
Kernel 4: Triplet Finding (cont.)

- Thread 3: 3rd middle-bottom doublet



Kernel 4: Triplet Finding (cont.)

- Thread 4: 4th middle-bottom doublet



Kernel 5: Weight Updating

- Thread block setup
  - Num threads: 2×32
  - Num blocks (N_b):

$$N_b = \sum_{i=1}^{130} N_i$$ where $N_i =$ num of triplets of i-th bin/num of threads + 1

- Every thread (for a triplet) iterates over triplets, whose middle-bottom doublet is the same, to update the weight of triplet

Kernel 6: Seed Selecting

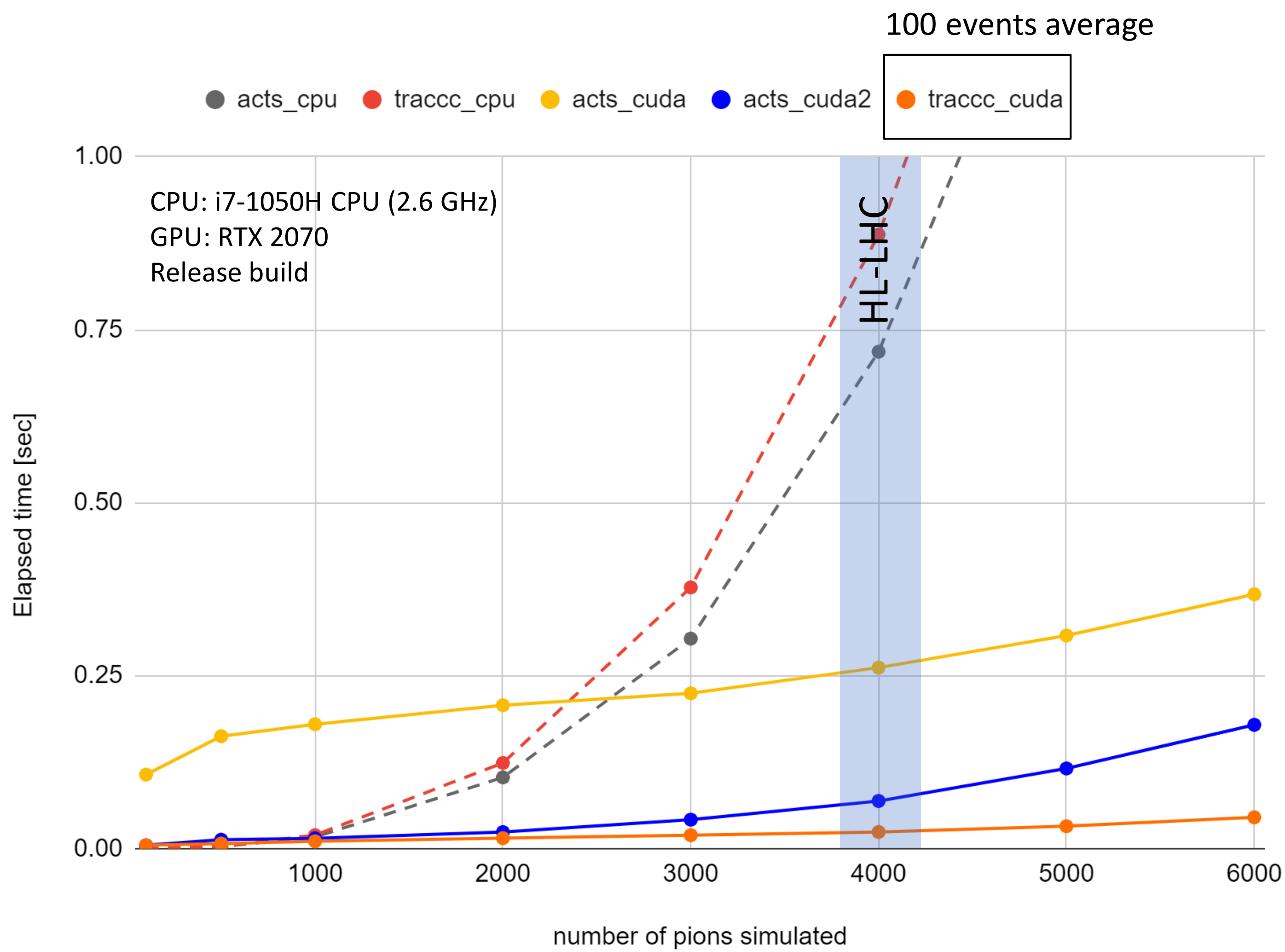
- Thread block setup
  - Num threads: 2×32
  - Num blocks (N_b):

$$N_b = \sum_{i=1}^{130} N_i$$

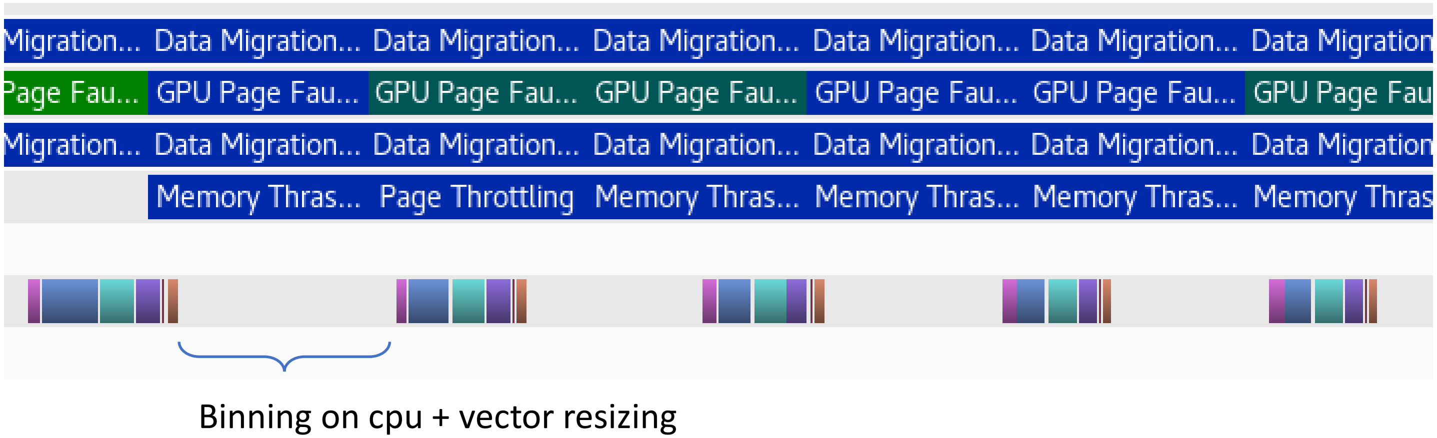
where N_i = num of compatible mid-bot doublets of i-th bin/num of threads + 1

- Seeds are filtered based on experiment-dependent parameters

Seed finding Timing Benchmark



Kernel timeline



- Spacepoint binning time on cpu ≈ seed finding time on gpu

Some issues found

- Multi-threading with NVIDIA mps server works normally
  - Have not checked the speedup with multi-threading yet since the most of wall time is occupied hit reading which makes harder it to make fair comparison

- It was found that V100 in cori server runs slightly slower than RTX 2070 of my laptop.

Summary and Outlooks

- The speedup over acts_cpu seed finding is about x29 for 4000 simulated pions
- Planning to profile what limits the performance on V100

What should be more implemented for GPU (any volunteer?):

1. Spacepoint binning with grid and axis
2. Track parameter estimation from seed