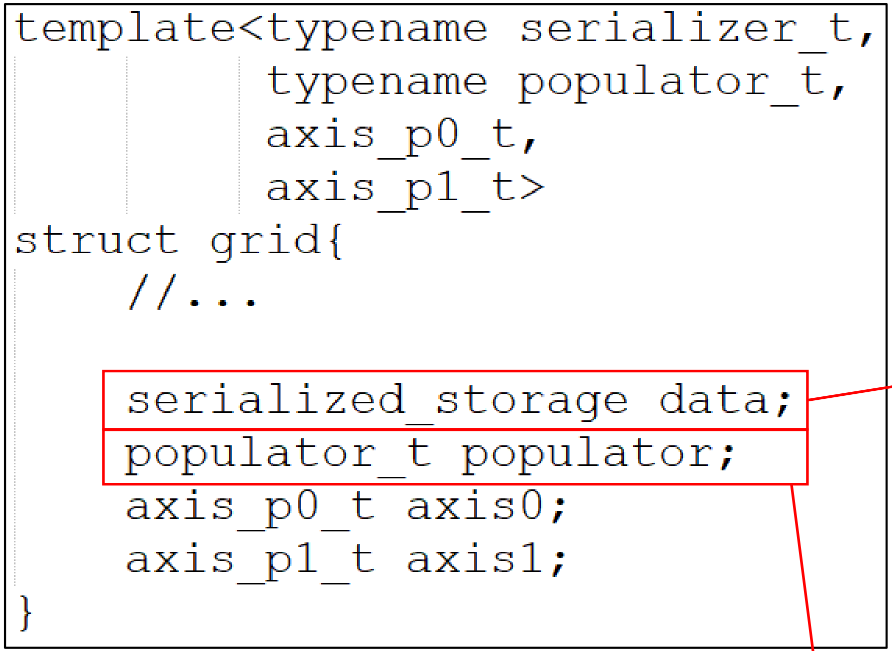
Detray Grid Development and Application to traccc

Beomki Yeo

Detray grid



```
template<typename serializer_t,
         typename populator_t,
         axis_p0_t,
         axis_p1_t>
struct grid{
    //...

    serialized_storage data;
    populator_t populator;
    axis_p0_t axis0;
    axis_p1_t axis1;
}
```

It’s where the data is stored

Type can be vecmem::vector or vecmem::device_vector depending on **populator** type

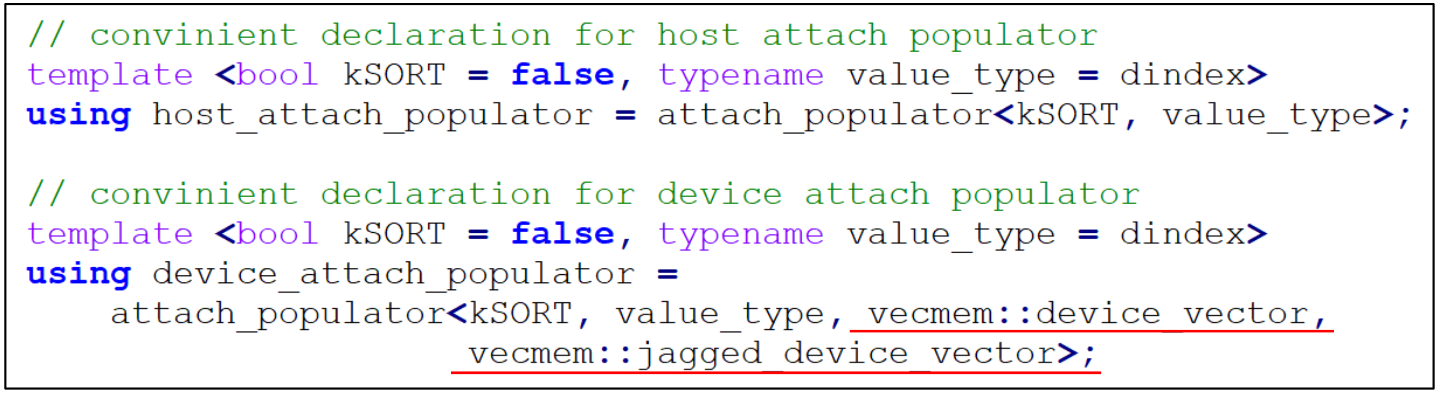
A tool to fill out the data storage

Currently three populator types:

- one object per bin (replace_populator)
- array of objects per bin (complete_populator)
- vector of objects per bin (attach_populator)

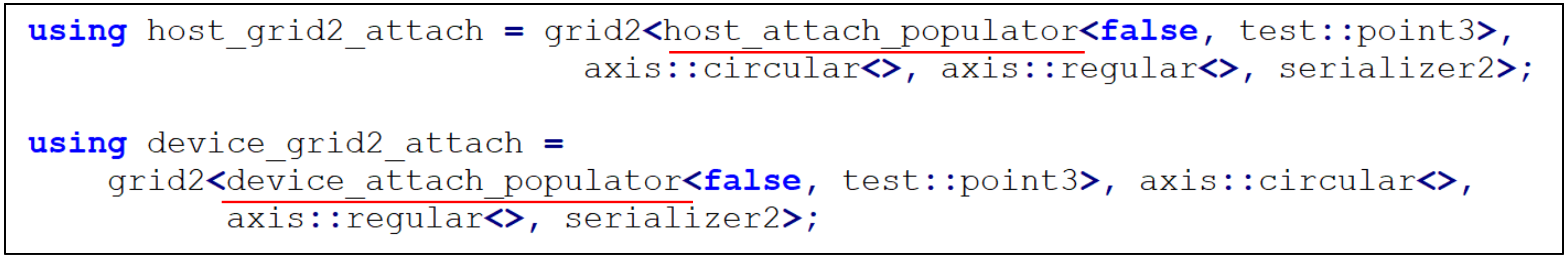
Populator w/ VecMem

- User can simply define populator for host or device



```
// convinient declaration for host attach populator
template <bool kSORT = false, typename value_type = dindex>
using host_attach_populator = attach_populator<kSORT, value_type>;

// convinient declaration for device attach populator
template <bool kSORT = false, typename value_type = dindex>
using device_attach_populator =
    attach_populator<kSORT, value_type, vecmem::device_vector,
                     vecmem::jagged_device_vector>;
```

- The side of grid is determined by populator type:



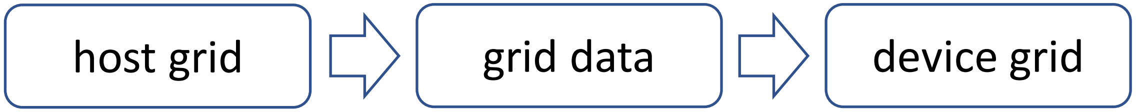
```
using host_grid2_attach = grid2<host_attach_populator<false, test::point3>,
                            axis::circular<>, axis::regular<>, serializer2>;

using device_grid2_attach =
    grid2<device_attach_populator<false, test::point3>, axis::circular<>,
          axis::regular<>, serializer2>;
```

How to use

- Follows the same semantics of EDM container in traccc

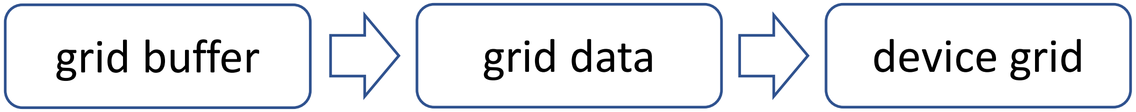
Use cases:

1. Transfer data in host grid to the device grid (for geometry/magnetic field)



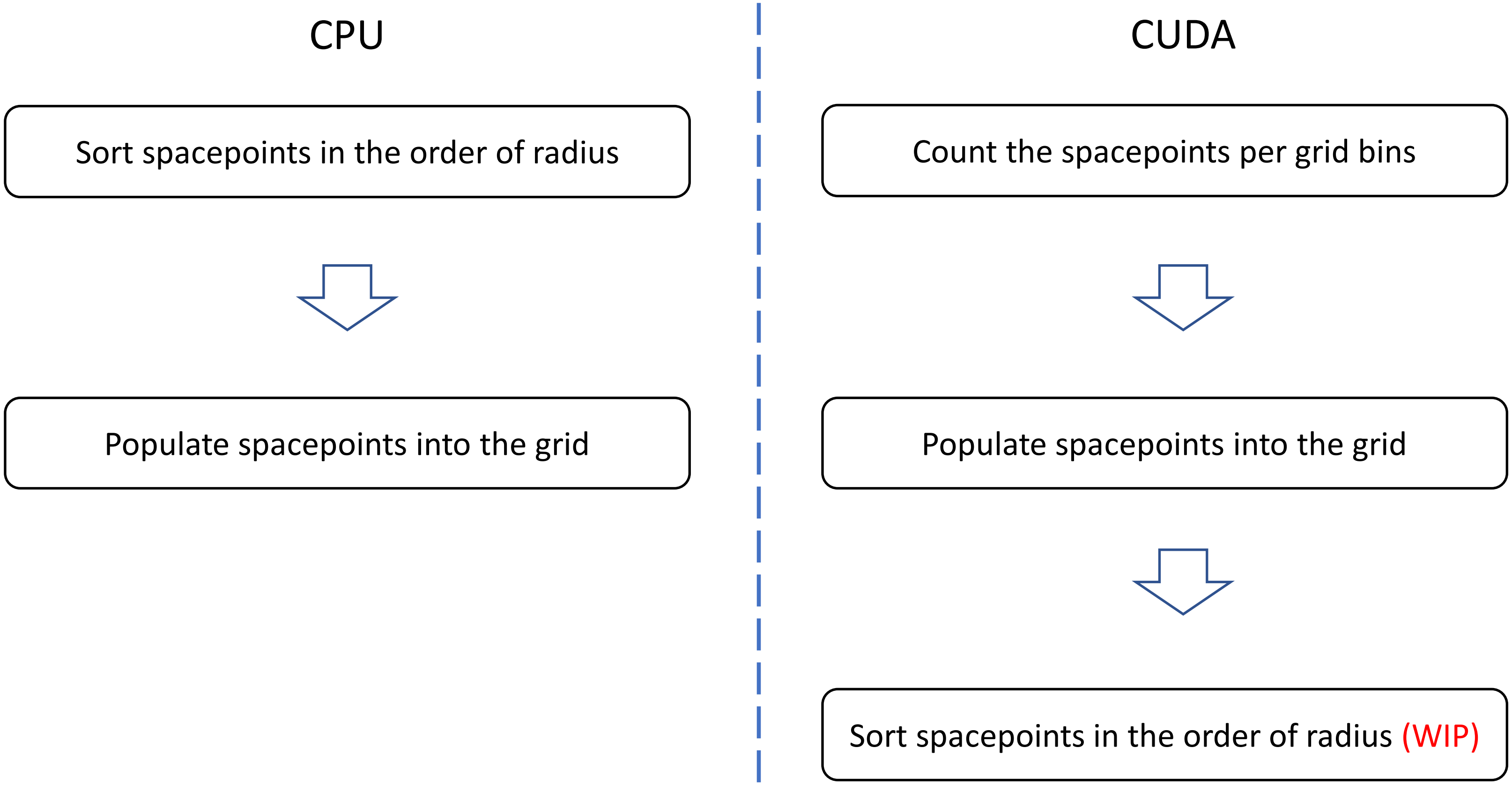
host grid → grid data → device grid

2. Pre-allocate memory space to the grid buffer, and fill out the device grid in the kernel (for spacepoint binning)



grid buffer → grid data → device grid

Application to Spacepoint Binning in traccc



CPU

Sort spacepoints in the order of radius

Populate spacepoints into the grid

CUDA

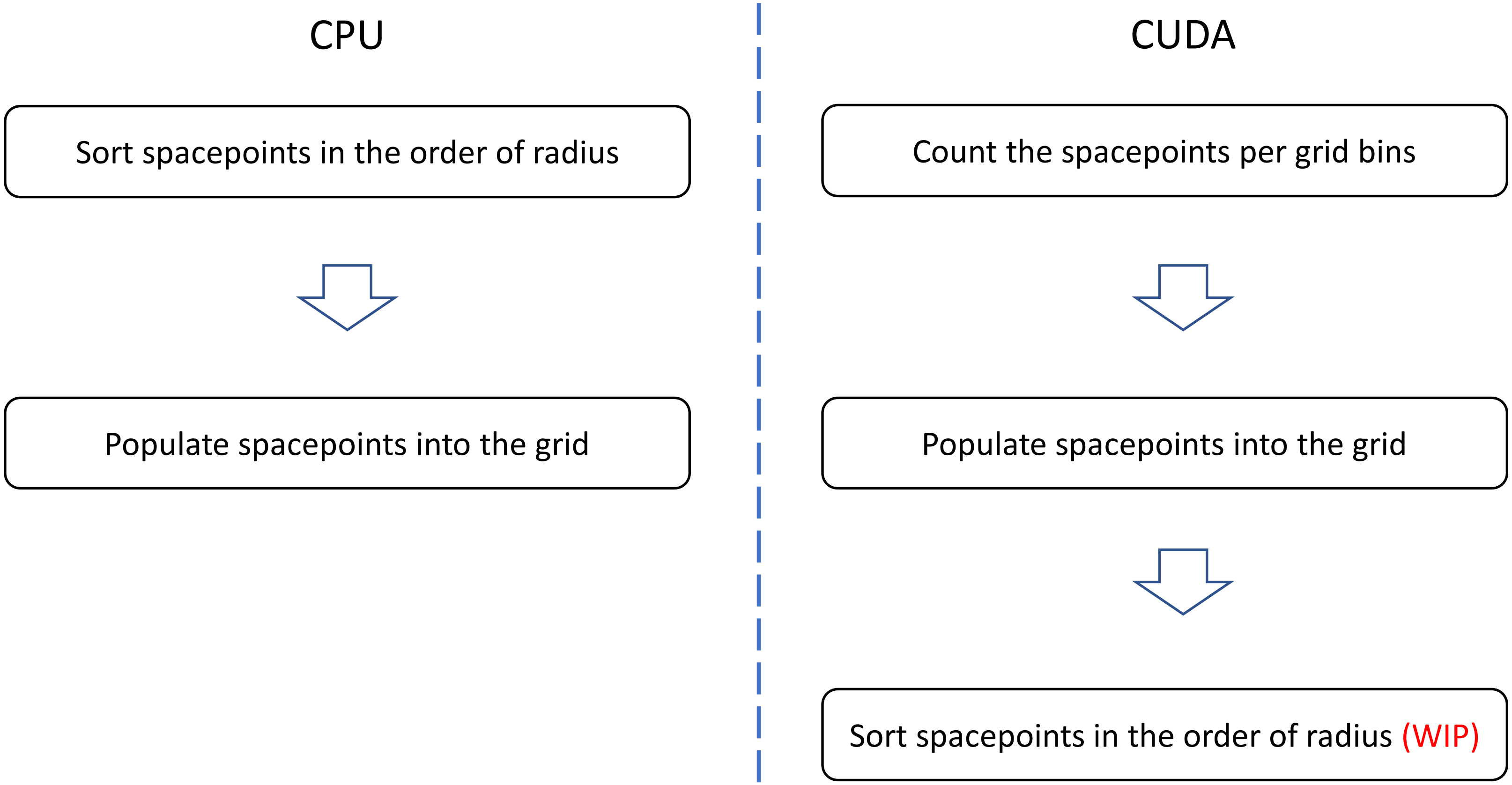
Count the spacepoints per grid bins

Populate spacepoints into the grid

Sort spacepoints in the order of radius (WIP)

- Sorting is not included in CUDA version yet, but the seed matching ratio is mostly 100% (I don’t know why :p)

Application to Spacepoint Binning in traccc



CPU

Sort spacepoints in the order of radius

Populate spacepoints into the grid

CUDA

Count the spacepoints per grid bins

Populate spacepoints into the grid

Sort spacepoints in the order of radius (WIP)

- Sorting is not included in CUDA version yet, but the seed matching ratio is mostly 100% (I don’t know why :p)

Spacepoint Binning and Speedup

- Couldn’t observe speedup in spacepoint binning

Single event (Very Noisy)							
ttbar average pileups	40	60	80	100	140	200	300
spacepoint_binning (cpu)	0.005537	0.006977	0.009355	0.008953	0.009116	0.008851	0.010269
spacepoint_binning (cuda)	0.007574	0.007689	0.008099	0.007973	0.008951	0.009488	0.010998
seed_finding (cpu)	0.018077	0.049172	0.094425	0.129714	0.190801	0.36138	1.137746
seed_finding (cuda)	0.009999	0.016878	0.016628	0.0184	0.015656	0.019822	0.046867

- Most of time seems occupied by the first kernel of counting the spacepoints for grid where I overused the atomic operation…

Tracking Chain Benchmark

- Speedup of seeding for tt-bar <200> is about 8-9
  - Actual speedup is higher than 10 considering the GPU warming-up
  - Losing some speedup due to spacepoint binning

Summed over 20 events							
average pileups	40	60	80	100	140	200	300
file reading (cpu)	2.83034	4.11401	5.75276	7.04383	9.59317	13.4349	19.8139
hit clusterization (cpu)	1.34521	1.5955	2.16827	2.43911	2.73963	3.06652	3.39017
spacepoint binning + seed finding (cpu)	0.617339	1.17063	2.15283	3.17346	6.00514	12.7521	31.8948
spacepoint binning + seed finding (cuda)	0.506827	0.578284	0.686695	0.783757	0.988713	1.47919	2.65675
track parameter estimation (cpu)	0.00201849	0.00387272	0.00772296	0.011835	0.0212835	0.040082	0.0729442
track parameter estimation (cuda)	0.010647	0.00999287	0.0132521	0.0145685	0.0190889	0.0263786	0.0417169

- Seeding results comparison
  1. Between traccc cpu and cuda: 99.9 – 100 %
  2. Between traccc cpu and Acts cpu: 97 – 100%