Detray Grid Development and Application to traccc

Beomki Yeo
Detray grid

```cpp
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

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

// convenient 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:

```cpp
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)

   ![Diagram](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)

   ![Diagram](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

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

- 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

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

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