Traccc cuda implementation

Beomki Yeo

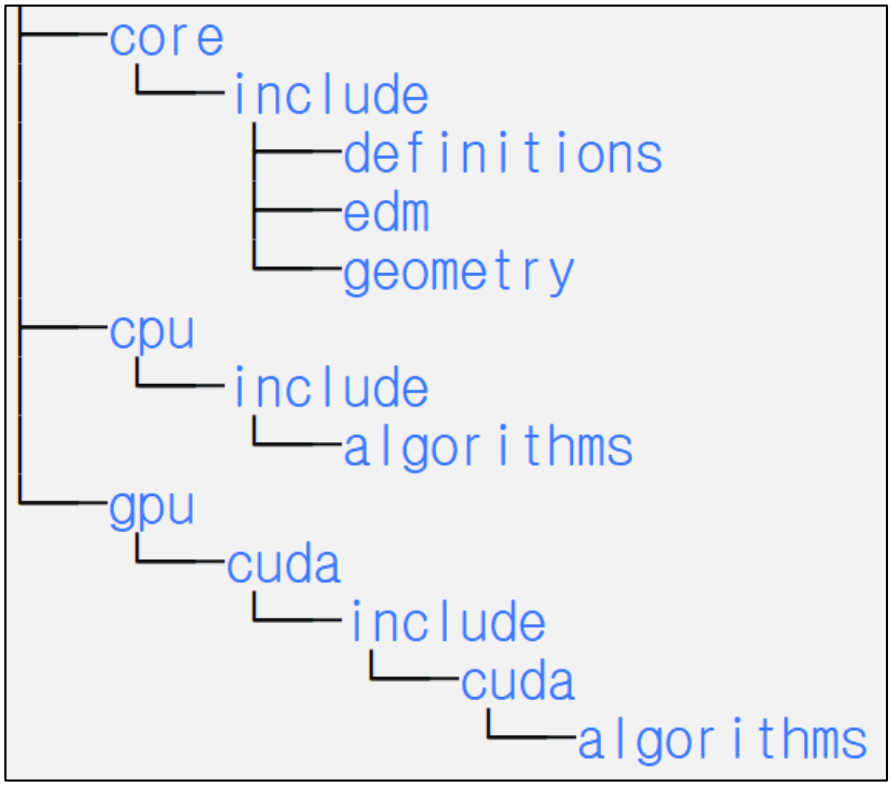
EDM

core/include/edm/cell.hpp

```
template< template< typename > class vector_t,
          template< typename > class jagged_vector_t >
class cell_container {

public:
    /// @name Type definitions
    /// @{

    /// Vector type used by the cell container
    template< typename T >
    using vector_type = vector_t< T >;
    /// Jagged vector type used by the cell container
    template< typename T >
    using jagged_vector_type = jagged_vector_t< T >;

    /// The cell module vector type
    using cell_module_vector = vector_type< cell_module >;
    /// The cell vector type
    using cell_vector = jagged_vector_type< cell >;

    /// @}

    /// Headers for all of the modules (holding cells) in the event
    cell_module_vector modules;
    /// All of the cells in the event
    cell_vector cells;

}; // class cell_container

/// Convenience declaration for the cell container type to use in host code
using host_cell_container =
    cell_container< vecmem::vector, vecmem::jagged_vector >;
/// Convenience declaration for the cell container type to use in device code
using device_cell_container =
    cell_container< vecmem::device_vector, vecmem::jagged_device_vector >;
```

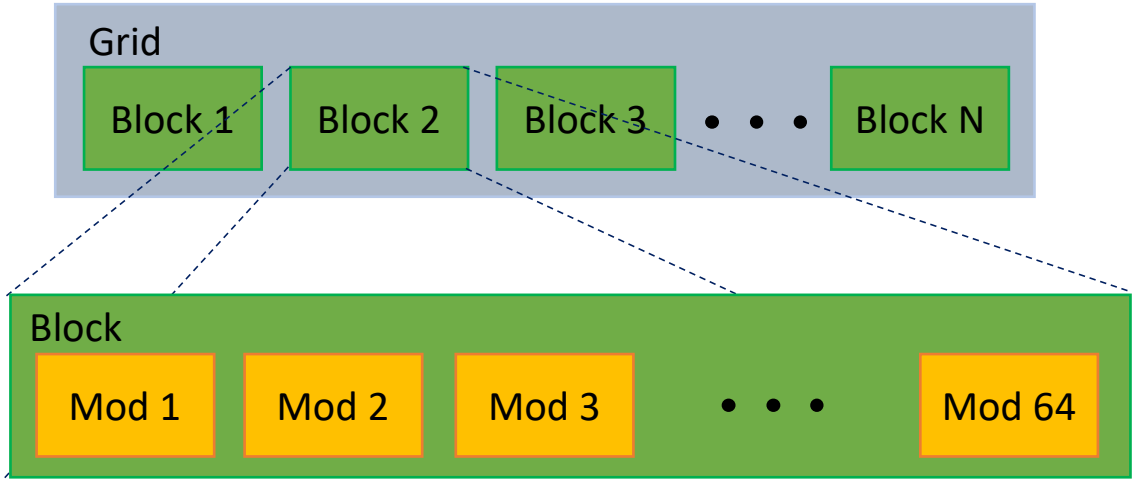
- Currently testing EDM format suggested by Atilla (PR23)
- Measurement and spacepoint have similar structure

Project structure



Thread block structure

- Each thread handles each module
- $\text{Number of threads per block} = 2 \times \text{Warp Size } (32) = 64$
- $\text{Number of blocks} = \text{Number of modules}/64 + 1$

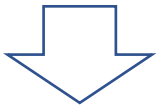


- Same structure was applied to sparse CCL, measurement creation, and spacepoint formation

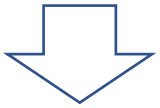
Implementation of Sparse CCL

Input

```
vector < vector < cell > >
```



Sparse CCL



Output

```
vector < num_of_clusters >
vector < vector < label > >
```

- Output was supposed to have `cluster` objects rather than label, i.e.:
  `vector<vector<vector< cell >>>`
- Since vector of vector of vector is not supported by vecmem, so decided to have labels as outputs.
- Example of ccl result of a module:

Num of unique labels (cluster)

4, { 1, 2, 1, 2, 2, 3, 4 ,4 }

Label per cell

Implementation of measurement creation (1)

- The number of measurements and the valid label indicies which can form measurements are counted first to set the size of measurement vector

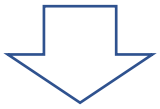
Input

```
vector < vector < cell > >
vector < num_of_clusters >
vector < vector < label > >
```

Num of unique labels (cluster)

4, { 1, 2, 1, 2, 2, 3, 4 ,4 }

Label per cell



Count measurement

Output

```
vector < num_of_measurements >
vector < vector < valid_label > >
```

Num of measurements

3, { 1, 2, 4}

Valid label

Implementation of measurement creation (2)

Input

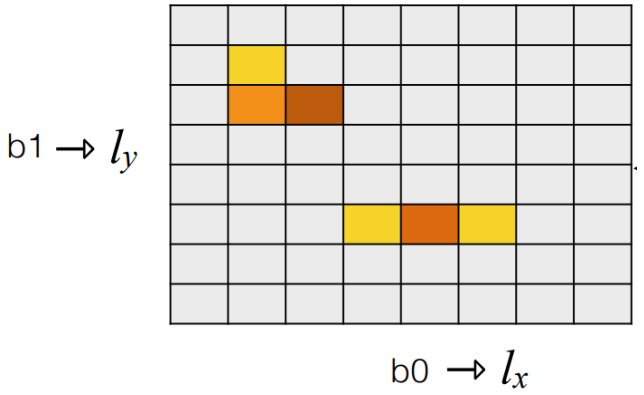
```
vector < vector < cell > >
vector < num_of_clusters >
vector < vector < label > >
vector < num_of_measurements >
vector < vector < valid_label > >
```

Measurement creation

Output

```
vector < vector < measurement > >
```

- Local coordinates of measurements are calculated



$$l = \frac{1}{\sum_{(i,j)} w_{(i,j)}} \sum_{(i,j)} w_{(i,j)} l_{(i,j)}$$

From Andi's slide

Spacepoint formation

Input

```
vector < vector < measurement > >
```

Spacepoint formation

Output

```
vector < vector < spacepoint > >
```

- Local coordinates of measurements are transformed into global coordinate

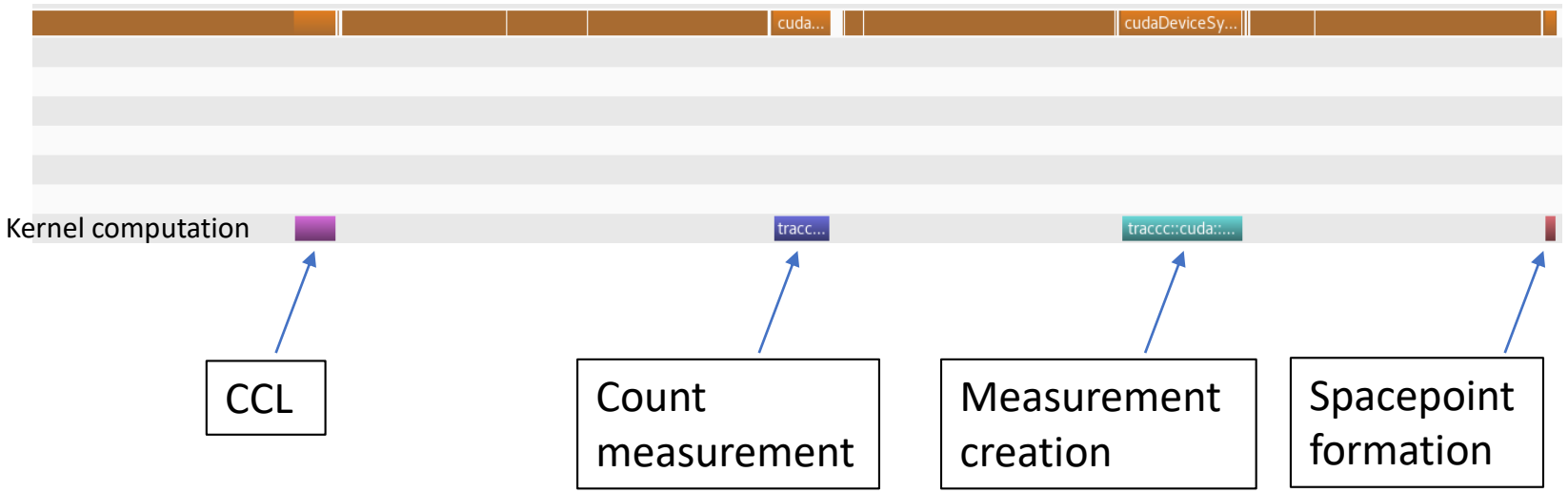
Timing breakdown

with vecmem

	original (cpu)	new edm (cpu)	new edm (cuda)
Reading time	3.701	4.024	5.602
CCL	0.226	0.221	0.018
MS creation	0.095	0.095	0.315
SP formation	0.038	0.041	0.110
Total	4.077	4.400	6.044

- The new and original edm show similar computing speed while the data reading with new edm takes slightly longer
- Reading data in cuda managed memory takes longer than in cpu memory
- CCL with cuda is faster, while measurement creation and spacepoint formation is slower with cuda

CUDA profiling



- Wall time is dominated by a number of memory allocations

```
traccc::host_spacepoint_container sp_container = {
     traccc::host_spacepoint_container::module_vector( n_modules, &mng_mr ),

     traccc::host_spacepoint_container::spacepoint_vector( n_modules, &mng_mr )
};
for ( int i=0; i< n_modules; ++i ){
    sp_container.spacepoints[i] =
        vecmem::vector< traccc::spacepoint >(ms_labels.counts[i]);
}
```

Summary

- The new edm works well with both cpu and cuda so far
- The cuda computing speed for CCL, measurement creation and spacepoint formation is limited by a number of memory allocation commands
- The codes can be found here (not PR yet)