Cuda Seed finding update in traccc

Beomki Yeo

Cpu implementation for Spacepoint grouping and seed finding

Brief introduction to seed finding

- Spacepoint Grouping
  - Based on configuration setup, spacepoints are grouped into the bins of 2-dimension (z and phi axis) grid
  - Spacepoints are converted into internal spacepoint in seed finding region

- Seed finding
  - For every internal spacepoints, other two compatible spacepoint in neighbor bins are found to form a triplet or seed
  - Algorithms:
    1. Doublet search
    2. Triplet search
    3. Calculate weights of triplets
    4. Filtering triplets based on weights

➢ Since the spacepoint grouping time is less than 1% of seed finding time, it does not have to be implemented on GPU urgently

ACTS spacepoint grouping

- Grouped spacepoints are stored in each bin of `Grid` class as a vector of internal spacepoints

```
template <typename external_spacepoint_t>
using SpacePointGrid =
    detail::Grid<std::vector<std::unique_ptr<
                    const InternalSpacePoint<external_spacepoint_t>>>,
                detail::Axis<detail::AxisType::Equidistant,
                             detail::AxisBoundaryType::Closed>,
                detail::Axis<detail::AxisType::Equidistant,
                             detail::AxisBoundaryType::Bound>>;
```

- The neighbor bins of each bin can be found with the iterator-based functions of `Axis` class

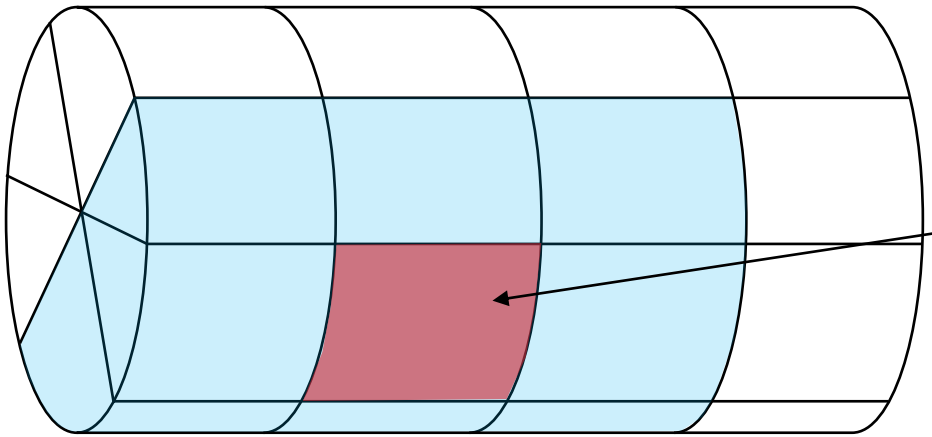
- There are some difficulties in passing the `Grid` objects to GPU device:
  - Need to put cuda architecture qualifiers (a.k.a __host__ __device__) for all functions of `Axis`, `Grid`, and etc, which are deeply implemented.
  - Index-based approach is favored in cuda code rather than iterator-based one
  - *Does not have a clever idea* in how to bring vecmem-based EDM into `Grid` objects

traccc spacepoint grouping – EDM for internal spacepoint

- PR35 : Port ACTS binning algorithm for CPU
- For traccc, adopted index-based iteration with vecmem EDM for internal spacepoints
  - `Grid` does not hold spacepoints but provides neighbor bin information on axis

```
// define spacepoint_grid
using spacepoint_grid = grid<traccc::axis<AxisBoundaryType::Closed>,
                             traccc::axis<AxisBoundaryType::Bound> >;
```

- `Internal_spacepoint_container` holds all internal spacepoints and neighbor bin indices for every bin



```
// header
std::array< size_t, 9 > neighbor_bin_indices
// item
vecmem::vector< internal_spacepoint > spacepoints
```

Seed finding in traccc

- Algorithm flow is same with Acts
- 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

Doublet definition

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

Spacepoint location

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

Circle information

```
struct lin_circle {
    float Zo;
    float cotTheta;
    float iDeltaR;
    float Er;
    float U;
    float V;
};
```

Triplet definition

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

Cuda implementation for seed finding

Github_branch

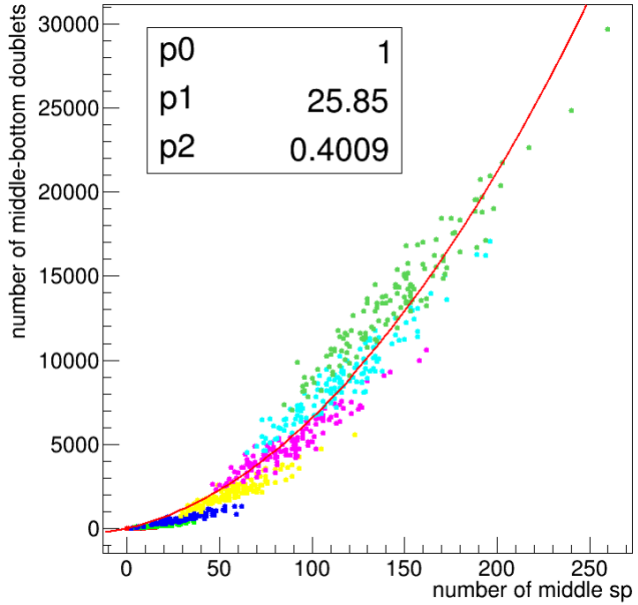
General comments on Cuda implementation

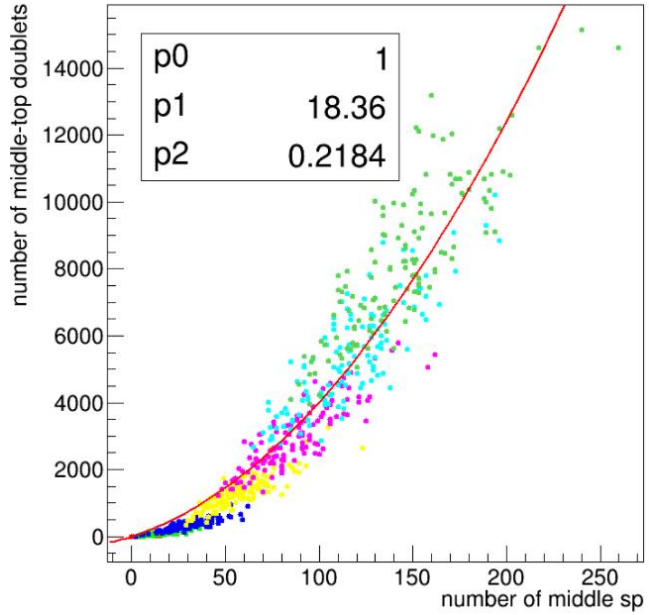
- Employs the same vecmem EDM (internal spacepoint, doublet, triplet and seed) used in cpu
  - Added several auxiliary vecmem containers

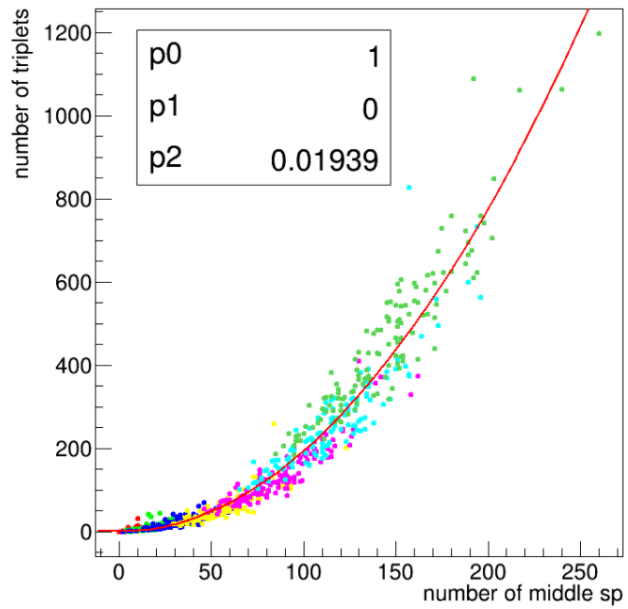
- Save gpu memory space as much as possible by sizing the vector of doublets and triplets (multiplets) with empirical values

- One cuda kernel for one event
  - Acts cuda plugin: one cuda kernel for one spacepoint bin

Estimation on mutiplet statistics







- Number of multiplets per bin as a function of number of middle spacepoints per bin is estimated using fatras simulation data
- The trend was fitted with second order polynomial which decides the size of doublet and triplet vector
- Safety factor of 1.5 was applied to the vector size

Estimation on mutiplet statistics

```
host_seed_collection operator()(host_internal_spacepoint_container& isp_container){

    // initialize multiplet container
    for (size_t i=0; i<isp_container.headers.size(); ++i){
        size_t n_spM = isp_container.items[i].size();
        size_t n_mid_bot_doublets = m_multiplet_config->get_mid_bot_doublets_size(n_spM);
        size_t n_mid_top_doublets = m_multiplet_config->get_mid_top_doublets_size(n_spM);
        size_t n_triplets = m_multiplet_config->get_triplets_size(n_spM);

        ///// Zero initialization
        doublet_counter_container.headers[i] = 0;
        mid_bot_container.headers[i] = 0;
        mid_top_container.headers[i] = 0;
        triplet_counter_container.headers[i] = 0;
        triplet_container.headers[i] = 0;
        compatseed_container.headers[i] = 0;

        doublet_counter_container.items[i].resize(n_spM);
        mid_bot_container.items[i].resize(n_mid_bot_doublets);
        mid_top_container.items[i].resize(n_mid_top_doublets);
        triplet_counter_container.items[i].resize(n_mid_bot_doublets);
        triplet_container.items[i].resize(n_triplets);
        compatseed_container.items[i].resize(n_triplets);

    }
```

mutiplet_config for the estimation of multiplet statistics (class is inherited for different detectors)

vecmem containers are resized for every event

Helper functions for multiplet compatibility

```
struct doublet_finding_helper{

static __CUDA_HOST_DEVICE__
bool isCompatible(const internal_spacepoint<spacepoint>& sp1,
                  const internal_spacepoint<spacepoint>& sp2,
                  const seedfinder_config& config,
                  bool bottom);


static __CUDA_HOST_DEVICE__
lin_circle transform_coordinates(const internal_spacepoint<spacepoint>& sp1,
                                 const internal_spacepoint<spacepoint>& sp2,
                                 bool bottom);

};
```

```
// Define a qualifier for cuda
#ifdef __CUDACC__
#include <cuda.h>
#include <cuda_runtime.h>
#define __CUDA_HOST_DEVICE__ inline __host__ __device__
#else
#define __CUDA_HOST_DEVICE__
#endif
```

```
struct triplet_finding_helper{

static __CUDA_HOST_DEVICE__
bool isCompatible(const internal_spacepoint<spacepoint>& spM,
                  const lin_circle& lb,
                  const lin_circle& lt,
                  const seedfinder_config& config,
                  const scalar& iSinTheta2,
                  const scalar& scatteringInRegion2,
                  scalar& curvature,
                  scalar& impact_parameter);
};
```

- Compatibility check function and conformal transformation is defined for both cpu and cuda

Seed finding flow in traccc

```
void operator()(host_internal_spacepoint_container& isp_container,
                host_seed_collection& seeds){

    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,
                                   m_seedfilter_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,
                                  compatseed_container,
                                  m_mr);
```

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. Experiment-dependent seed filter based on weights (cpu)

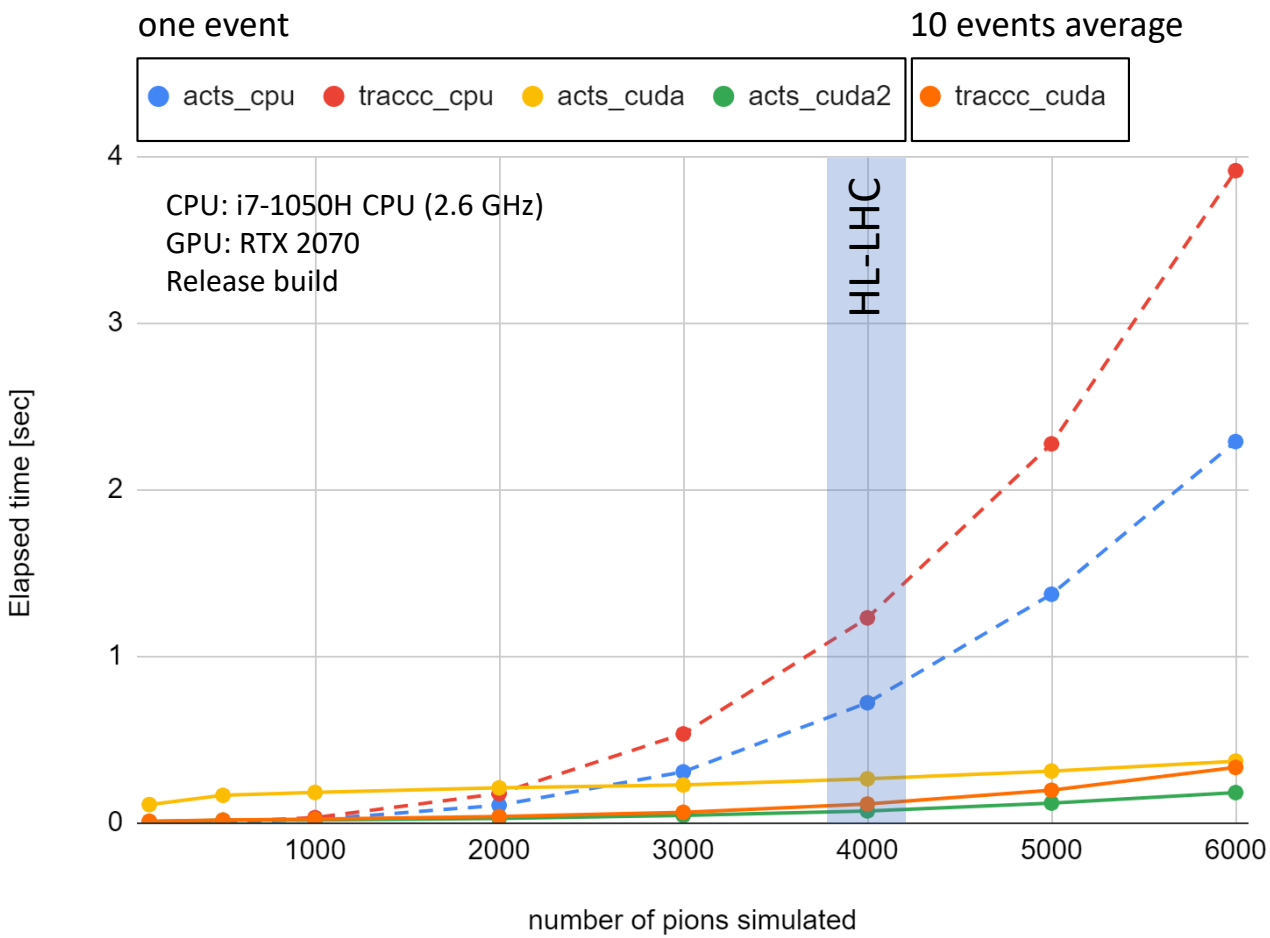
Thread-block configuration

```
void operator()(host_internal_spacepoint_container& isp_container,
                host_seed_collection& seeds){

    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,
                                   m_seedfilter_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,
                                  compatseed_container,
                                  m_mr);
```

One Kernel for one event

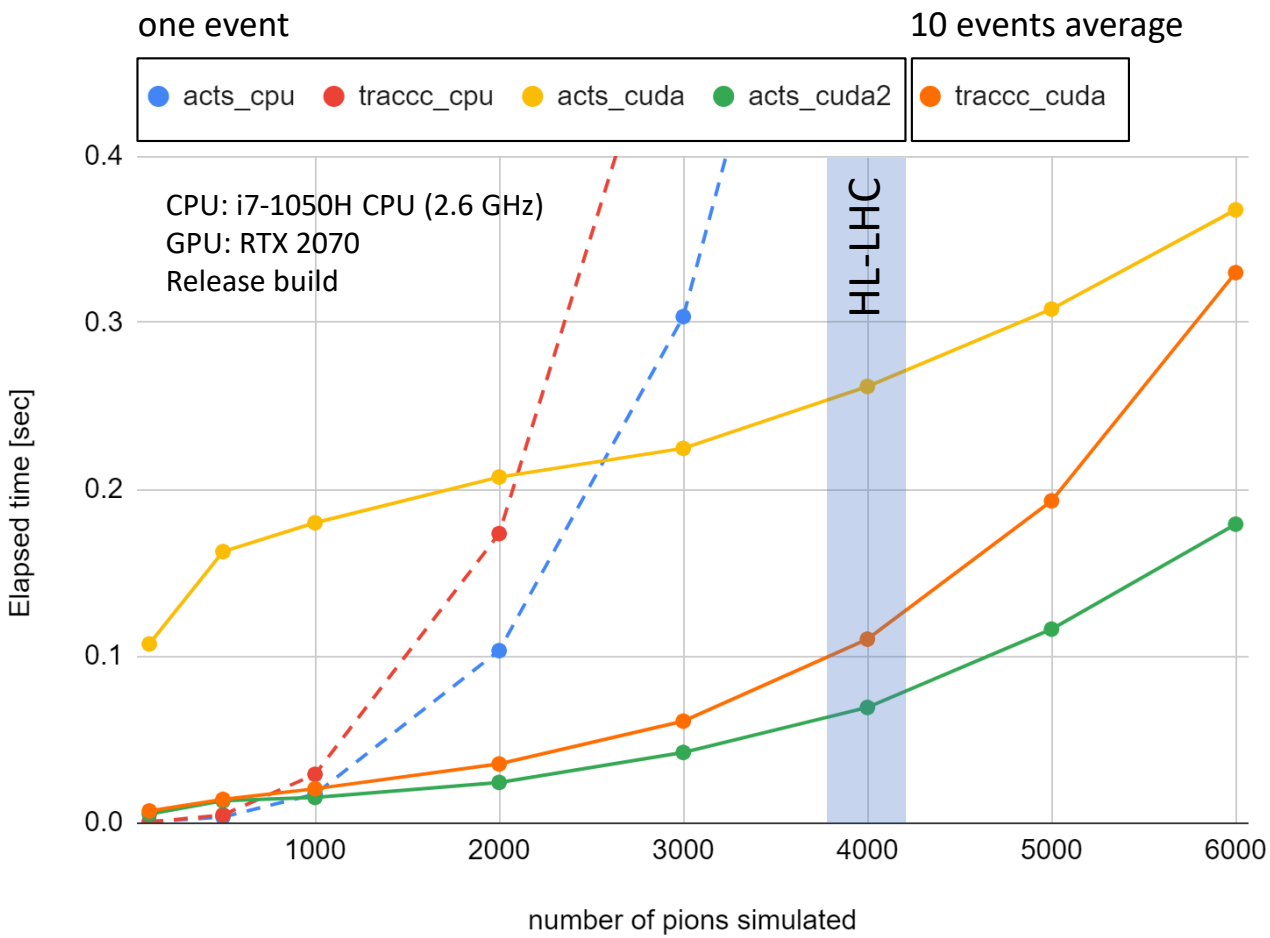
- Doublet counting & finding
  - One block for one spacepoint bin → 130 blocks in total
  - One thread for int(# mid_spacepoints/block_size) + 1
- Triplet counting & finding
  - One block for one spacepoint bin
  - One thread for int(# mid_bot_doublets/block_size) + 1
- Weight updating
  - One block for one spacepoint bin
  - One thread for int(# triplets/block_size) + 1

Seed finding Throughputs



- traccc_cpu is 1.5-2 times slower than acts_cpu:
  - Index-based algorithm might be slower than iterator-based one
  - vecmem library?
- traccc_cuda is 1.5-2 tims slower than acts_cuda2 (by Attila)

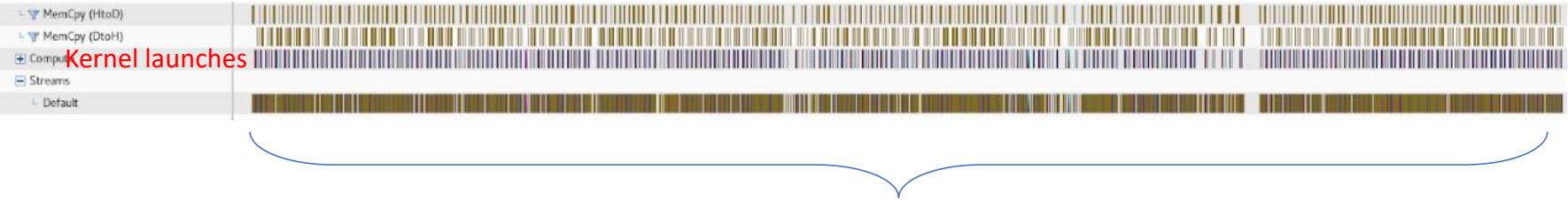
Seed finding Throughputs



- traccc_cpu is 1.5-2 times slower than acts_cpu:
  - Index-based algorithm might be slower than iterator-based one
  - vecmem library?
- traccc_cuda is 1.5-2 tims slower than acts_cuda2 (by Attila)

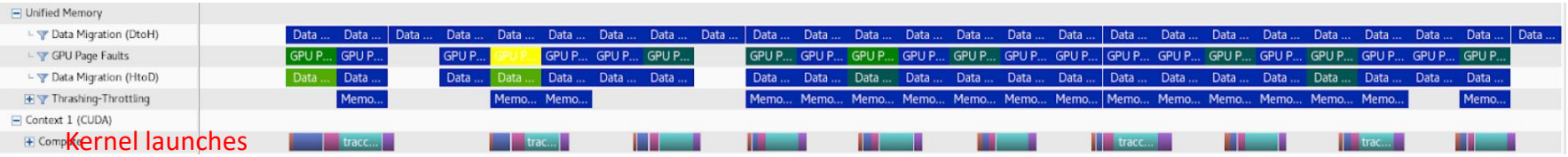
Cuda Timeline

acts_cuda2



1 event

traccc_cuda



10 events

- For traccc_cuda, the first kernel launches take more time than later ones (overhead issue?)
- For traccc_cuda, half of wall time is spent in resizing vecmem container, which calls a bunch of cudaMalloc and cudafree

Outlooks

- Need to think about how to avoid resizing issue
  - Using the same vector size for all events can be considered (by averaging the number of middles spacepoints across events)

- Need to modify the algorithms to speed up the computation in kernels
  - Current algorithms have too much branching conditions

- For the safety factor of 1.5, 99.0 - 99.9 % seed matching rate in HL-LHC size data is confirmed

```
event 0 seed matching rate: 0.999608
event 1 seed matching rate: 0.993572
event 2 seed matching rate: 0.99883
event 3 seed matching rate: 0.999208
event 4 seed matching rate: 0.99824
event 5 seed matching rate: 0.998257
event 6 seed matching rate: 0.997405
event 7 seed matching rate: 0.990413
event 8 seed matching rate: 0.999049
event 9 seed matching rate: 0.999416
```

- 100% of seed matching rate could not be obtained for some large events even with enough safety factor
  - Minor bugs exist or fast math mode was not turned off correctly

- Investigate the multi thread performance with nvidia mps